PAGNON Alexis
SANCHEZ Adam

Compte Rendu du Projet d'Architecture Orientée Services

Ce rapport contient les résultats de nos manipulations de minikube, ainsi que les captures d'écran du projet sur la mise en place d'une architecture de type EDA

Date : 16/01/2026

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES HAUTS-DE-FRANCE

Université Polytechnique HAUTS-DE-FRANCE

Sommaire

Questions Kubernetes

Gestion de minikube

Vérifiez que minikube pointe correctement vers le moteur docker ?

```
alexis@alexis-VirtualBox:~$ minikube docker-env
export DOCKER_TLS_VERIFY="1"
export DOCKER_HOST="tcp://192.168.58.2:2376"
export DOCKER_CERT_PATH="/home/alexis/.minikube/certs"
export MINIKUBE_ACTIVE_DOCKERD="minikube"

# To point your shell to minikube's docker-daemon, run:
# eval $(minikube -p minikube docker-env)
```

Quels sont les addons actuellement installés ?

```
alexis@alexis-VirtualBox:~$ 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)

Installez celle qui vous semble intéressante, pourquoi ?

On peut commencer par installer l'addon "dashboard" qui nous permettra de gérer les clusters via une interface web, et qui sera demandé d'utiliser dans les questions suivantes.

```
alexis@alexis-VirtualBox:~$ minikube addons enable dashboard
💡  dashboard est un addon maintenu par Kubernetes. Pour toute question, contactez minikube sur GitHub
.
Vous pouvez consulter la liste des mainteneurs de minikube sur : https://github.com/kubernetes/minikub
e/blob/master/OWNERS
    ▪ Utilisation de l'image docker.io/kubernetesui/dashboard:v2.7.0
    ▪ Utilisation de l'image docker.io/kubernetesui/metrics-scraper:v1.0.8
💡  Certaines fonctionnalités du tableau de bord nécessitent le module complémentaire metrics-server.
Pour activer toutes les fonctionnalités, veuillez exécuter :

        minikube addons enable metrics-server

🌟  Le module 'dashboard' est activé
```

Lister les profils actifs sous minikube avec toutes ses caractéristiques ?

```
alexis@alexis-VirtualBox:~$ minikube profile list --detailed
```

PROFILE	DRIVER	RUNTIME	IP	PORT	VERSION	STATUS	NODES	ACTIVE PROFILE	ACTIVE KUBECONTEXT
minikube	docker	docker	192.168.58.2	8443	v1.34.0	OK	1	*	*

Quels sont les profils en cours ?

Le profil en cours est le profil "minikube". Ce profil est le profil par défaut de minikube, et peut être re-sélectionné avec :

```
alexis@alexis-VirtualBox:~$ minikube profile default
✅  Le profil de minikube a été défini avec succès sur minikube
```

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

On peut créer un profil au démarrage de minkube avec "minikube start -p dev". Chaque profil représente un cluster Kubernetes, indépendant des autres et avec sa propre configuration.

```
alexis@alexis-VirtualBox:~$ minikube start -p dev
😄  [dev] minikube v1.37.0 sur Linuxmint 21.2 (vbox/amd64)
    ▪ MINIKUBE_ACTIVE_DOCKERD=minikube
✨  Choix automatique du pilote docker

❗  L'allocation de mémoire demandée de 2968MiB ne laisse pas de place pour la surcharge système (mémo
ire système totale : 2968MiB). Vous pouvez rencontrer des problèmes de stabilité.
💡  Suggestion : Start minikube with less memory allocated: 'minikube start --memory=2968mb'

📌  Utilisation du pilote Docker avec le privilège root
👍  Démarrage du nœud "dev" primary control-plane dans le cluster "dev"
🚜  Extraction de l'image de base v0.0.48...
🔥  Création de docker container (CPU=2, Memory=2968Mo) ...
🐳  Préparation de Kubernetes v1.34.0 sur Docker 28.4.0...
🔗  Configuration de bridge CNI (Container Networking Interface)...
🔎  Vérification des composants Kubernetes...
❗  L'exécution de "docker container inspect dev --format={{.State.Status}}" a pris un temps inhabitue
llement long : 4.475114748s
💡  Le redémarrage du service docker peut améliorer les performances.
    ▪ Utilisation de l'image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Modules activés: storage-provisioner, default-storageclass
🏄  Terminé ! kubectl est maintenant configuré pour utiliser "dev" cluster et espace de noms "default"
 par défaut.
```

Si on réaffiche la liste des profils, on voit bien que le nouveau profil "dev" a été ajouté :

```
alexis@alexis-VirtualBox:~$ minikube profile list --detailed
```

PROFILE	DRIVER	RUNTIME	IP	PORT	VERSION	STATUS	NODES	ACTIVE PROFILE	ACTIVE KUBECONTEXT
dev	docker	docker	192.168.49.2	8443	v1.34.0	OK	1		*
minikube	docker	docker	192.168.58.2	8443	v1.34.0	OK	1	*	

On peut aussi changer de profil actif avec “minikube profil dev” :

```
alexis@alexis-VirtualBox:~$ minikube profile dev
✅  Le profil de minikube a été défini avec succès sur dev
alexis@alexis-VirtualBox:~$ minikube profile list --detailed
```

PROFILE	DRIVER	RUNTIME	IP	PORT	VERSION	STATUS	NODES	ACTIVE PROFILE	ACTIVE KUBECONTEXT
dev	docker	docker	192.168.49.2	8443	v1.34.0	OK	1	*	*
minikube	docker	docker	192.168.58.2	8443	v1.34.0	OK	1		

Afficher le statut de minikube ?

On peut utiliser “minikube status” pour afficher le statut du cluster local:

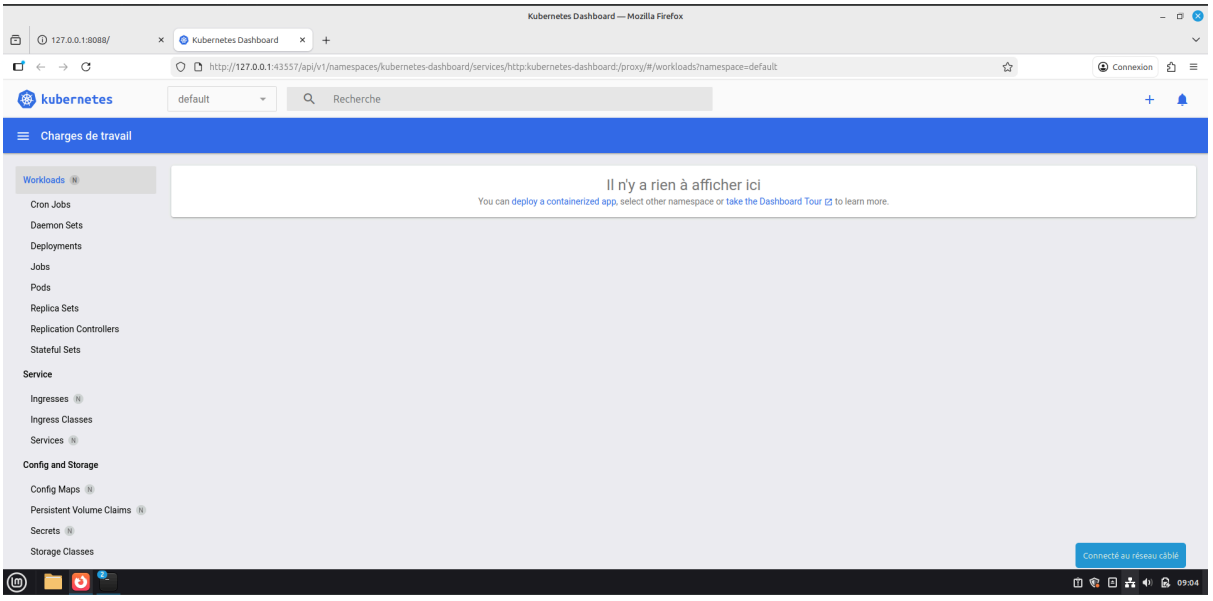
```
alexis@alexis-VirtualBox:~$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
docker-env: in-use
```

Comment puis-je accéder au dashboard de minikube ?

On utilise “minikube dashboard” (en ayant auparavant activé l’addon) :

```
linuxmint@linuxmint-VirtualBox:~$ minikube dashboard
🤔  Vérification de l'état du tableau de bord...
🚀  Lancement du proxy...
🤔  Vérification de l'état du proxy...
🎉  Ouverture de http://127.0.0.1:43557/api/v1/namespaces/kubernetes-dashboard/services
/http:kubernetes-dashboard:/proxy/ dans votre navigateur par défaut...
```

On obtient le dashboard web :



Qu'est ce que le DashBoard, que présente-t-il ?

Le dashboard Kubernetes est une interface web qui permet de visualiser et gérer un cluster Kubernetes. Il présente l'état des ressources (pods, services, déploiements, nœuds), leurs statuts, logs, événements et permet d'effectuer des actions simples comme déployer ou supprimer des applications.

Lister les nœuds d'un profil ?

On sélectionne d'abord notre profil avec "minikube profile <nom>" et on affiche les noeuds avec "kubectl get nodes" :

```
linuxmint@linuxmint-VirtualBox:~$ kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   13m   v1.34.0
```

On peut également utiliser "minikube node list", mais cette commande fournit moins d'informations :

```
alexis@alexis-VirtualBox:~$ minikube node list
minikube        192.168.58.2
```

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

On peut ajouter un noeud à un profil avec "minikube node add" :

```
alexis@alexis-VirtualBox:~$ minikube node add
😄  Ajout du nœud m02 au cluster minikube en tant que [worker]
❗  Le cluster a été créé sans aucun CNI, l'ajout d'un nœud peut provoquer un réseau inopérant.
👍  Démarrage du nœud "minikube-m02" worker dans le cluster "minikube"
🚜  Extraction de l'image de base v0.0.48...
🔥  Création de docker container (CPU=2, Memory=2200Mo) ...
🐳  Préparation de Kubernetes v1.34.0 sur Docker 28.4.0...
🔎  Vérification des composants Kubernetes...
🏄  m02 a été ajouté avec succès à minikube !
alexis@alexis-VirtualBox:~$
```

Par défaut, le profil "minikube" est ciblé. On aurait pu cibler un profil en ajoutant "-p <profil>":

```
alexis@alexis-VirtualBox:~$ minikube node add -p minikube
😄  Ajout du nœud m03 au cluster minikube en tant que [worker]
👍  Démarrage du nœud "minikube-m03" worker dans le cluster "minikube"
🚜  Extraction de l'image de base v0.0.48...
🔥  Création de docker container (CPU=2, Memory=2200Mo) ...-
```

On peut vérifier que le noeud a bien été créé :

```
alexis@alexis-VirtualBox:~$ kubectl get nodes
NAME           STATUS   ROLES           AGE     VERSION
minikube       Ready    control-plane   13d     v1.34.0
minikube-m02   Ready    <none>          2m24s   v1.34.0
```

On peut supprimer notre nouveau noeud avec “minikube node delete <nom>” :

```
alexis@alexis-VirtualBox:~$ minikube node delete minikube-m02
🔥  Suppression de noeuds minikube-m02 de cluster minikube
✋  Nœud d'arrêt "minikube-m02" ...
🛑  Mise hors tension du profil "minikube-m02" via SSH…
🔥  Suppression de "minikube-m02" dans docker...
💀  Le nœud minikube-m02 a été supprimé avec succès.
```

Consulter les logs de minikube, comment faire ?

On utilise “minikube logs” :

```
linuxmint@linuxmint-VirtualBox:~$ minikube logs

==> Audit <==
```

COMMAND	ARGS	PROFILE	USER	VERSION	START TIME	END TIME
docker-env		minikube	linuxmint	v1.37.0	06 Jan 26 08:49 CET	
start		minikube	linuxmint	v1.37.0	06 Jan 26 08:50 CET	
start		minikube	linuxmint	v1.37.0	06 Jan 26 08:50 CET	
docker-env		minikube	linuxmint	v1.37.0	06 Jan 26 08:52 CET	
start		minikube	linuxmint	v1.37.0	06 Jan 26 08:52 CET	06 Jan 26 08:55 CET
docker-env		minikube	linuxmint	v1.37.0	06 Jan 26 08:53 CET	
docker-env		minikube	linuxmint	v1.37.0	06 Jan 26 08:55 CET	06 Jan 26 08:55 CET
addons	list	minikube	linuxmint	v1.37.0	06 Jan 26 08:56 CET	06 Jan 26 08:56 CET
addons	enable configure dashboard	minikube	linuxmint	v1.37.0	06 Jan 26 09:02 CET	
addons	enable dashboard	minikube	linuxmint	v1.37.0	06 Jan 26 09:02 CET	06 Jan 26 09:02 CET
dashboard		minikube	linuxmint	v1.37.0	06 Jan 26 09:03 CET	

```
==> Dernier démarrage <==
Log file created at: 2026/01/06 08:52:17
Running on machine: linuxmint-VirtualBox
Binary: Built with gc go1.24.6 for linux/amd64
Log line format: [IWEF]mmdd hh:mm:ss.uuuuuu threadid file:line] msg
I0106 08:52:17.400378    2464 out.go:360] Setting OutFile to fd 1 ...
I0106 08:52:17.400559    2464 out.go:413] isatty.IsTerminal(1) = true
```

Gestion des pods et services sous kubernetes

Lister les images actuellement en exécution dans votre environnement minikube.

On peut obtenir les images actuellement en exécution dans notre environnement minikube de plusieurs manières avec “kubectl get pods”, la plus simple mais la moins lisible est celle-ci:

```
linuxmint@linuxmint-VirtualBox:~$ kubectl get pods -A -o jsonpath="{..image}"
registry.k8s.io/coredns/coredns:v1.12.1 registry.k8s.io/coredns/coredns:v1.12.1 registry.k8s.io/etcd:3.6.4-0 registry.k8s.io/etc
d:3.6.4-0 registry.k8s.io/kube-apiserver:v1.34.0 registry.k8s.io/kube-apiserver:v1.34.0 registry.k8s.io/kube-controller-manager:
v1.34.0 registry.k8s.io/kube-controller-manager:v1.34.0 registry.k8s.io/kube-proxy:v1.34.0 registry.k8s.io/kube-proxy:v1.34.0 re
gistry.k8s.io/kube-scheduler:v1.34.0 registry.k8s.io/kube-scheduler:v1.34.0 gcr.io/k8s-minikube/storage-provisioner:v5 gcr.io/k8
s-minikube/storage-provisioner:v5 docker.io/kubernetesui/metrics-scraper:v1.0.8@sha256:76049887f07a0476dc93efc2d3569b9529bf982b2
2d29f356092ce206e98765c kubernetesui/metrics-scraper@sha256:76049887f07a0476dc93efc2d3569b9529bf982b22d29f356092ce206e98765c doc
ker.io/kubernetesui/dashboard:v2.7.0@sha256:2e500d29e9d5f4a086b908eb8dfe7ecac57d2ab09d65b24f588b1d449841ef93 kubernetesui/dashbo
ard@sha256:2e500d29e9d5f4a086b908eb8dfe7ecac57d2ab09d65b24f588b1d449841ef93linuxmint@linuxmint-VirtualBox:~$
```

Mais pour une meilleure lisibilité, celle-là est mieux:

```
linuxmint@linuxmint-VirtualBox:~$ kubectl get pods -A -o wide
NAMESPACE              NAME                                         READY   STATUS    RESTARTS   AGE     IP             NODE       NOMINATED NODE   READINESS GATES
kube-system            coredns-66bc5c9577-vs8k4                     1/1     Running   0          17m     10.244.0.2     minikube   <none>           <none>
kube-system            etcd-minikube                                1/1     Running   0          17m     192.168.49.2   minikube   <none>           <none>
kube-system            kube-apiserver-minikube                      1/1     Running   0          17m     192.168.49.2   minikube   <none>           <none>
kube-system            kube-controller-manager-minikube             1/1     Running   0          17m     192.168.49.2   minikube   <none>           <none>
kube-system            kube-proxy-276m6                             1/1     Running   0          17m     192.168.49.2   minikube   <none>           <none>
kube-system            kube-scheduler-minikube                      1/1     Running   0          17m     192.168.49.2   minikube   <none>           <none>
kube-system            storage-provisioner                          1/1     Running   0          16m     192.168.49.2   minikube   <none>           <none>
kubernetes-dashboard   dashboard-metrics-scraper-77bf4d6c4c-rvkqx   1/1     Running   0          9m13s   10.244.0.4     minikube   <none>           <none>
kubernetes-dashboard   kubernetes-dashboard-855c9754f9-cvmdd        1/1     Running   0          9m13s   10.244.0.5     minikube   <none>           <none>
```

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

Pour lancer une image nginx dans un pod, il faut utiliser cette commande:

```
linuxmint@linuxmint-VirtualBox:~$ kubectl run nginx-pod --image=nginx --restart=Never
pod/nginx-pod created
linuxmint@linuxmint-VirtualBox:~$ kubectl get pods
NAME        READY   STATUS              RESTARTS   AGE
nginx-pod   0/1     ContainerCreating   0          10s
```

(on vérifie que notre pod a bien été lancé avec la seconde commande)

Pour lancer une image nginx dans un déploiement en mode impératif, il faut utiliser cette commande:

```
linuxmint@linuxmint-VirtualBox:~$ kubectl create deployment nginx --image=nginx
deployment.apps/nginx created
linuxmint@linuxmint-VirtualBox:~$ kubectl get deployments
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   1/1     1            1           40s
```

(de même que tout à l'heure, la seconde commande permet de vérifier que nous avons bien lancer notre deployment)

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

On va créer notre service avec la commande "kubectl expose pod" :

```
alexis@alexis-VirtualBox:~$ kubectl expose pod nginx-pod --port=80
  --target-port=80 --type=NodePort --name=nginx-service
service/nginx-service exposed
```

On utilise le type NodePort pour exposer le service et activer l'accès au service depuis l'extérieur du cluster.

Visualiser les informations du pod et du service. ?

Pour obtenir les informations des tous les pods et service:

```
alexis@alexis-VirtualBox:~$ kubectl get pods -o wide
NAME                     READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATES
nginx-66686b6766-k7rdb   1/1     Running   0          13m   10.244.0.8   minikube   <none>           <none>
nginx-pod                1/1     Running   0          12m   10.244.0.9   minikube   <none>           <none>
alexis@alexis-VirtualBox:~$ kubectl get services -o wide
NAME            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE     SELECTOR
kubernetes      ClusterIP   10.96.0.1        <none>        443/TCP        13d     <none>
nginx-service   NodePort    10.110.217.104   <none>        80:30905/TCP   2m30s   run=nginx-pod
```

Par contre si l'on ne veut qu'un seul pod, il faut utiliser "kubectl describe pod <nom>" :

```
linuxmint@linuxmint-VirtualBox:~$ kubectl describe pod nginx-pod
Name:             nginx-pod
Namespace:        default
Priority:         0
Service Account:  default
Node:             minikube/192.168.49.2
Start Time:       Tue, 06 Jan 2026 09:15:18 +0100
Labels:           run=nginx-pod
Annotations:      <none>
Status:           Running
IP:               10.244.0.6
IPs:
  IP:  10.244.0.6
Containers:
  nginx-pod:
    Container ID:   docker://4981e8b9b5c696dd2e98ead8c637ab7c134c2f74df86837133b49d0bf198a6dd
    Image:          nginx
    Image ID:       docker-pullable://nginx@sha256:ca871a86d45a3ec6864dc45f014b11fe626145569ef0e74deaffc95a3b15b430
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Tue, 06 Jan 2026 09:15:35 +0100
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-2lvjf (ro)
```

Pareil pour le service:

```
linuxmint@linuxmint-VirtualBox:~$ 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:                       10.102.169.179
IPs:                      10.102.169.179
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  30294/TCP
Endpoints:                10.244.0.6:80
Session Affinity:         None
External Traffic Policy:  Cluster
Internal Traffic Policy:  Cluster
Events:                   <none>
```

Obtenir l'url du service ?

Pour obtenir l'url du service nginx, on utilise cette commande:

```
linuxmint@linuxmint-VirtualBox:~$ minikube service nginx-service --url
http://192.168.49.2:30294
```

Exécuter le service dans lynx ou Firefox ou tout autre browser. ?

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

Pour lancer une commande bash directement dans un conteneur, on utilise la commande “kubectl exec <nom du pod> -- <commande>” :

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

```
alexis@alexis-VirtualBox:~$ kubectl exec nginx-pod -- hostname
nginx-pod
```

Lister les logs du conteneur nginx du pod ?

```
linuxmint@linuxmint-VirtualBox:~$ kubectl logs nginx-pod
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform con
figuration
/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/defau
lt.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/def
ault.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
2026/01/06 08:15:36 [notice] 1#1: using the "epoll" event method
2026/01/06 08:15:36 [notice] 1#1: nginx/1.29.4
2026/01/06 08:15:36 [notice] 1#1: built by gcc 14.2.0 (Debian 14.2.0-19)
2026/01/06 08:15:36 [notice] 1#1: OS: Linux 6.14.0-36-generic
2026/01/06 08:15:36 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2026/01/06 08:15:36 [notice] 1#1: start worker processes
2026/01/06 08:15:36 [notice] 1#1: start worker process 30
2026/01/06 08:15:36 [notice] 1#1: start worker process 31
2026/01/06 08:15:36 [notice] 1#1: start worker process 32
2026/01/06 08:15:36 [notice] 1#1: start worker process 33
10.244.0.1 - - [06/Jan/2026:09:22:56 +0000] "GET / HTTP/1.1" 200 615 "-" "Mozilla/5.0
(X11; Linux x86_64; rv:145.0) Gecko/20100101 Firefox/145.0" "-"
2026/01/06 09:22:56 [error] 30#30: *1 open() "/usr/share/nginx/html/favicon.ico" faile
d (2: No such file or directory), client: 10.244.0.1, server: localhost, request: "GET
 /favicon.ico HTTP/1.1", host: "192.168.49.2:30294", referrer: "http://192.168.49.2:30
294/"
10.244.0.1 - - [06/Jan/2026:09:22:56 +0000] "GET /favicon.ico HTTP/1.1" 404 153 "http:
//192.168.49.2:30294/" "Mozilla/5.0 (X11; Linux x86_64; rv:145.0) Gecko/20100101 Firef
ox/145.0" "-"
```

Projet d'une architecture de type EDA

Tous les fichiers de notre projet peuvent être retrouvés sur le github suivant :
https://github.com/alexis-pagnon/Projet_architecture_EDA

Vous y retrouverez également le fichier "README.md" qui décrit toutes les commandes à effectuer pour utiliser nos différents scripts, permettant de construire/lancer/publier/arrêter nos images Docker ainsi que de déployer/arrêter notre architecture avec Kubernetes.
Tous les manifestes Kubernetes se trouvent également dans le dossier "kubernetes" de notre projet.

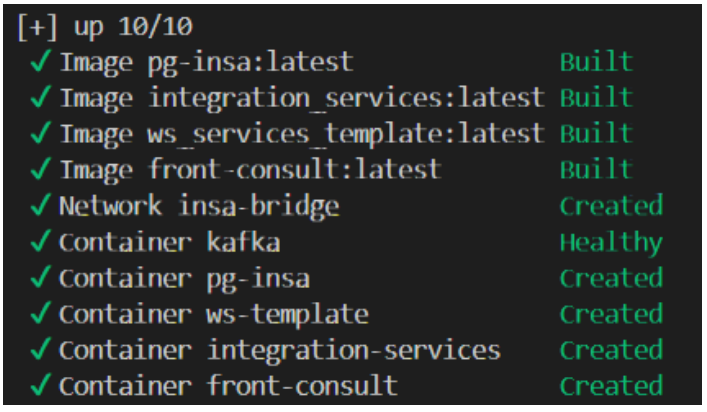
Pour notre projet, nous avons les services :

- kafka : bus Kafka avec les deux topics "students-request" et "students-response"
- pg-insa : base de données PostgreSQL issue du code de TD fournit
- ws-template : service REST API fait avec Spring Boot, inspiré du code de TD fournit, et modifié pour écrire sur le topic "students-request" du bus Kafka les requêtes reçues par l'API.
- front-consult : interface utilisateur web faite avec Node.js, qui permet d'envoyer des requêtes à l'API de ws-template et qui lit les réponses sur le topic "students-response" du bus Kafka.
- integration-services : interface entre le bus Kafka et pg-insa, permettant de lire les requêtes sur le topic "students-request", d'effectuer les requêtes SQL correspondantes, et d'écrire si besoin une réponse sur le topic "students-response". Ce service permet également de créer les topics de Kafka lors du premier déploiement des services.

Docker

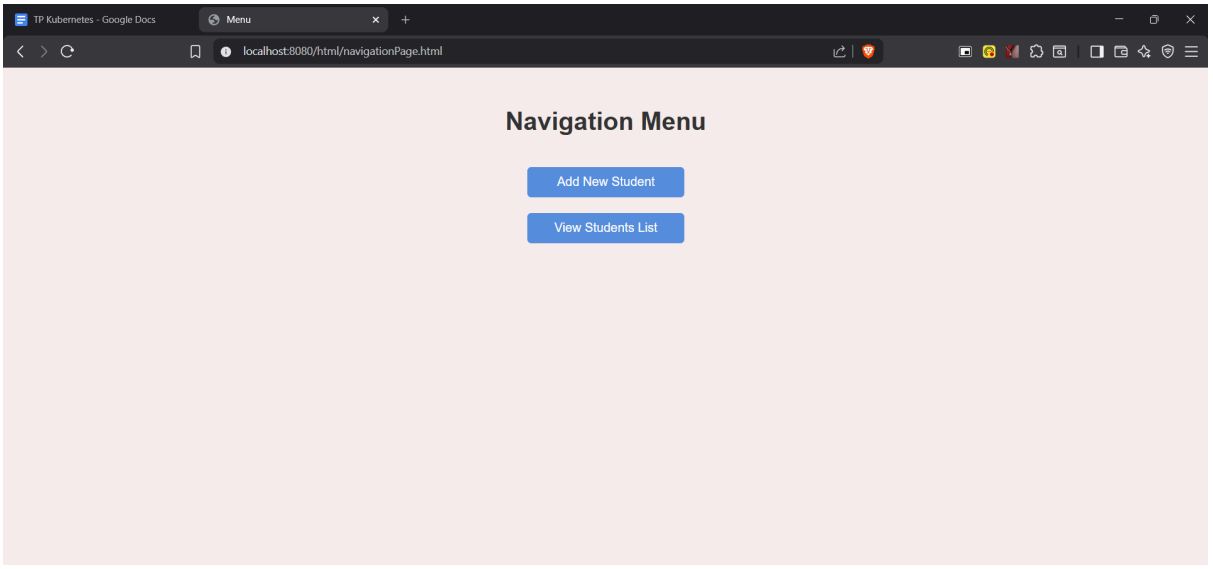
Pour commencer avec Docker, nous avons créé des scripts bash pour tout exécuter. Les commandes à effectuer sont décrites, dans l'ordre, dans le README du projet.

On peut également construire et lancer les containers avec "docker-compose up -d --build" :

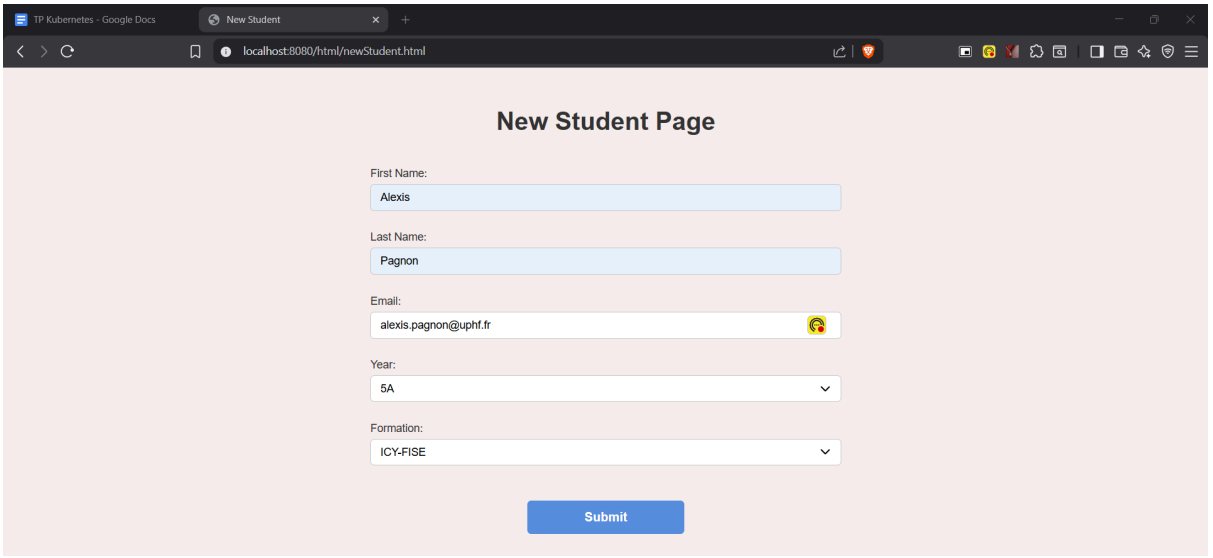


Nous avons les services front-consult et integration-services qui attendent que Kafka valide un healthcheck afin d'atteindre que Kafka soit opérationnel, car nos deux services ont besoin soit de créer soit de s'abonner à des topics, et pour faire ceci Kafka doit être opérationnel. Sans healthcheck, l'usage de "depends_on" ne permet d'assurer que le lancement du container, et non pas le fait qu'il est prêt à opérer.

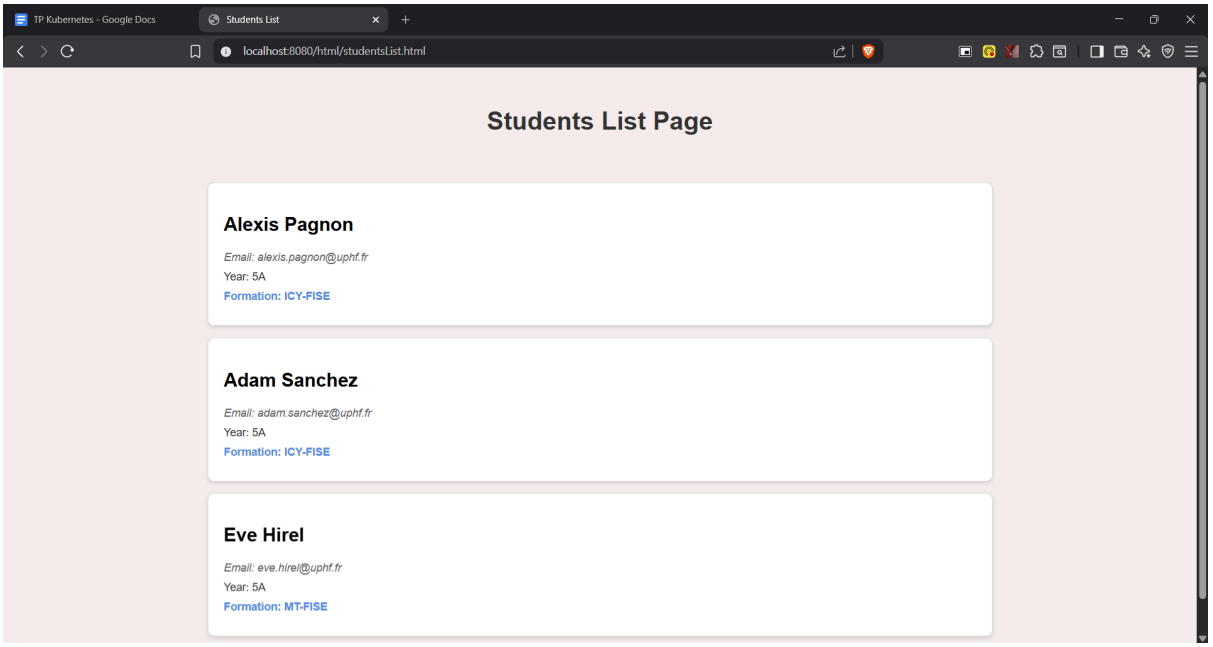
On peut ensuite accéder au service avec l'adresse "localhost:8080" :



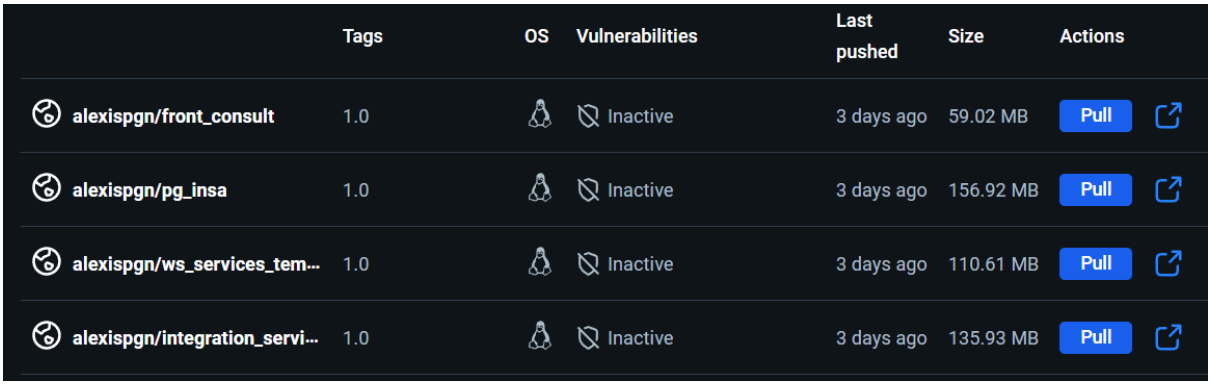
On peut ajouter des étudiants :



Et récupérer les étudiants :



Nous avons également push les containers sur DockerHub :



Kubernetes

Nous utilisons notre fichier “deploy.sh” pour déployer l’architecture avec Kubernetes :

- Dans un premier temps cela lance minikube

```
linuxmint@linuxmint-VirtualBox:~/Documents/Projet_architecture_EDA$ bash ./kubernetes/deploy.sh
Starting minikube...
😄  minikube v1.37.0 sur Linuxmint 22.2 (vbox/amd64)
✨  Utilisation du pilote docker basé sur le profil existant

❗  L'allocation de mémoire demandée de 3072MiB ne laisse pas de place pour la surcharge système (mémoire système totale : 3915MiB). Vous pouvez rencontrer des probl
èmes de stabilité.
💡  Suggestion : Start minikube with less memory allocated: 'minikube start --memory=3072mb'

👍  Démarrage du nœud "minikube" primary control-plane dans le cluster "minikube"
🚜  Extraction de l'image de base v0.0.48...
🔄  Redémarrage du docker container existant pour "minikube" ...
🐳  Préparation de Kubernetes v1.34.0 sur Docker 28.4.0...
🔎  Vérification des composants Kubernetes...
    ▪ Utilisation de l'image gcr.io/k8s-minikube/storage-provisioner:v5
    ▪ Utilisation de l'image docker.io/kubernetesui/metrics-scraper:v1.0.8
    ▪ Utilisation de l'image docker.io/kubernetesui/dashboard:v2.7.0
💡  Certaines fonctionnalités du tableau de bord nécessitent le module complémentaire metrics-server. Pour activer toutes les fonctionnalités, veuillez exécuter :

        minikube addons enable metrics-server

🌟  Modules activés: storage-provisioner, dashboard, default-storageclass
🏄  Terminé ! kubectl est maintenant configuré pour utiliser "minikube" cluster et espace de noms "default" par défaut.
```

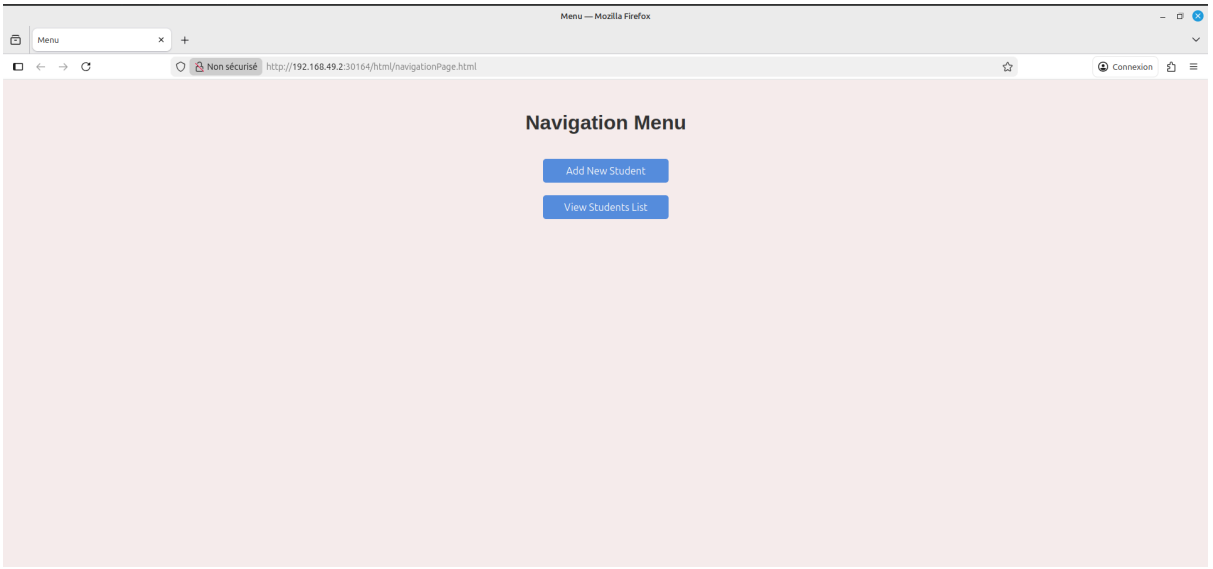
- Ensuite on crée un namespace puis on récupère nos images afin de tout pouvoir lancer :

```
Creating namespace...
namespace/architecture-eda created
Deploying pg-insa and kafka...
service/pg-insa created
statefulset.apps/pg-insa created
service/kafka created
statefulset.apps/kafka created
Waiting for pg-insa and kafka to be ready...
pod/pg-insa-0 condition met
pod/kafka-0 condition met
Deploying integration and ws services...
service/ws-template created
deployment.apps/ws-template created
deployment.apps/integration-services created
Deploying frontend service...
service/front-consult created
deployment.apps/front-consult created
Accessing the frontend service...
pod/front-consult-7d48c78b59-vnbgz condition met
```

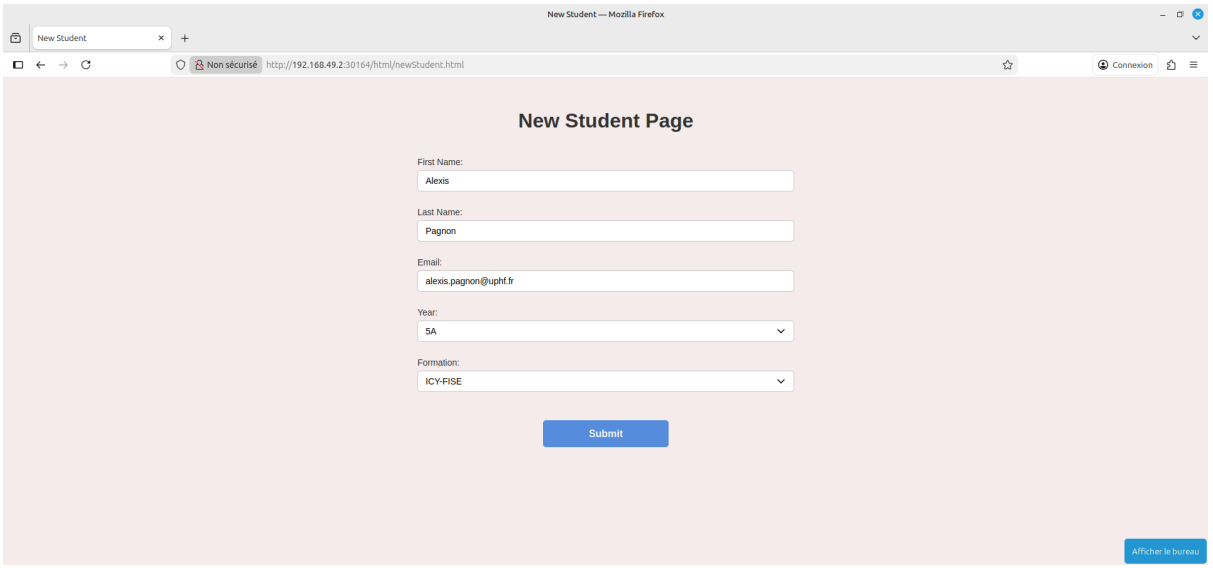
NAMESPACE	NAME	TARGET PORT	URL
architecture-eda	front-consult	8080	http://192.168.49.2:30164

```
🎉  Ouverture du service architecture-eda/front-consult dans le navigateur par défaut...
```

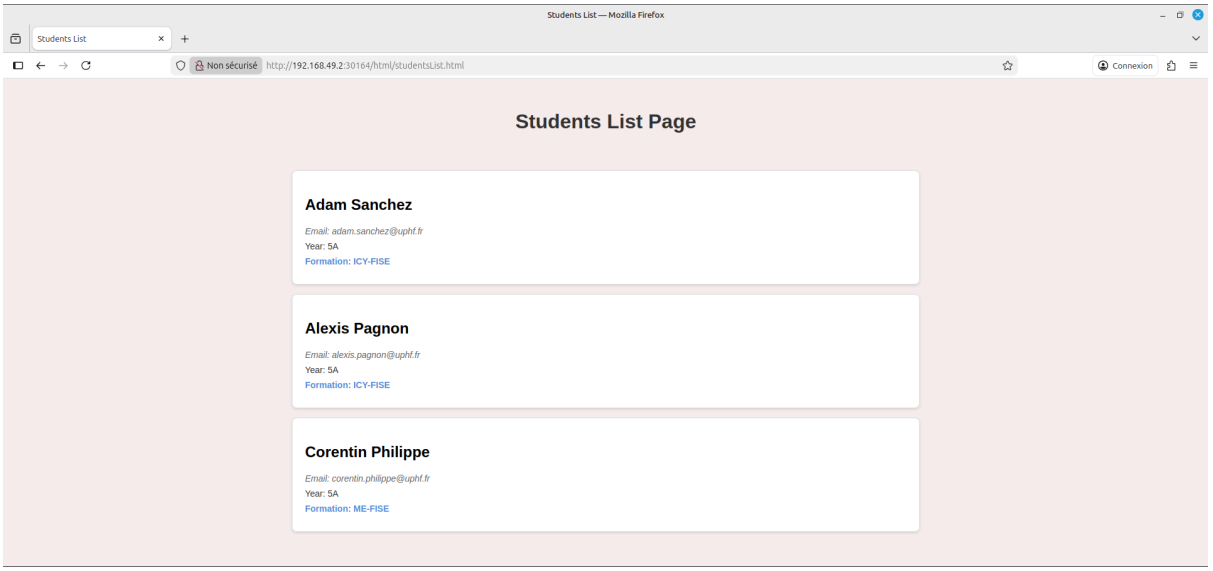
- Pour finir, cela lance notre navigateur par défaut pour accéder à notre site:



Page pour ajouter des étudiants:



Listes des étudiants :



Pour fermer le projet, il suffit simplement de lancer le script “undeploy.sh” :

```
linuxmint@linuxmint-VirtualBox:~/Documents/Projet_architecture_EDA$ bash ./kubernetes/undeploy.sh
Undeploying all resources...
service "front-consult" deleted from architecture-eda namespace
deployment.apps "front-consult" deleted from architecture-eda namespace
deployment.apps "integration-services" deleted from architecture-eda namespace
service "ws-template" deleted from architecture-eda namespace
deployment.apps "ws-template" deleted from architecture-eda namespace
service "kafka" deleted from architecture-eda namespace
statefulset.apps "kafka" deleted from architecture-eda namespace
service "pg-insa" deleted from architecture-eda namespace
statefulset.apps "pg-insa" deleted from architecture-eda namespace
Undeploying namespace...
namespace "architecture-eda" deleted
Undeploying completed !
Stopping minikube...
✋  Nœud d'arrêt "minikube" ...
🛑  Mise hors tension du profil "minikube" via SSH…
🛑  1 nœud arrêté.
```