Projet Virtualisation 4A

Réalisé par :

LARSONNEUR Sasha

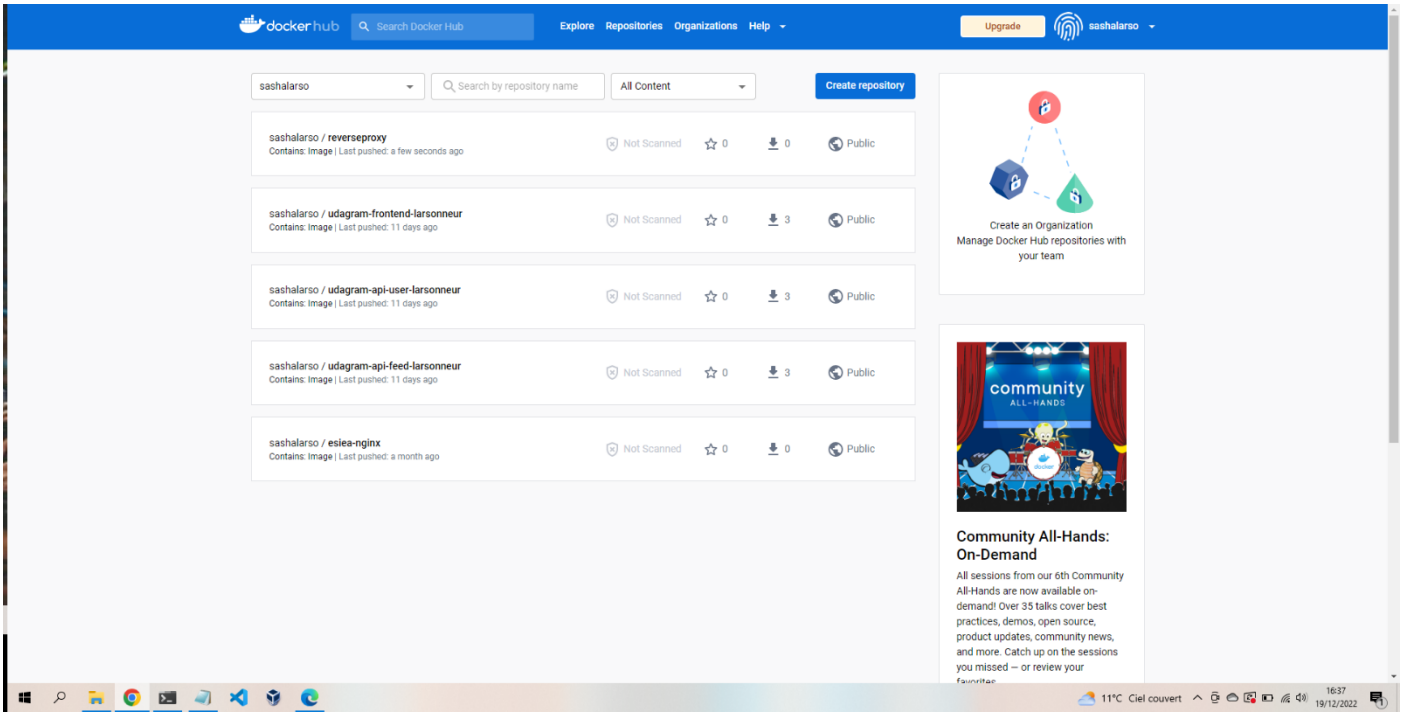
HOUNAKEY-AKAKPO Antony

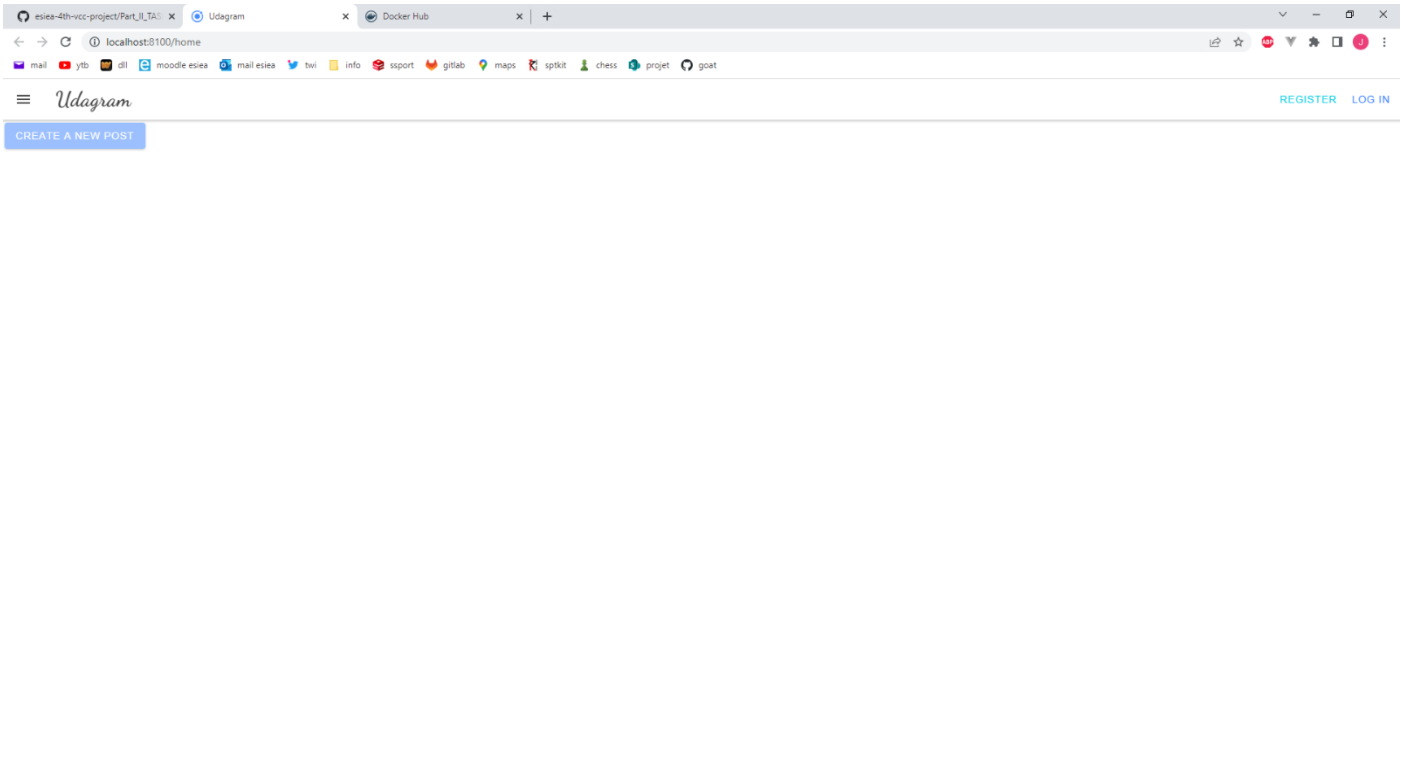
Dirigé par :

M. Mewenemesse Jean-Pascal

Lien du repo : https://github.com/sashalarso/ESIEA-4A-Virtualisation-Project.git

Partie 1 : Conteneurisation





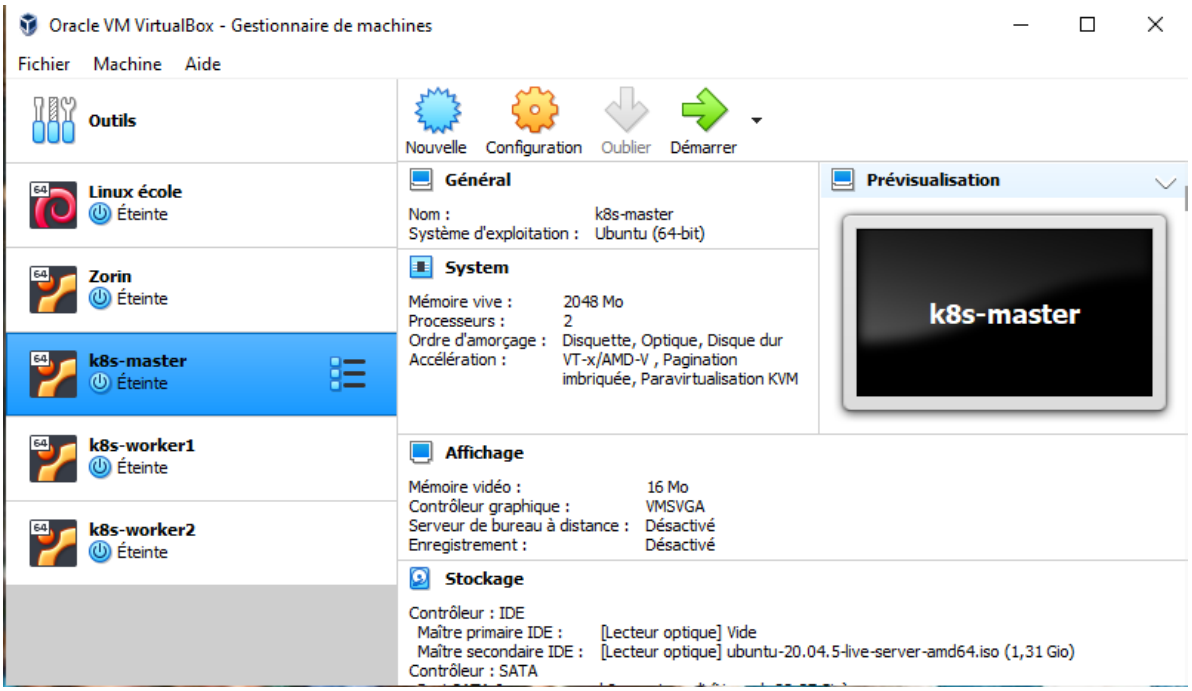
On voit bien que les containers tournent et donc l'application avec.

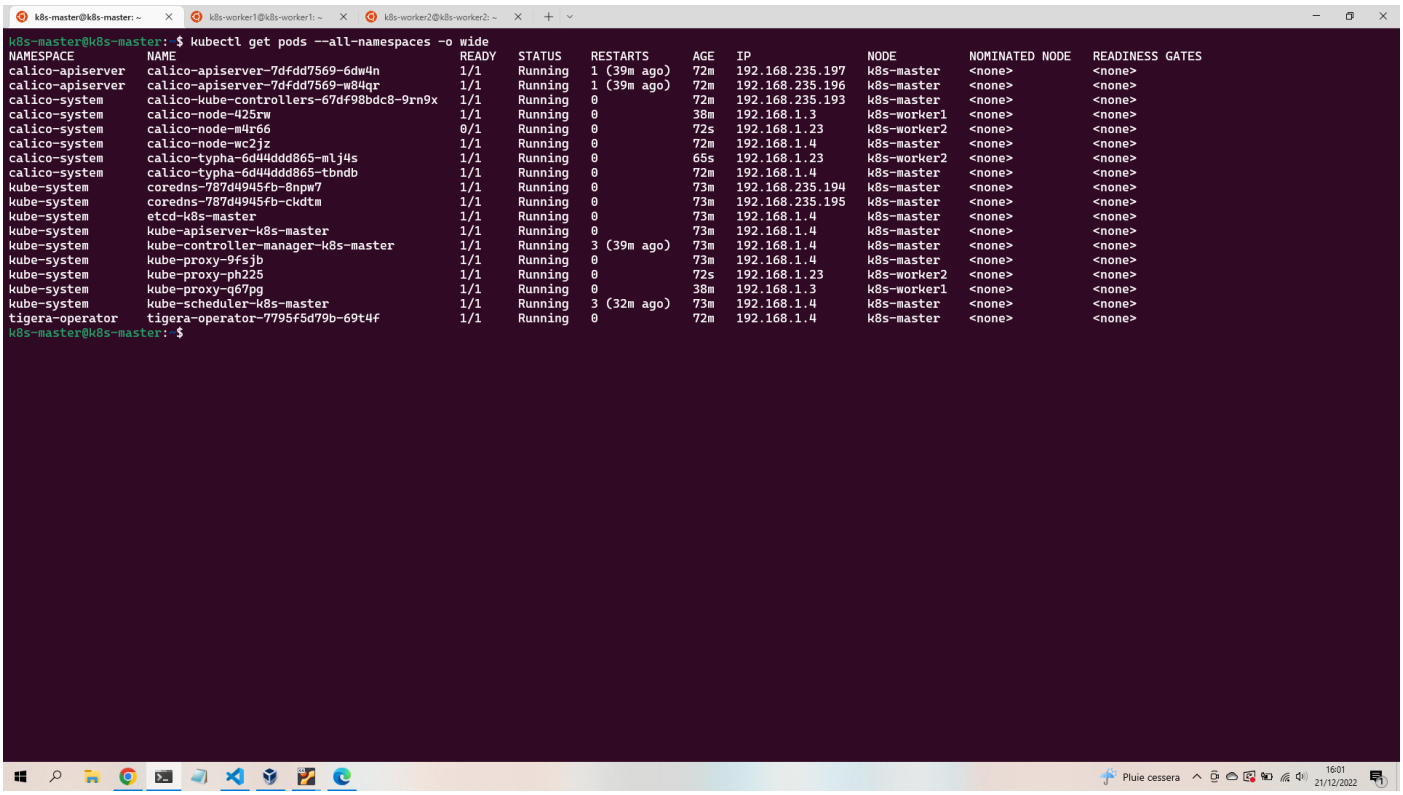
```
PS C:\Users\Sasha\Documents\Ecole\4A\Virtualisation\esiea-4th-vcc-project> docker-compose up -d
time="2022-12-19T15:30:36+01:00" level=warning msg="The \"HOME\" variable is not set. Defaulting to a blank string."
[+] Running 6/6
 - Network udagram                                  Created                                                    0.8s
 - Container esiea-4th-vcc-project-udagramd-db-1    Started                                                    2.9s
 - Container esiea-4th-vcc-project-frontend-1       Started                                                    3.0s
 - Container esiea-4th-vcc-project-backend-feed-1   Started                                                    4.6s
 - Container esiea-4th-vcc-project-backend-user-1   Started                                                    4.6s
 - Container esiea-4th-vcc-project-reverseproxy-1   Started                                                    5.9s
PS C:\Users\Sasha\Documents\Ecole\4A\Virtualisation\esiea-4th-vcc-project>
```

```
PS C:\Users\Sasha\Documents\Ecole\4A\Virtualisation\esiea-4th-vcc-project> docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED         STATUS         PORTS                              NAMES
46ac640f4c24   reverseproxy           "/docker-entrypoint.…"   2 minutes ago   Up 2 minutes   80/tcp, 0.0.0.0:8080->8080/tcp     esiea-4th-vcc-project-reverseproxy-1
8ed05684356e   udagram-api-user       "docker-entrypoint.s…"   2 minutes ago   Up 2 minutes   8080/tcp                           esiea-4th-vcc-project-backend-user-1
bcff7caf4a65   udagram-api-feed       "docker-entrypoint.s…"   2 minutes ago   Up 2 minutes   8080/tcp                           esiea-4th-vcc-project-backend-feed-1
e4c1b00c789d   udagram-frontend:local "/docker-entrypoint.…"   2 minutes ago   Up 2 minutes   0.0.0.0:8100->80/tcp               esiea-4th-vcc-project-frontend-1
daa7e162ee5c   postgres               "docker-entrypoint.s…"   2 minutes ago   Up 2 minutes   0.0.0.0:5432->5432/tcp             esiea-4th-vcc-project-udagramd-db-1
```

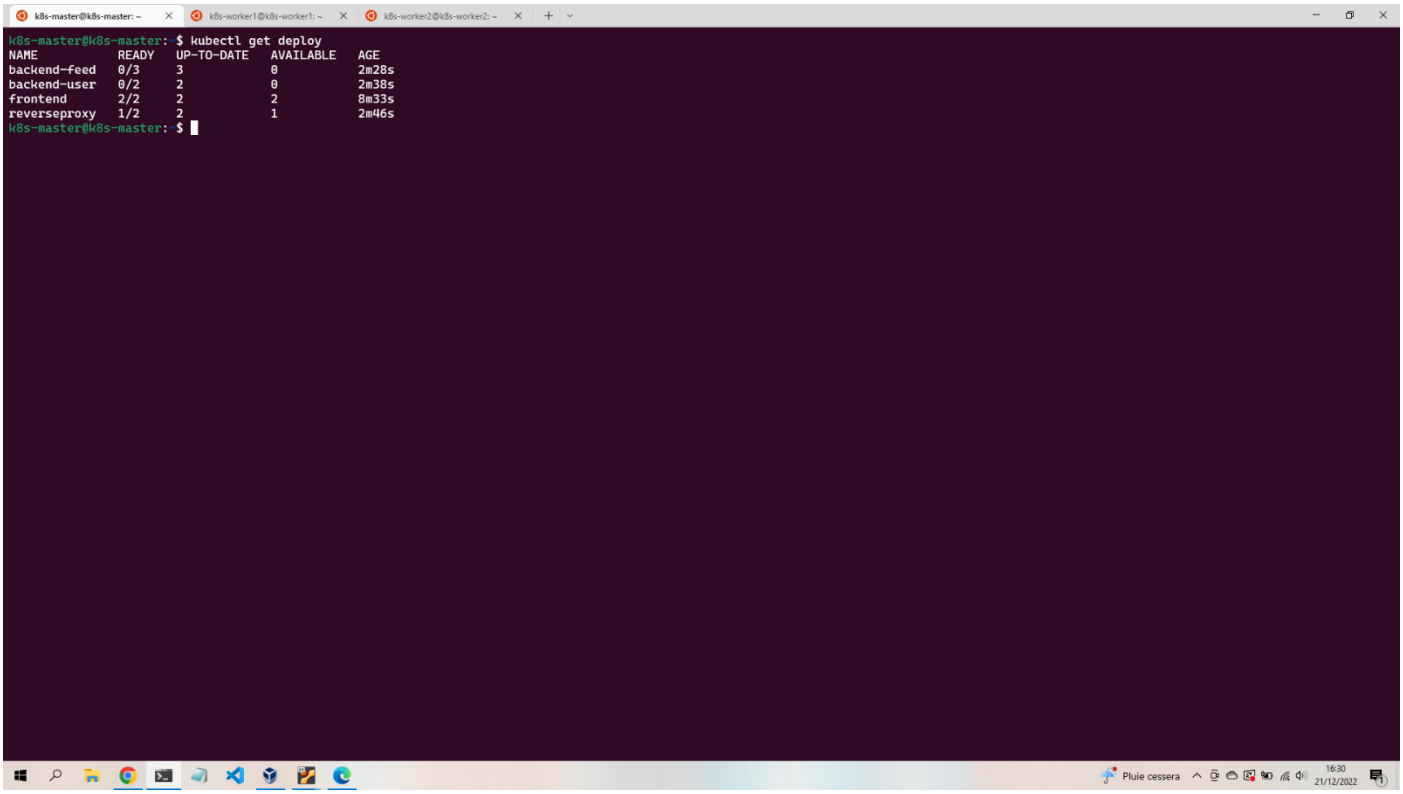
Partie 2 : Orchestration

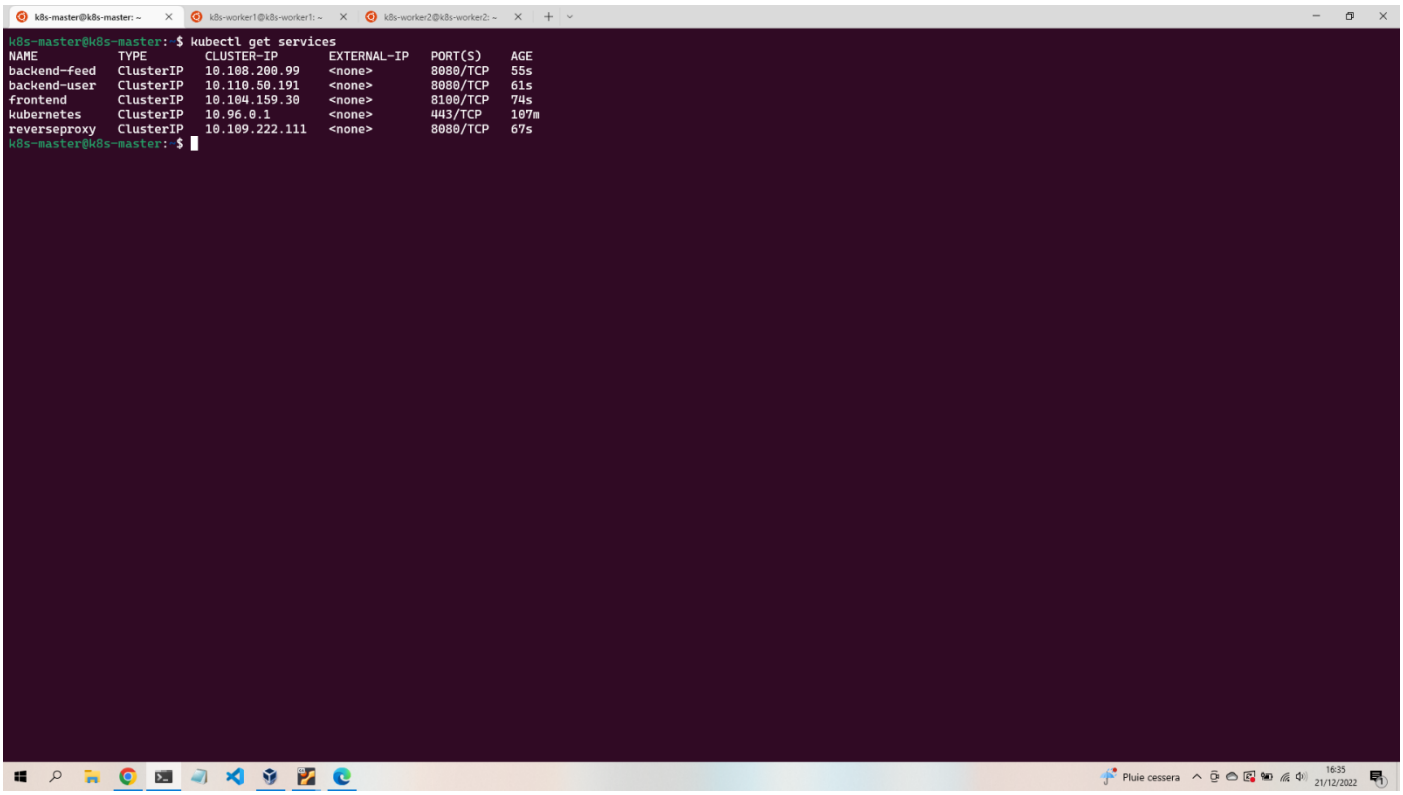
```
k8s-master@k8s-master:~$ kubectl get nodes
NAME          STATUS   ROLES           AGE    VERSION
k8s-master    Ready    control-plane   41h    v1.25.4
k8s-worker1   Ready    <none>          38h    v1.25.4
k8s-worker2   Ready    <none>          110s   v1.25.4
k8s-master@k8s-master:~$
```

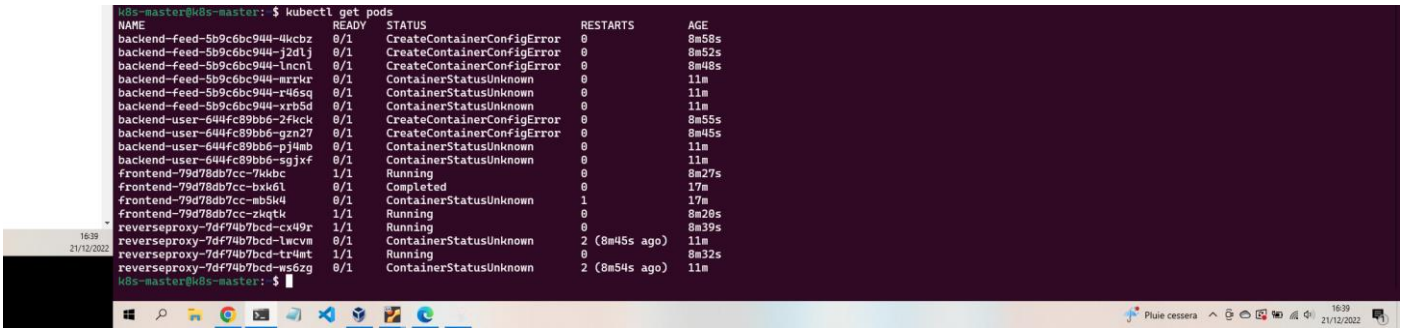




```
k8s-master@k8s-master: $ kubectl get pods --all-namespaces -o wide
NAMESPACE          NAME                                       READY   STATUS    RESTARTS      AGE   IP               NODE          NOMINATED NODE   READINESS GATES
calico-apiserver   calico-apiserver-7dfdd7569-6dw4n           1/1     Running   1 (39m ago)   72m   192.168.235.197  k8s-master    <none>           <none>
calico-apiserver   calico-apiserver-7dfdd7569-w84qr           1/1     Running   1 (39m ago)   72m   192.168.235.196  k8s-master    <none>           <none>
calico-system      calico-kube-controllers-67df98bdc8-9rn9x   1/1     Running   0             72m   192.168.235.193  k8s-master    <none>           <none>
calico-system      calico-node-425rw                          1/1     Running   0             38m   192.168.1.3      k8s-worker1   <none>           <none>
calico-system      calico-node-m4r66                          0/1     Running   0             72s   192.168.1.23     k8s-worker2   <none>           <none>
calico-system      calico-node-wc2jz                          1/1     Running   0             72m   192.168.1.4      k8s-master    <none>           <none>
calico-system      calico-typha-6d44ddd865-mlj4s              1/1     Running   0             65s   192.168.1.23     k8s-worker2   <none>           <none>
calico-system      calico-typha-6d44ddd865-tbndb              1/1     Running   0             72m   192.168.1.4      k8s-master    <none>           <none>
kube-system        coredns-787d4945fb-8npw7                   1/1     Running   0             73m   192.168.235.194  k8s-master    <none>           <none>
kube-system        coredns-787d4945fb-ckdtm                   1/1     Running   0             73m   192.168.235.195  k8s-master    <none>           <none>
kube-system        etcd-k8s-master                            1/1     Running   0             73m   192.168.1.4      k8s-master    <none>           <none>
kube-system        kube-apiserver-k8s-master                  1/1     Running   0             73m   192.168.1.4      k8s-master    <none>           <none>
kube-system        kube-controller-manager-k8s-master         1/1     Running   3 (39m ago)   73m   192.168.1.4      k8s-master    <none>           <none>
kube-system        kube-proxy-9fsjb                           1/1     Running   0             73m   192.168.1.4      k8s-master    <none>           <none>
kube-system        kube-proxy-ph225                           1/1     Running   0             72s   192.168.1.23     k8s-worker2   <none>           <none>
kube-system        kube-proxy-q67pg                           1/1     Running   0             38m   192.168.1.3      k8s-worker1   <none>           <none>
kube-system        kube-scheduler-k8s-master                  1/1     Running   3 (32m ago)   73m   192.168.1.4      k8s-master    <none>           <none>
tigera-operator    tigera-operator-7795f5d79b-69t4f           1/1     Running   0             72m   192.168.1.4      k8s-master    <none>           <none>
k8s-master@k8s-master: $
```


k8s-master@k8s-master: ~
k8s-worker1@k8s-worker1: ~
k8s-worker2@k8s-worker2: ~
k8s-master@k8s-master:~$ kubectl get deploy
NAME READY UP-TO-DATE AVAILABLE AGE
backend-feed 0/3 3 0 2m28s
backend-user 0/2 2 0 2m38s
frontend 2/2 2 2 8m33s
reverseproxy 1/2 2 1 2m46s
k8s-master@k8s-master:~$
Pluie cessera
16:30
21/12/2022


k8s-master@k8s-master: ~
k8s-worker1@k8s-worker1: ~
k8s-worker2@k8s-worker2: ~
k8s-master@k8s-master:~$ kubectl get services
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
backend-feed ClusterIP 10.108.200.99 <none> 8080/TCP 55s
backend-user ClusterIP 10.110.50.191 <none> 8080/TCP 61s
frontend ClusterIP 10.104.159.30 <none> 8100/TCP 74s
kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 107m
reverseproxy ClusterIP 10.109.222.111 <none> 8080/TCP 67s
k8s-master@k8s-master:~$
Pluie cessera
16:35
21/12/2022

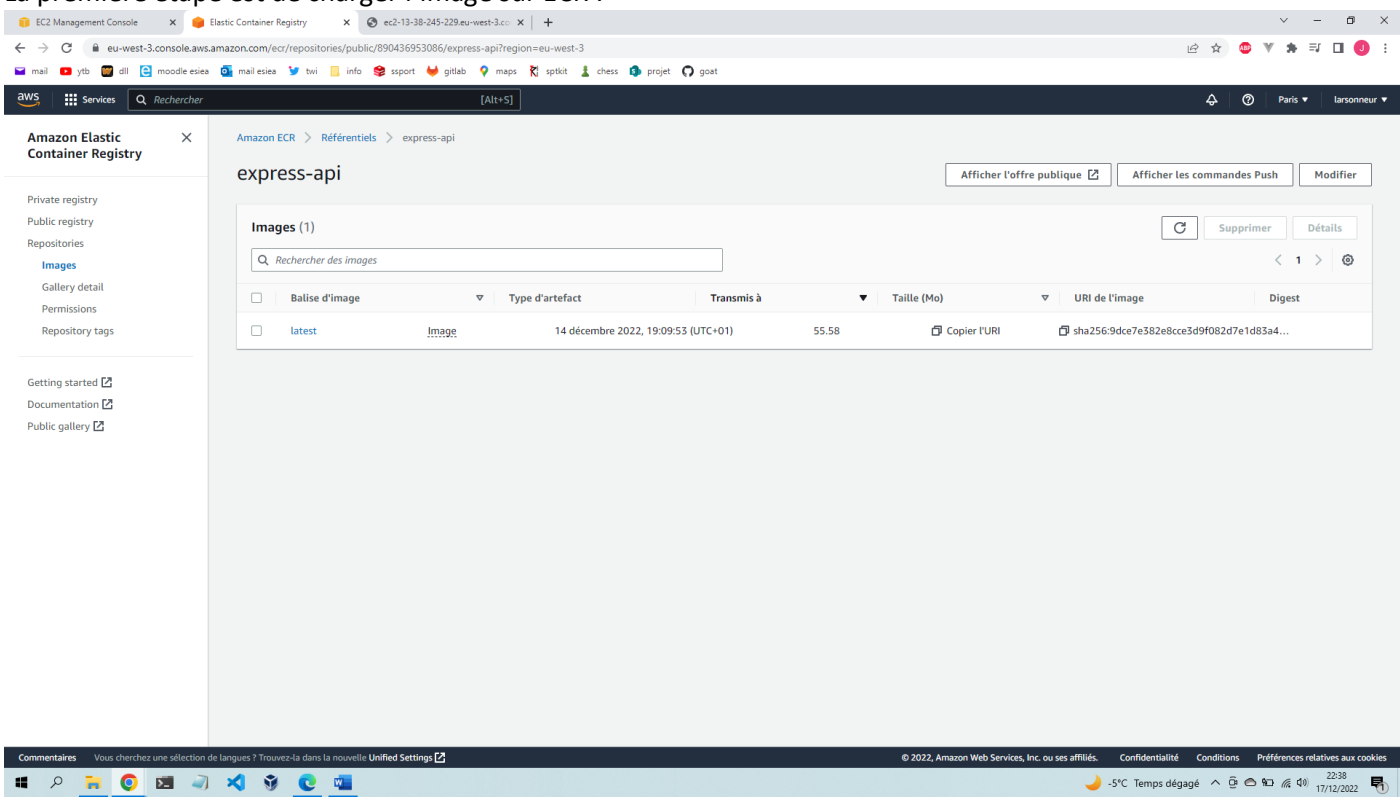

k8s-master@k8s-master:~$ kubectl get pods
NAME READY STATUS RESTARTS AGE
backend-feed-5b9c6bc944-4kcbz 0/1 CreateContainerConfigError 0 8m58s
backend-feed-5b9c6bc944-j2dlj 0/1 CreateContainerConfigError 0 8m52s
backend-feed-5b9c6bc944-lncnl 0/1 CreateContainerConfigError 0 8m48s
backend-feed-5b9c6bc944-mrrkr 0/1 ContainerStatusUnknown 0 11m
backend-feed-5b9c6bc944-r46sq 0/1 ContainerStatusUnknown 0 11m
backend-feed-5b9c6bc944-xrb5d 0/1 ContainerStatusUnknown 0 11m
backend-user-644fc89bb6-2fkck 0/1 CreateContainerConfigError 0 8m55s
backend-user-644fc89bb6-gzn27 0/1 CreateContainerConfigError 0 8m45s
backend-user-644fc89bb6-pj4mb 0/1 ContainerStatusUnknown 0 11m
backend-user-644fc89bb6-sgjxf 0/1 ContainerStatusUnknown 0 11m
frontend-79d78db7cc-7kkbc 1/1 Running 0 8m27s
frontend-79d78db7cc-bxk6l 0/1 Completed 0 17m
frontend-79d78db7cc-mb5k4 0/1 ContainerStatusUnknown 1 17m
frontend-79d78db7cc-zkqtk 1/1 Running 0 8m20s
reverseproxy-7df74b7bcd-cx49r 1/1 Running 0 8m39s
reverseproxy-7df74b7bcd-lwcvm 0/1 ContainerStatusUnknown 2 (8m45s ago) 11m
reverseproxy-7df74b7bcd-tr4mt 1/1 Running 0 8m32s
reverseproxy-7df74b7bcd-ws6zg 0/1 ContainerStatusUnknown 2 (8m54s ago) 11m
k8s-master@k8s-master:~$
16:39
21/12/2022
Pluie cessera
16:39
21/12/2022

Partie 3 : Cloud

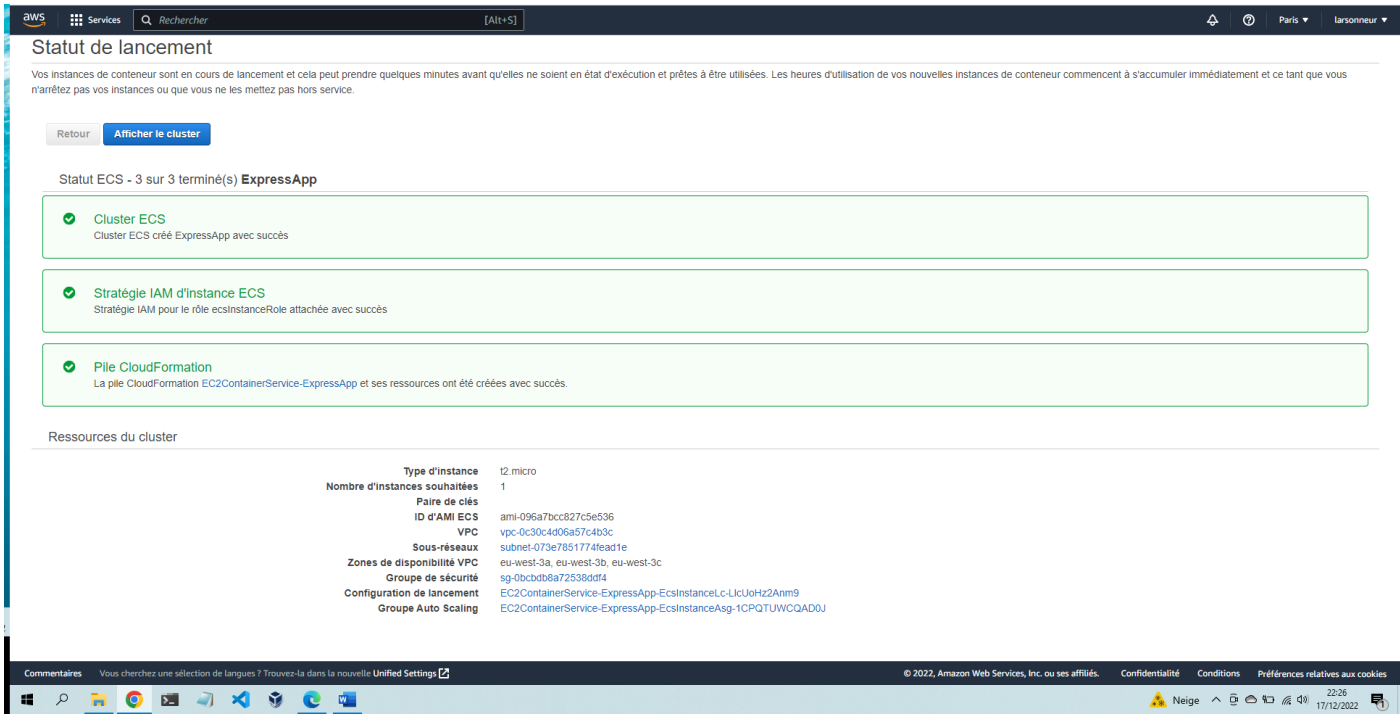
1. Le cloud est une infrastructure dans laquelle la puissance de calcul et le stockage sont gérés par des serveurs distants auxquels les usagers se connectent via une liaison Internet sécurisée. L'ordinateur de bureau ou portable, le téléphone mobile, la tablette tactile et autres objets connectés deviennent des points d'accès pour exécuter des applications ou consulter des données qui sont hébergées sur les serveurs. Le cloud se caractérise également par sa souplesse qui permet aux fournisseurs d'adapter automatiquement la capacité de stockage et la puissance de calcul aux besoins des utilisateurs.
   Un des avantages de l'utilisation du Cloud est le fait de payer seulement pour ce que l'on utilise : par exemple, en cas de pic de trafic sur un site : quand on utilise pas le cloud il faudrait des machines prévues pour ces pics et donc payer pour des appareils utilisés seulement lors de ces pics.
   Enfin le cloud nous permet un gain de temps considérable sur la configuration des machines, cela est paramétrable en quelques clics contrairement aux machines physiques qu'il faudrait configurer, etc.

2. Le fait de dire « il n'y a pas de cloud, c'est juste l'ordinateur de quelqu'un d'autre » est le fait que les machines que l'on utilise dans le Cloud ne sont pas des machines existant par magie dans un nuage, c'est simplement une machine de la société proposant le Cloud. Par exemple, Amazon possède des datacenters ou sont stockées des milliers de machines louables et utilisables à distance par des utilisateurs.
3. On peut citer 5 services d'AWS :
   -EC2 : permettant de louer des machines à distance
   -S3 : permettant de stocker des données
   -Amazon RDS : base de données relationnelle (plus simple à mettre en place qu'une SQL classique)
   - ECR : service permettant de stocker des images Docker et de les exploiter aisément
   - Amazon DynamoDB : base de données NoSQL utilisant le système de clés valeurs (avantage : latence extrêmement faible)

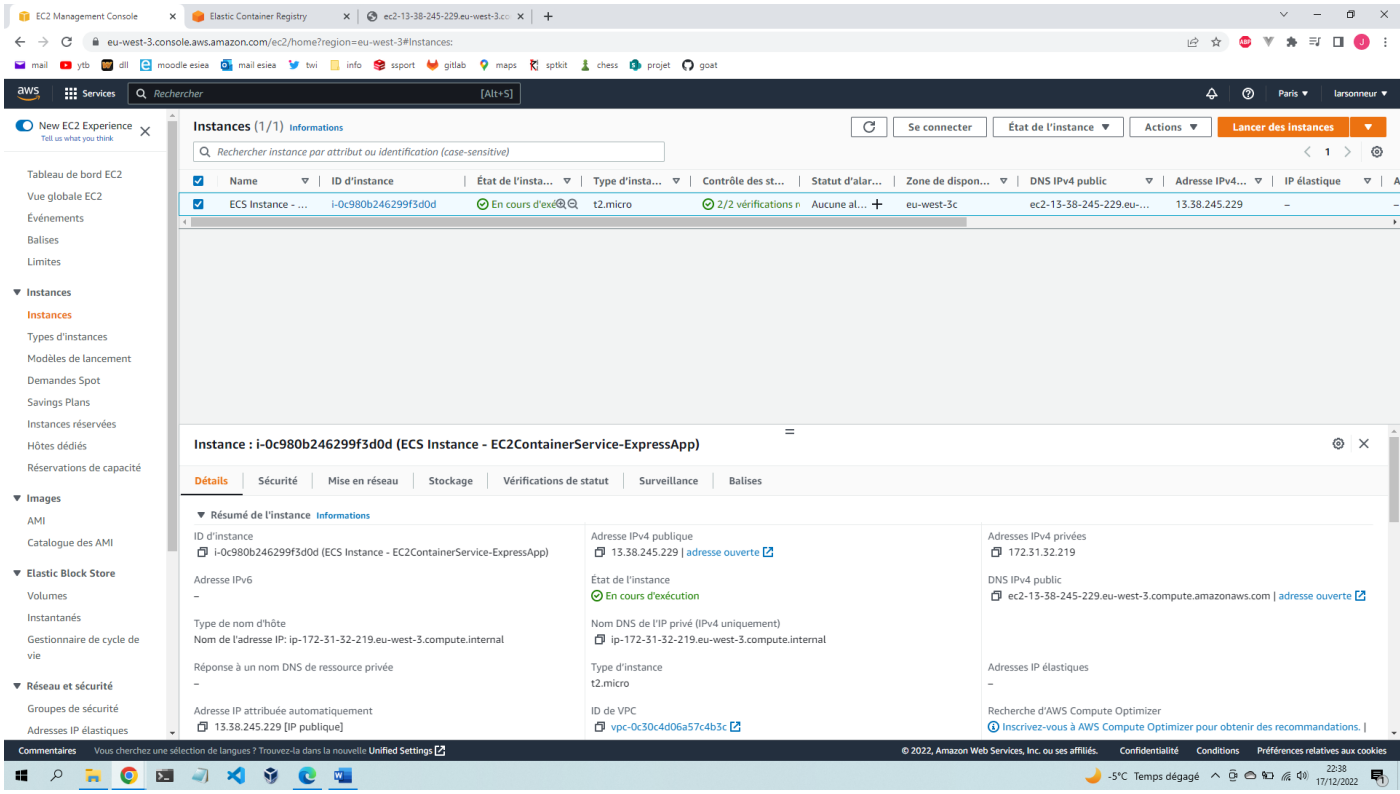
   J'ai choisi comme exemple d'utilisation d'AWS le déploiement d'une API sur le web en utilisant EC2, ECS, ECR.
   J'ai utilisé ce repo Github : https://github.com/harblaith7/Docker-AWS-Crash-Course.
   Ce repo contient déjà un Dockerfile qui installe les dépendances, run l'application et expose les bon ports.
   La première étape est de charger l'image sur ECR :



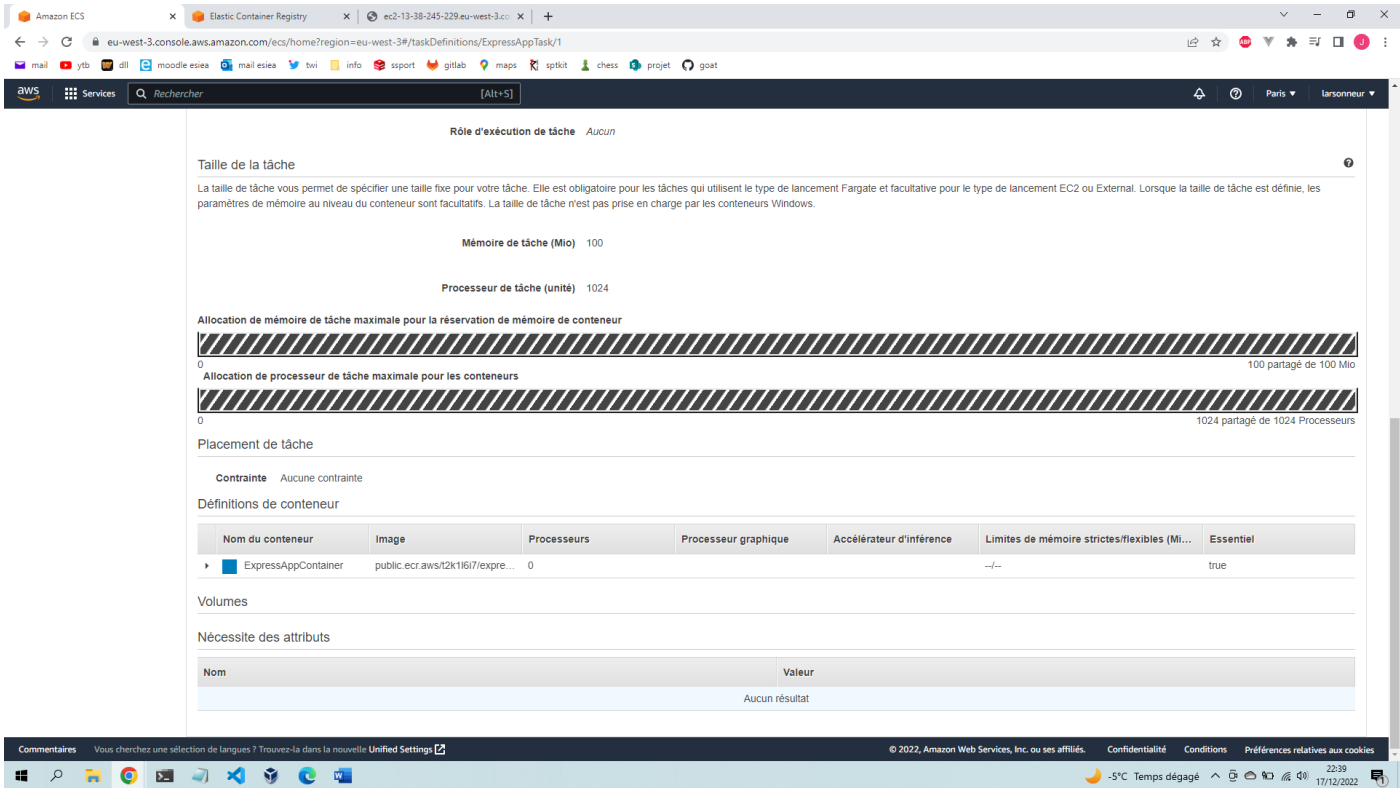
Cela a été réalisé à l'aide de commandes que ce service nous donne pour pusher l'image dans ECR (« afficher les commandes Push »).
Ensuite nous avons utilisé le service cluster de ECS afin de lancer nos instances (ici, on a seulement besoin d'une instance).



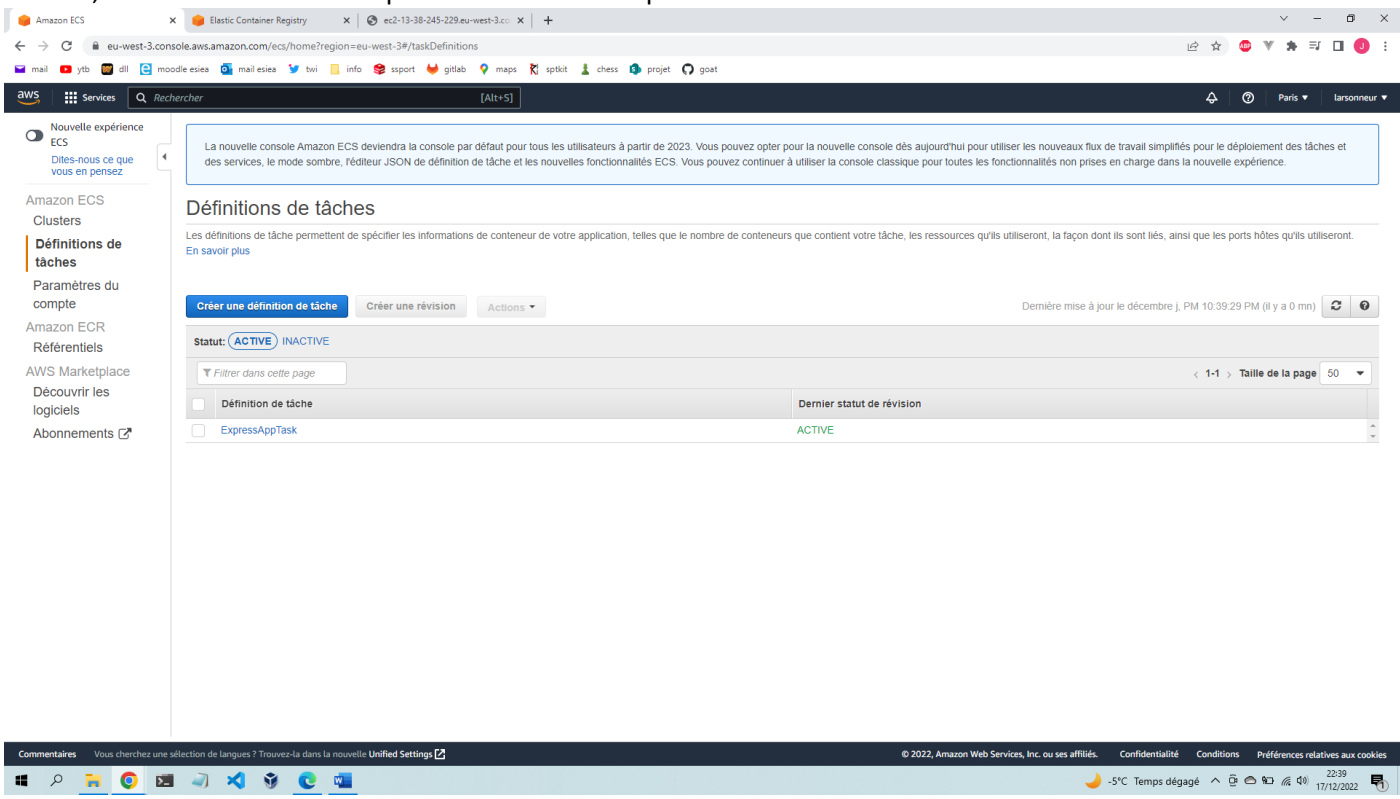
On peut voir que cela a bien lancé une instance dans EC2 :



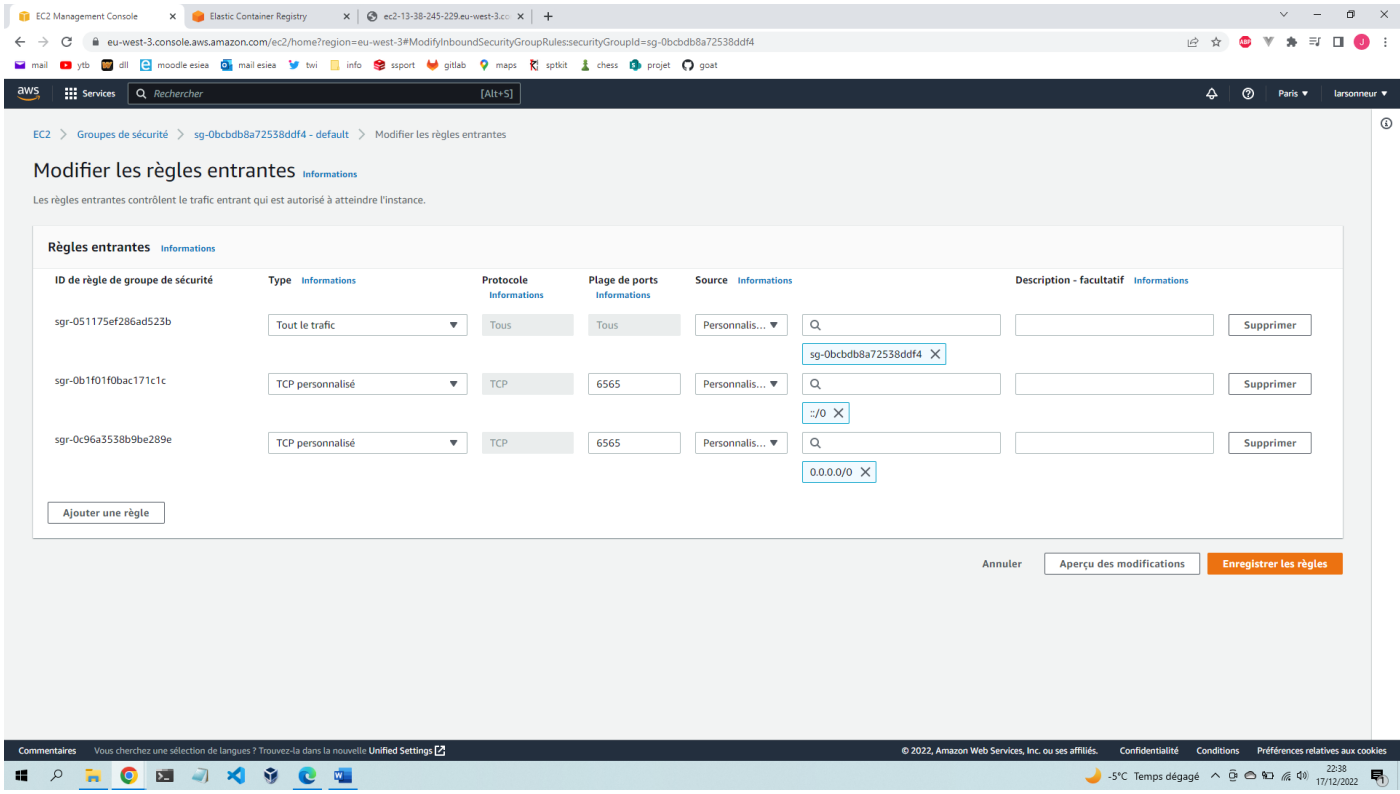
Une fois cela fait, dans le service cluster on a lancé une tâche : c'est un service permettant de mieux gérer le déploiement des containers dans des instances, l'exposition des bon ports, etc.
On peut voir ci-dessous que l'on a choisi comme image celle que l'on avait déposé dans ECR :



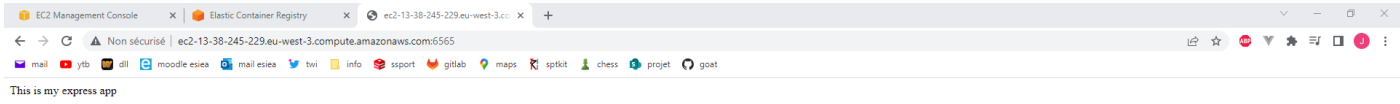
Dans la définition de la tâche on a aussi spécifié notre port d'écoute pour la machine ainsi que le port de l'api, cela se résume à effectuer un mapping de ports.
Ensuite, on a lancé la tâche : on peut voir ci -dessous qu'elle est bien lancée :



La dernière chose à faire est modifier les règles de sécurité de l'instance, c'est-à-dire autoriser le trafic entrant sur le port de notre machine qu l'on a choisi pour mapper avec le port de l'api. (ici 6565/5000).
On peut voir ci-dessous que l'on a autorisé toutes les adresses du réseau sur le port 6565.

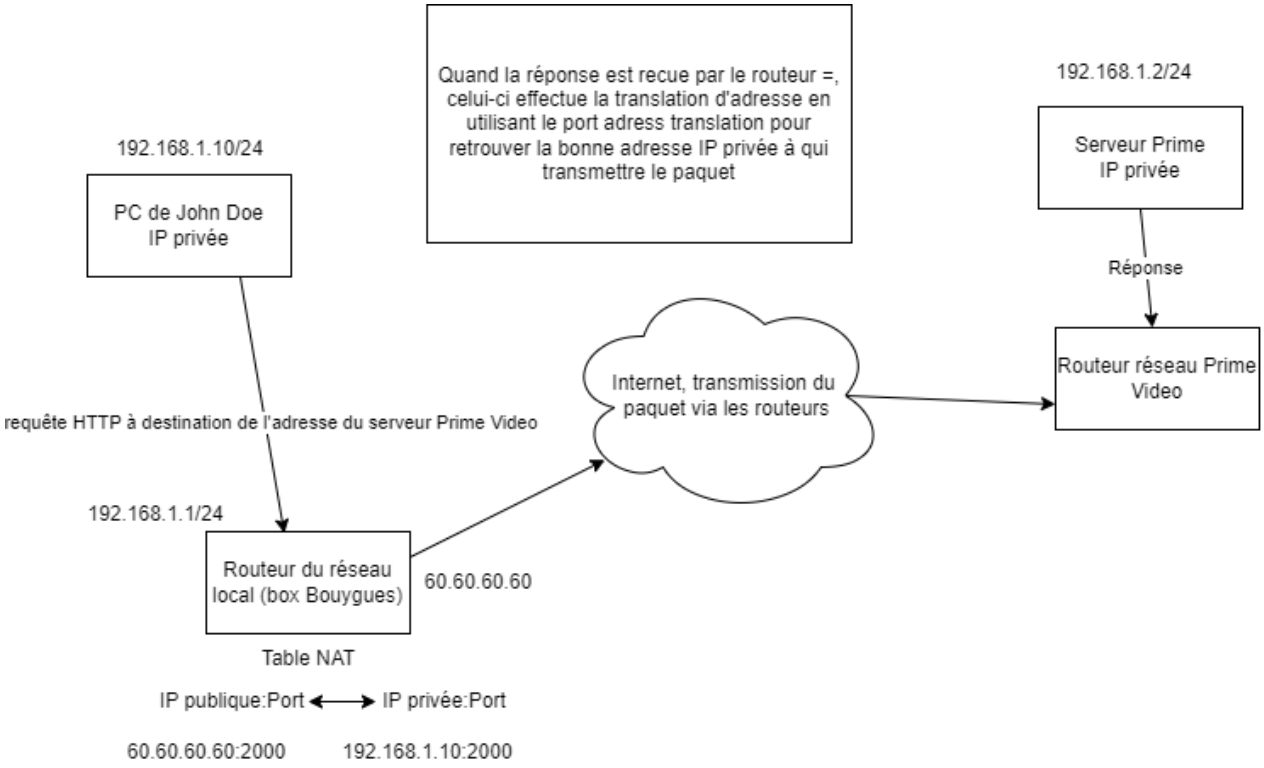


Enfin, nous n'avons plu qu'à récupérer l'adresse Ipv4 de notre instance (voir au dessus), y associer le bon numéro de port et le bon point de terminaison afin de vérifier si l'api est bien déployée sur notre instance :



On peut voir ci-dessus que c'est bien le cas. On a évidemment pris soin à la fin de supprimer le cluster et donc l'instance afin de ne pas laisser tourner dans le vide.

4. Le NAT est le fait d'utiliser un système d'adresses IP publiques et privées pour un réseau informatique. Ce processus a été inventé afin de remédier à la limite naturelle d'adresses IP disponibles (255^4 = 2milliards environ). En effet, il y'a plus de machines connectées à internet aujourd'hui que d'adresses Ipv4 disponibles ainsi en utilisant le NAT chaque réseau se voit attribuer une adresse IP publique commune à toutes les machines de ce réseau et c'est le routeur qui fait la translation avec les machines du réseau. Ces machines possèdent chacune une adresse IP privée, non visible par les autres machines en dehors du réseau, cela permet a chaque réseau de gérer ses attributions d'adresses comme il l'entend sans être contraint par la limitation d'adresses disponibles.
5. Les adresses IP publiques sont visibles par tout le monde à l'éxterieur du réseau, elles sont utilisées pour l'envoi de paquet au réseau possédant cette adresse. Une fois le paquet arrivé au routeur connecté à internet, celui-ci effectue la translation IP publique vers privée en utilisation une table NAT qu'il contient. Ainsi l'adresse IP privée est l'adresse IP unique au sein du réseau de la machine. Cette adresse n'est cependant pas unique au monde, d'autres machines peuvent avoir la même adresse mais cela n'est de toute facon pas visible à l'exterieur du réseau puisque ce sont les adresses publiques qui sont utilisées pour communiquer entre les réseaux.
6.


Quand la réponse est recue par le routeur =, celui-ci effectue la translation d'adresse en utilisant le port adress translation pour retrouver la bonne adresse IP privée à qui transmettre le paquet
192.168.1.2/24
Serveur Prime
IP privée
192.168.1.10/24
PC de John Doe
IP privée
Réponse
Internet, transmission du paquet via les routeurs
Routeur réseau Prime Video
requête HTTP à destination de l'adresse du serveur Prime Video
192.168.1.1/24
Routeur du réseau local (box Bouygues)
60.60.60.60
Table NAT
IP publique:Port
IP privée:Port
60.60.60.60:2000
192.168.1.10:2000