République Algérienne Démocratique et Populaire
Ministre de L'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITÉ ABDERRAHMANE MIRA BEJAIA
Département d'Informatique
Faculté des sciences Exactes
Spécialité Génie Logiciel

Réaliser par

ADRAR NABILLA

Thème

Déploiement cloud des algorithmes de machine learning

Devant un jury composé de :
Présidente: *Dr.ALOUI Soraya*
Examinateur: *Dr. MOUKATFI Mouhand*

Supervisé par:
Encadreur: *Dr. BELAID Ahror*
Co-encadreur : *Mr.AKILAL Abdellah*

Remerciements

Je tiens à exprimer ma gratitude à M. Abdellah Akilal et au Dr BELAID Ahror pour leurs soutiens enthousiastes lors de la réalisation de ce projet ainsi que pour leurs conseils et recommandations qui m'ont aidé à mener à bien cette recherche.

Je remercie tous les membres sincères de ma famille, mes parents, mes sœurs et mon frère, qui m'ont soutenu et encouragé tout au long de ma vie, ainsi que pour leur aide précieuse, leur patience et leur soutien indéfectible.

Je tiens à remercier chacun des membres du jury pour l'intérêt qu'ils ont porté à notre travail en acceptant de l'examiner et de l'enrichir de leurs suggestions. Merci à tous les professeurs et membres du Département d'Informatique de l'Université ABDERRAHMANE MIRA de Béjaïa.

Enfin, je remercie mes amis Imene, Fatima, Lyes, Ferhat, Nesrine, Hichem et Fouad d'être toujours à mes côtés. Leur soutien inconditionnel et leurs encouragements ont été d'une grande aide.

Dédicaces

Je dédie ce travail à :
Mes parents
Mes chers Soeurs: Hakima, Sabrina, Fouzia, Karima et Samia
Mon petit frère Sofiane
Ma grand-mère que son âme repose en paix
Tous les membres de ma famille
Mes amis : Imene, Fatima, Nesrine, Lyes et Ferhat
Mes tantes, mes cousines et Tata Samira
Hichem et Fouad avec qui j'ai partagé le fruit de ce travail.
Toutes les personnes qui m'ont offert leur aide.

Contents

List of Figures

Liste des abréviations

Dev	développeurs
Ops	opérationnels
KL	Machine learning
GPU	Graphics Processing Unit
CPU	Central Processing Unit
K8s	Kubernetes
CLI	Command Line Interface
VDI	Virtual desktop infrastructure
API	Application Programming Interface
VM	Virtual Machine
CD	Continuous delivery
CI	Continuous integration
IP	Internet Protocol
OS	Operating system
Iaas	Infrastructure as a service
Paas	Platform as a service
Saas	Software as a service

Introduction Générale

L'intelligence artificielle (IA) est un processus qui imite l'intelligence humaine, basé sur la création et l'application d'algorithmes qui s'exécutent dans un environnement informatique dynamique, son but est de permettre aux ordinateurs de penser et d'agir comme des êtres humains. Cela dit, comme on pouvait s'y attendre, la pensée humaine est non seulement complexe mais aussi assez versatile ce qui rend la tâche de sa traduire en algorithmes extrêmement fastidieuse voir quasiment impossible.

C'est ici que le Machine Learning (ML) entre en jeu, ayant pour but et objectif de donner aux ordinateurs la possibilité et la capacité d'apprendre par eux-mêmes simplement en se basant sur des données numériques massives (Big Data) qu'ils traitent. Cela permet la création d'applications d'apprentissages automatiques pouvant analyser et assimiler un volume énorme de donnés en un temps record prennant avantage du fait que la vitesse de traitement d'un humain restera toujours inférieure à celle d'un ordinateur moderne et ce fossé ne cesse de se creuser aux avancées dans le domaine de l'électronique.

Le Machine Learning a introduit un nouveau chapitre dans le domaine de l'automatisation, en d'autres termes il a permi l'automatisation et l'optimisation de plusieures taches ce qui a entraîné le développement fulgurant de certain domaines tels que l'automobile (voitures intelligentes), l'industrie lourde (machines automatiques), la santé ... etc.

Cependant les scientifiques des données créent et développent une variété de modèles différents, dont 87% n'arrivent pas au stade de déploiement[1]. La mise en production et la maintenance des modèles d'apprentissage automatique sont l'un des défis les plus pressants auxquels les organisations sont confrontées aujourd'hui. Le flux de travail ML, qui comprend la formation, la construction et le déploiement de modèles d'apprentissage automatique, peut être un long processus avec de nombreux obstacles en cours de route[2].

Les conteneurs contiennent tous les éléments nécessaires au fonctionnement du code d'apprentissage automatique, ce qui garantit un environnement cohérent. La solution passe donc par un paradigme d'orchestration de conteneurs tels que Kubernetes qui contribue à l'automatisation de la gestion des conteneurs, comme la surveillance, la programmation et la mise à l'échelle.

Cet ouvrage couvre de prime abord les trois premiers chapitres qui décrivent tout ce qui touche au développement, au déploiement, au machine learning, à la conteneurisation et aux outils à utiliser.Le 4ème est dédié au déploiement local avec minikube et kubeadm, enfin il se termine par le déploiement cloud avec AWS et Azure.

Chapter 1

Développement et déploiement

1.1 Introduction

Nous présentons dans ce premier chapitre les concepts liés Aux phases du déploiement. Nous allons ainsi aborder des concepts tels que : Virtualisation, conteneurs, Kubernetes, etc. L'objectif principal est de décrire l'impact de ces techniques dans la phase de déploiement d'une application.

1.2 Développement logiciel

Pour réaliser une application, un logiciel ou un système informatique il faut impérativement passer par les phases de cycle de vie du développement logiciel. Afin de garantir une production réussie d'une solution de haute qualité qui répondra ou dépassera les exigences d'un client à toutes ses phases, dans le respect du budget et des délais prévus.

phases du développement logiciel :

- **Analyse des besoins :** Cette étape consiste à collecter tous les besoins du client (fonctionnels ou non fonctionnels), établir l'étude de l'existant (étude des produits similaires dans le marché), et analyse de faisabilité tout en déterminant les délais et les coups.
- **Planification :** Se résume à établir un concept ou une idée sur laquelle le logiciel sera développé. (planifier le calendrier de travail, les outils, les méthodes à utiliser et les risques auxquels ils pourraient être confrontés.)
- **Conception :**il s'agit de concevoir le système qui doit répondre aux exigences et de définir son architecture et les différents composants nécessaire.[3]
- **Développement :** Comprends l'étape où les programmeurs transforment des solutions proposées lors de la conception en un code opérationnel.
- **Tests :** Il s'agit de la phase de test et de validation qui déterminent les bugs techniques ou fonctionnels et la qualité du logiciel.

- **Déploiement et maintenance :** Le déploiement consiste à transférer le logiciel depuis l'environnement de développement vers l'environnement de production. Les utilisateurs pourront ainsi commencer à l'utiliser. La maintenance

qui s'effectue après la livraison du produit au client comprends la correction des erreurs et des anomalies du système, modification du système en ajoutant ou supprimant des fonctionnalités.

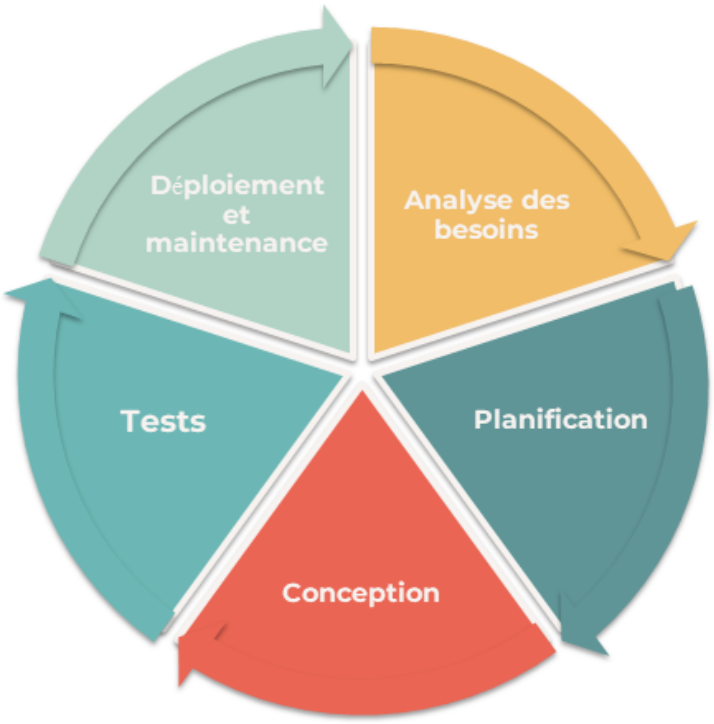


Figure 1.1: : Cycle de vie.

1.3 Concept du DevOps:

Commençant par comprendre comment les entreprises fonctionnaient traditionnellement sans DevOps.
Les développeur ou les devs recueillent les exigences du métier avant de développer le code, ensuite cette même équipe teste le nouveau code avant de le livrer à l'équipe qui se chargera de la production et d'exploitation Qu'on appelle Ops pour opérations. Cependant ce modèle pose des problèmes car les deux équipes ont des objectifs différents :

Les "dev" ont pour but d'obtenir un "time to market réduit", le temps entre la conception du produit et la mise en production. Un time to market rapide permet à une entreprise de s'adapter et de répondre plus rapidement à la demande du marché mais pour aussi proposé de l'innovation aux clients avant les concurrents. De l'autre côté l'Objectif de l'équipe Ops est de faire garantir la stabilité du système cela passe donc par un contrôle très sévère, Une panne de production pouvant couter très très cher à l'entreprise. Cet antagonisme provoque des conflits d'intérêts, les devs blâment les retards et les problèmes de livraison des Ops et les Ops blâment les incidents de l'équipe de production liés à la mauvaise qualité du code.
Afin d'améliorer le processus précédent, la méthodologie DevOps a émergé pour créer un environnement dans lesquels la conception, les tests et la publication de logiciels peuvent

se produire rapidement, fréquemment et efficacement en alignant les objectifs à court terme autour d'un objectif commun sur le long terme.

- Le cycle de vie DevOps :

Le cycle de vie DevOps comprend six phases [4]

1. **Développement continue :** C'est la phase dans laquelle le logiciel est "planifié" et "codé". Il n'y a pas besoin d'outils DevOps pour la planification, mais il existe de nombreux outils de maintenance du code.Il est aussi possible écrire du code dans n'importe quel langage, mais il est géré par un outil de contrôle de version. La maintenance du code est connue sous le nom de contrôle du code source. Les outils les plus couramment utilisés sont Git, SVN, Mercurial, CVS et JIRA.

2. **Intégration continue (CI) :** Cette phase est au cœur de l'ensemble du cycle de vie DevOps. Il s'agit d'une pratique de développement logiciel dans laquelle les développeurs doivent apporter des modifications au code source plus fréquemment. Cela peut être fait quotidiennement ou hebdomadairement. Cela permet d'identifier rapidement les problèmes potentiels. L'écriture de code comprend non seulement la compilation, mais également la révision du code, les tests unitaires, les tests d'intégration et l'empaquetage. Le code prenant en charge les nouvelles fonctionnalités continuera d'être intégré au code existant. Comme le développement logiciel est continu, le code mis à jour doit être intégré en continu et en douceur dans le système pour refléter les changements de l'utilisateur final.

3. **Tests continus :** C'est l'étape au cours de laquelle le logiciel développé est constamment testé pour détecter les erreurs. des outils de tests automatisés sont utilisés qui permettent de tester plusieurs bases de code de manière approfondie en parallèle pour s'assurer qu'il n'y a pas de défauts dans les fonctionnalités. À ce stade, il est possible d'utiliser le conteneur Docker pour simuler lenvironnement de test. L'automatisation des tests fait gagner beaucoup de temps, d'efforts et de travail pour exécuter les tests au lieu de les exécuter manuellement. Le rapport est également un gros plus. Il est également possible de planifier l'exécution du scénario de test à une heure prédéfinie. Après les tests, le code continuera à être intégré dans le code existant.

4. **Déploiement continu (CD):**C'est à ce moment que le code est déployé sur le serveur de production. Il faut s'assurer que le code est correctement déployé sur tous les serveurs.Dans cette phase il y a un ensemble d'outils permet un déploiement continu.

    * Outils de gestion de la configuration est le processus d'établissement et de maintien de la cohérence entre les exigences fonctionnelles et les performances de l'application. C'est-à-dire le déploiement, la planification des mises à jour pour tous les serveurs et plus important encore, le maintien de la cohérence de la configuration sur tous les serveurs.
    * Les outils de conteneurisation aident à maintenir la cohérence entre les environnements dans lesquels l'application est développée, testée et déployée. Grâce à ces outils, l'intégration et reproduction des mêmes dépendances et packages

utilisés dans le développement, les tests et la préparation, il n'y a donc aucun risque d'erreur ou d'échec en production.

5. **Surveillance et alerte:** Il s'agit d'une étape très importante du cycle de vie "DevOps" qui surveille en permanence les performances des applications. Des informations importantes sur l'utilisation du logiciel sont stockées ici. Ces informations sont traitées pour reconnaître le bon fonctionnement de l'application, les erreurs système telle que mémoire insuffisante, serveur inaccessible, etc. Dans cette phase, la cause de chaque problème est identifiée.

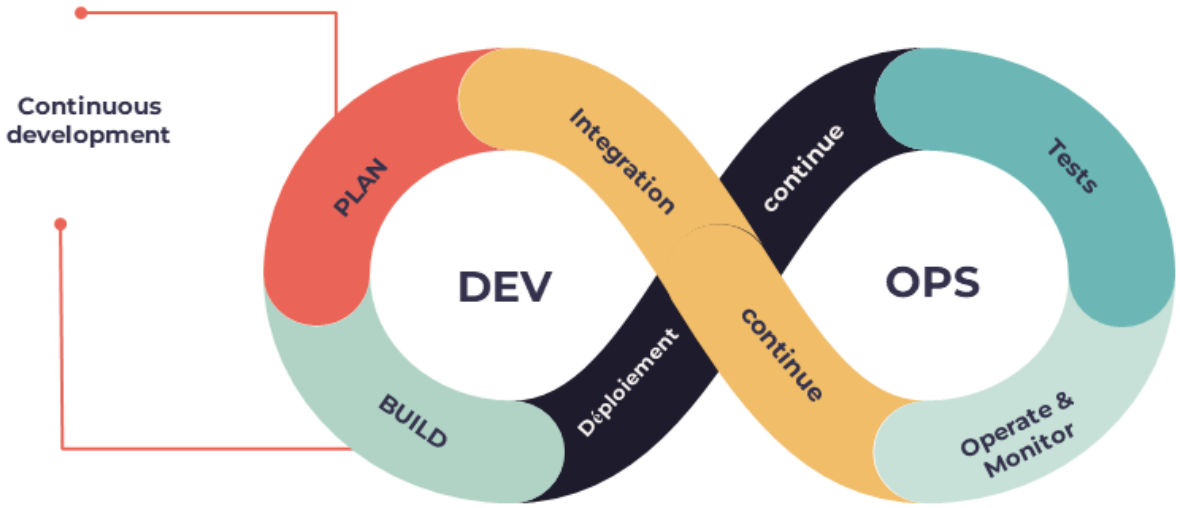


Figure 1.2: : Cycle de vie DevOps.
[5]

1.4 Déploiement d'applications

Le déploiement est le mécanisme par lequel les applications, les modules, les mises à jour et les correctifs sont livrés des développeurs aux utilisateurs.[6]
Expliquer de manière plus simple (voir figure 1.3):

Lorsqu'on crée une application, on y a accès que sur notre propre ordinateur ce qu'on appelle localhost (environnement de développement local), cette application est pas disponible au public. Si on veut la diffuser aux utilisateurs, On aura besoin de la rendre disponible sur des serveurs web qui hébergent notre code d'application, l'exécutent et le livrent a toutes personne qui le demande, Cela permet a d'autre ordinateurs sur internet d'y accéder. les différentes étapes nécessaires à la diffusion d'une application à un publique prédéfini est ce qu'on appelle le processus de déploiement.

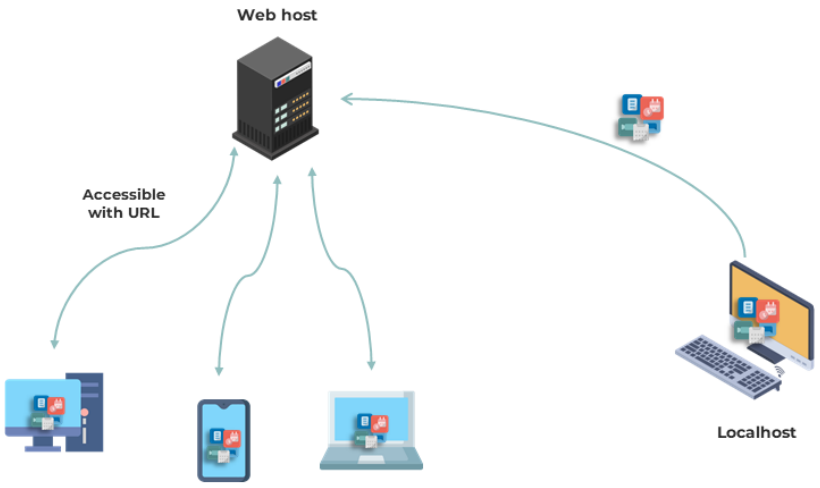


Figure 1.3: : Déploiement.

1.5 Type de déploiement

- **On premise:** Concerne le déploiement local ou l'entreprise est propriétaire et responsable de la prise en charge de l'infrastructure et des serveurs dédiés au système, Son installation comprend un investissement initial considérable, mais offre un contrôle complet sur la sécurité, la maintenance et les mises à jour.

- **On ligne:** Concerne le déploiement Cloud computing qui implique que le système soit hébergé par un fournisseur tiers, la solution est louer sur une base horaires,journalière, mensuel ou annuel.Le fournisseur est responsable de la sécurité,Les mises à jour et la planification des patchs.

  - Les modèles de déploiement du Cloud Computing comprennent:

    (a) **Cloud public:** Les ressources sont ouvertes au grand public, etsont partagées avec d'autres entreprises, ce qui diminuent les coûts et la sécurité

    (b) **Cloud priver:** Est plus chère car l'ensembles des ressources sont réservées à l'usage exclusive d'une seule entreprise, ne sont donc pas partagés avec d'autres firmes, assurant ainsi un plus grand contrôle sur la vitesse et la sécurité.

    (c) **Cloud hybride:** Corresponds a un mélange entre le cloud privé et public c'est-à-dire avoir des ressources dédiées en interne et des ressources publiques en externe

  Dans ce qui suit nous allons parler d'un concept associé au Devops et qui MLops.

1.6 MLOPS

MLOps ou Machine Learning Operations est l'équivalent ML de DevOps qui est un ensemble d'approches de gestion du cycle de vie ML pour rationaliser, automatiser et accélérer le déploiement de modèles ML afin d'aider les équipes Data Science (DS), les ingénieurs devops et les informaticiens.[7]

1.6.1 Pourquoi avons-nous besoin de MLOps ?

L'introduction de l'apprentissage automatique dans un environnement de production est difficile. Le cycle de vie de l'apprentissage automatique comprend de nombreux composants complexes tels que: l'ingestion de données, la préparation des données, la formation de modèles, l'optimisation de modèles, le déploiement de modèles, la surveillance de modèles et la responsabilité.
Nous avons également besoin de collaboration et de transfert entre les équipes, de l'ingénierie des données à la science des données en passant par l'ingénierie ML.[8]

-Les phases clés de MLOps sont les suivantes :

1. Création: Comprend la préparation des données, l'ingénierie des fonctionnalités, la création de modèles et les tests.

2. **G**estion: une fois les modèles créés, ils sont généralement placés dans un référentiel vérifiable soumis au contrôle des versions afin de favoriser leur réutilisation dans l'ensemble de l'entreprise.

3. **D**éploiement: exportation, déploiement et intégration du modèle ou du pipeline à des systèmes et applications de production.

4. : une surveillance continue est nécessaire pour garantir des performances optimales. À mesure que les données changent, le modèle peut être réentraîné ou remplacé par un nouveau modèle.

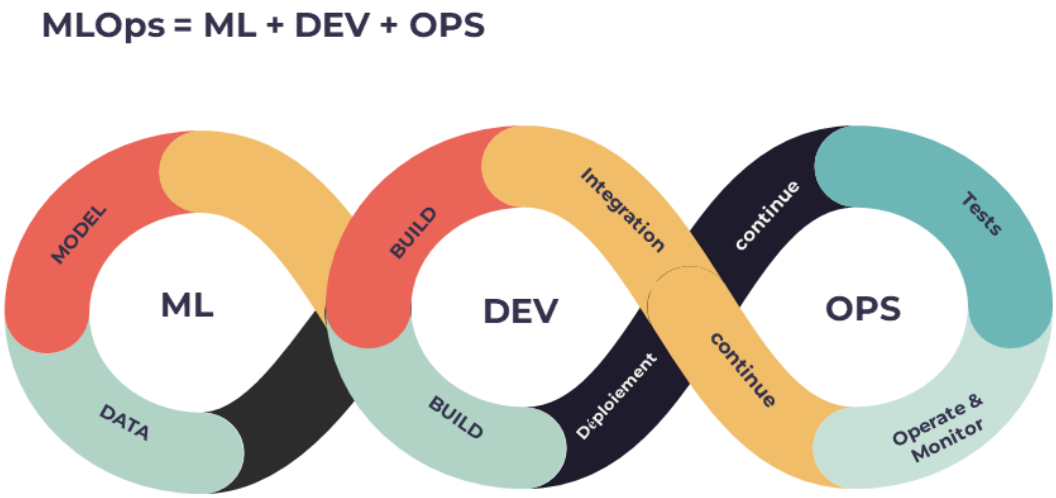


Figure 1.4: : Relation entre MLops et DevOps.

1.7 Stratégie de déploiement

Une stratégie de déploiement est un moyen de modifier ou de mettre à niveau une application. L'objectif est d'apporter des modifications sans temps d'arrêt de manière à ce que l'utilisateur remarque à peine les améliorations. Il est donc important de bien choisir la bonne stratégie.[9]

1. **Recreate:** Consiste à désactiver et arrêter la version A puis à déployer la version B (voir figure 1.5)

2. **Ramped(rolling):** La version B est lentement déployée et remplace la version A, Les déploiements par phases impliquent la mise à jour d'un sous-ensemble de versions d'application (plutôt que la mise à jour d'un déploiement de base complet),Un ReplicaSet secondaire est créé avec la nouvelle version de l'application, puis le nombre de répliques de l'ancienne version est réduit et la nouvelle version est augmentée jusqu'à ce que le nombre correct de répliques soit atteint (voir figure 1.6).

3. **Bleu/Vert :** Est une stratégie pour déployer deux versions d'une application en même temps. Un environnement (bleu) exécute la version actuelle de l'application et un environnement (vert) exécute la nouvelle version de l'application. La version bleue continue de fonctionner tandis que la version verte subit des tests (voir figure 1.7).

4. **Canary:** La version B est publiée pour un sous-ensemble d'utilisateurs, puis procède à un déploiement complet permettant de réduire le risque d'introduction d'une mise à jour logicielle en production (voir figure 1.8).

5. **Tests A/B:** La version B est diffusée à un sous-ensemble d'utilisateurs dans des conditions spécifiques. En bref, testez une nouvelle fonctionnalité avec un ensemble ciblé (et non aléatoire) d'utilisateurs, mais en fonction de segments de paramètres utilisateur courants tels que la géographie ou le type d'appareil (voir figure 1.9).

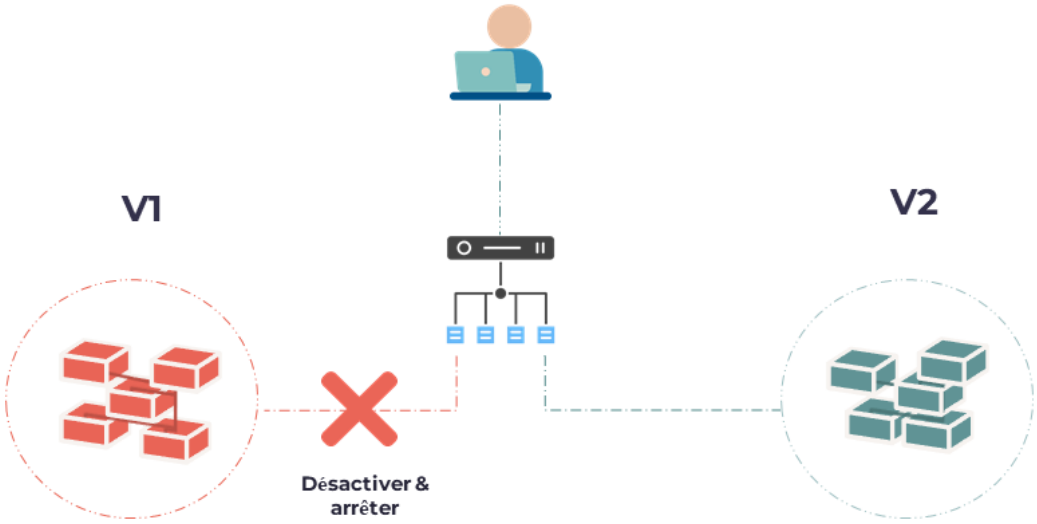


Figure 1.5: Recreate
[9]

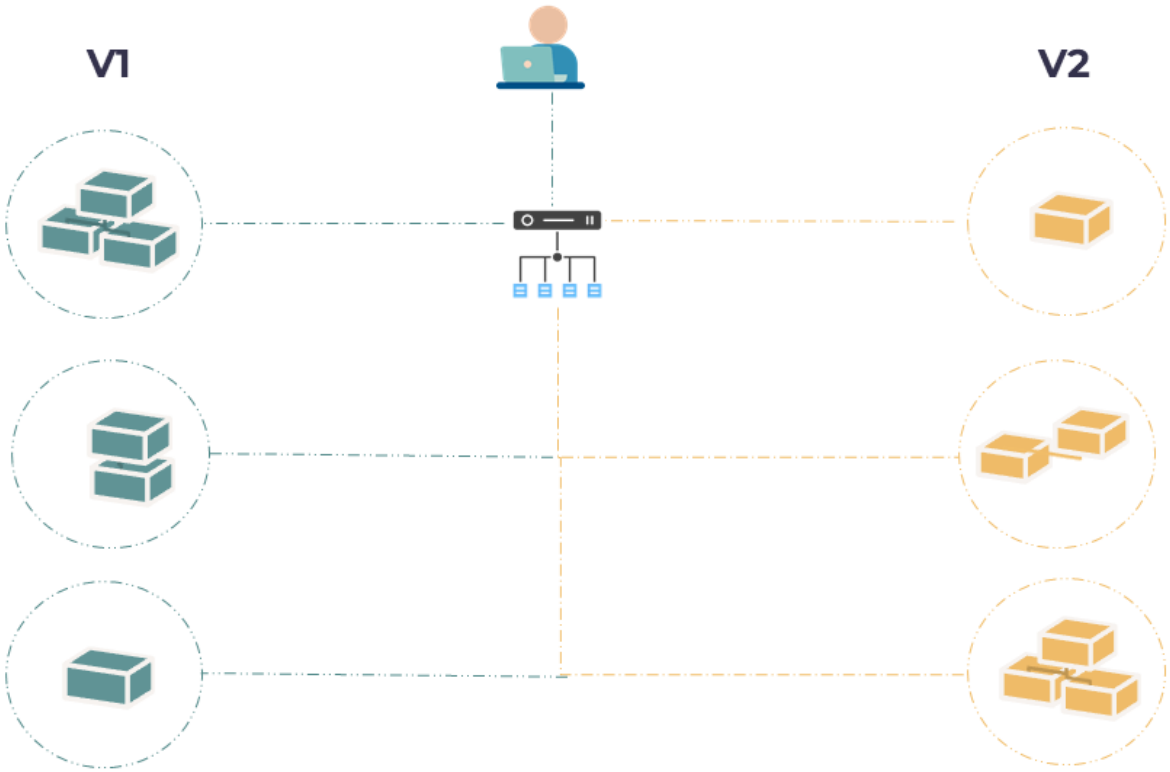


Figure 1.6: Ramped
[9]

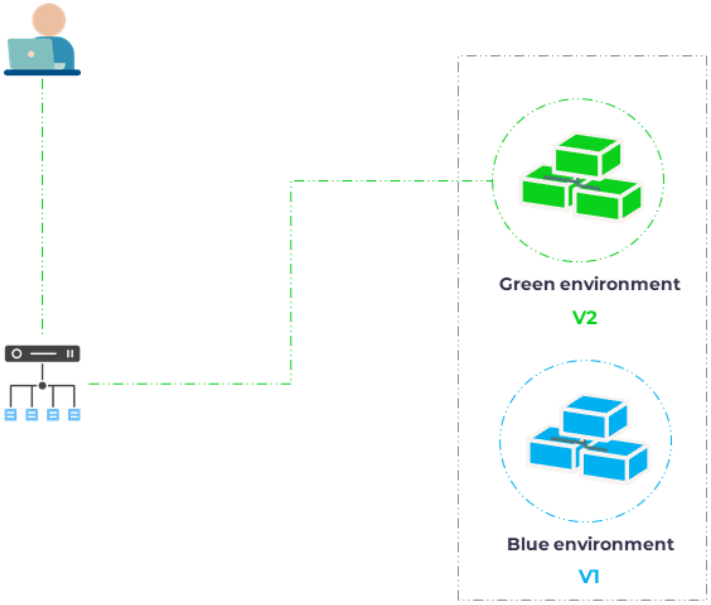


Figure 1.7: Bleu/Vert
[9]

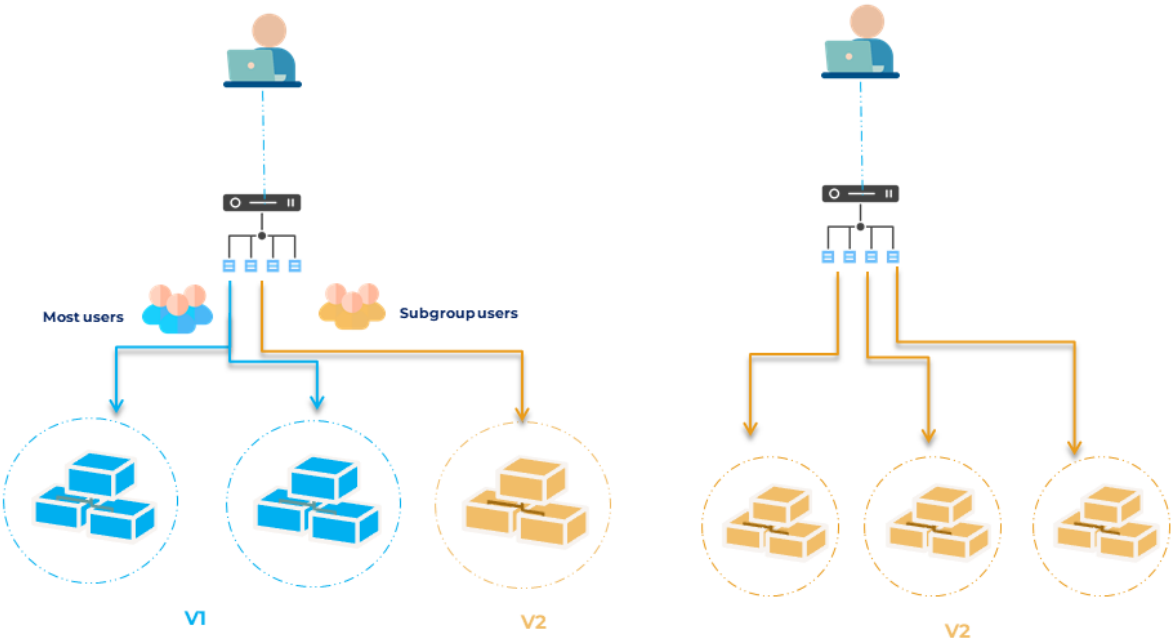


Figure 1.8: Canary et Tests A/B
[9]

Figure 1.9: Tableau comparatif
[9]

1.8 Conclusion

Dans ce premier chapitre nous avons expliqué quelques généralites en relation avec notre théme, Nous avons pu voir ce que sont le devops, le mlops et le déploiement.
Dans le prochain chapitre nous allons exposer la problématique, les contraintes et les différentes propositins pour y remédier

Chapter 2

Machine Learning

2.1 Introduction

Dans ce chapitre nous allons découvrir la problématique que nous devrions résoudre au cours de ce travail, durant lequel nous serons amenés à analyser et à traiter le déploiement des modèles de machine learning.

2.2 Machine Learning

Lorsque nous interagissons avec les banques, achetons en ligne ou utilisons les réseaux sociaux, les algorithmes de machine Learning entrent en jeu pour optimiser, fluidifier et sécuriser notre expérience. Quand nous visitons un site Web complexe comme Facebook, Amazon ou Netflix, il est probable que chaque partie du site Web contienne un modèle d'apprentissage automatique (ML) qui permet d'analyser et tracker l'activité des clients afin d'améliorer l'expérience utilisateurs.
Le machine Learning une discipline scientifique centrée sur le développement, l'analyse et l'implémentation de méthodes automatisables, qui offrent la possibilité à une machine d'évoluer grâce à un processus d'apprentissage. En raison de son omniprésence croissante chaque compagnie est susceptible de le rencontrer et aura besoin de connaissances pratiques dans ce domaine. Un sondage Deloitte 2020 a révélé que 67% des entreprises utilisent l'apprentissage automatique et que 97% l'utilisent ou prévoient de l'utiliser au cours des prochaines années.[10]

Cependant, l'utilisation de l'apprentissage automatique nécessite beaucoup de ressources, une capacité de stockage énorme, une puissance de calcul élevée et une bande passante importante,ect.[11]
Dans ce qui suit nous allons citer certains mots clés relatifs au domaine ML:

1. **GPU:** Unité de traitement spécialisée avec une capacité de calcul mathématique améliorée, ce qui le rend idéal pour l'apprentissage automatique. les GPU dotés d'une mémoire plus grande et plus rapide, tels que le GPU NVIDIA A100 Tensor Core , sont capables de traiter plus rapidement des lots de données. L'utilisation de GPU pour la science des données réduit le temps de traitement de l'apprentissage automatique et augmente la vitesse de calcul, l'entraînement des modèles de Machine Learning peut être effectué 215 fois plus rapidement.
   [12]

2. **CPU:** Unité central de traitement considérer comme le cerveau de l'ordinateur où la plupart des calculs sont effectués. En termes de puissance de calcul, le cpu est l'élément le plus important d'un système informatique. Les calculs de préparation et de prétraitement des données nécessaires à l'‘apprentissage du ML sont généralement effectués sur le CPU, bien que des innovations récentes aient permis d'en réaliser de plus en plus sur les GPU.

3. **Mémoire et stockage:** En particulier pour les plus grands modèles actuels, l'entraînement ML ne fonctionne que lorsqu'il y a une quantité extrêmement importante de données d'entrée à entraîner. Ces données sont extraites du stockage par lots, puis traitées par le CPU dans la mémoire du système avant d'être transmises au GPU.

4. **Réseau:** Les systèmes d'‘IA étant souvent regroupés en clusters pour faire évoluer les performances, ils disposent de plusieurs interfaces Ethernet de 10 gbps ou plus. Certains comprennent également des interfaces InfiniBand ou des interfaces GPU dédiées (NVLink) pour les communications intra-clusters.

5. **Cluster:** L'utilisation des ressources citées précédemment nécessite de les regrouper en cluster, qui désigne un groupe de serveur vu de l'extérieur comme un seul et même serveur logique, le cluster répond aux demandes de traitements d'applications continues au quel un seul serveur peut difficilement répondre et garantis les continuités des services.

   Il y a 2 types de clustering:

   (a) **Actif/Passif:** Le principe est de doubler un serveur avec un second serveur, les deux sont démarrés mais seul un serveur traite les requêtes, Il répond a besoin de disponibilités mais de montée en charge (voir figure 2.1).

   (b) **Actif/Actif:** Le principe est La redondance du serveur actif ce qui fait que la charge de travail est répartie entre eux , Si l'un d'eux tombe ce sont les autres serveurs actifs qui doivent reprendre le relais, il permet de gérer la montée en charge mais la mise en place et plus complexe (voir figure 2.1).

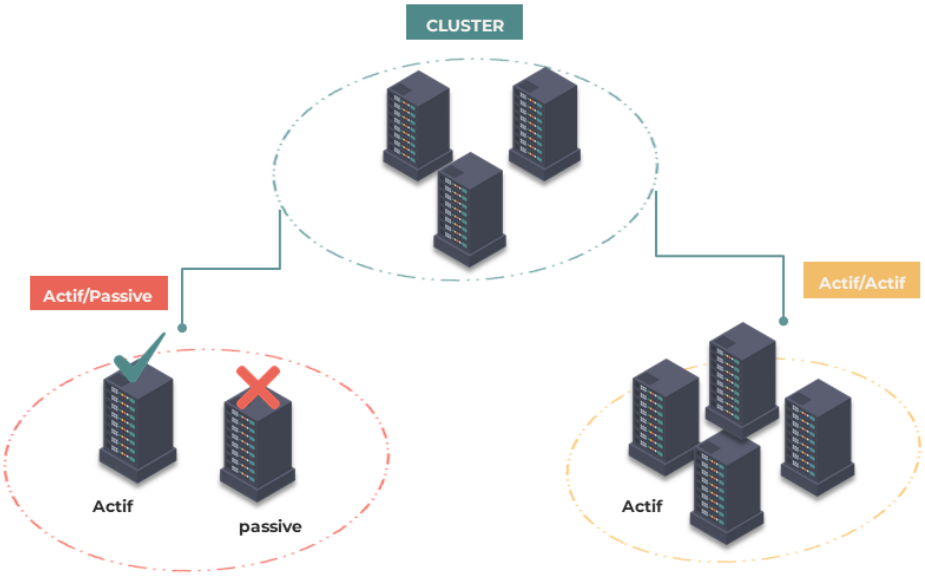


Figure 2.1: Clutsering

Le déploiement des modèles de machine Learning est l'un des processus les plus difficiles pour tirer parti de l'apprentissage automatique. Cela nécessite une coordination entre les scientifiques des données, les équipes informatiques, les développeurs de logiciels et les professionnels de l'entreprise pour garantir que le modèle fonctionne de manière fiable dans l'environnement de production de l'organisation.[13]

2.3 Approche et contexte

Afin de garantir le succès lors des déploiements en production, chaque application nécessite un certain type et stratégie de déploiement cela dépend de l'architecture de son infrastructure qu'elle soit sur le cloud ou on premise, et des facteurs tels que le coût, les risques et le temps de déploiement.

2.4 Déploiement et releases

Le déploiement, comme nous l'avons expliqué précédemment, est défini comme la poussée de tous les composants d'une application d'un environnement à l'autre. Une release comprend le déploiement de l'ensemble de l'application ou de plusieurs applications intégrées dans un environnement de production.[14]

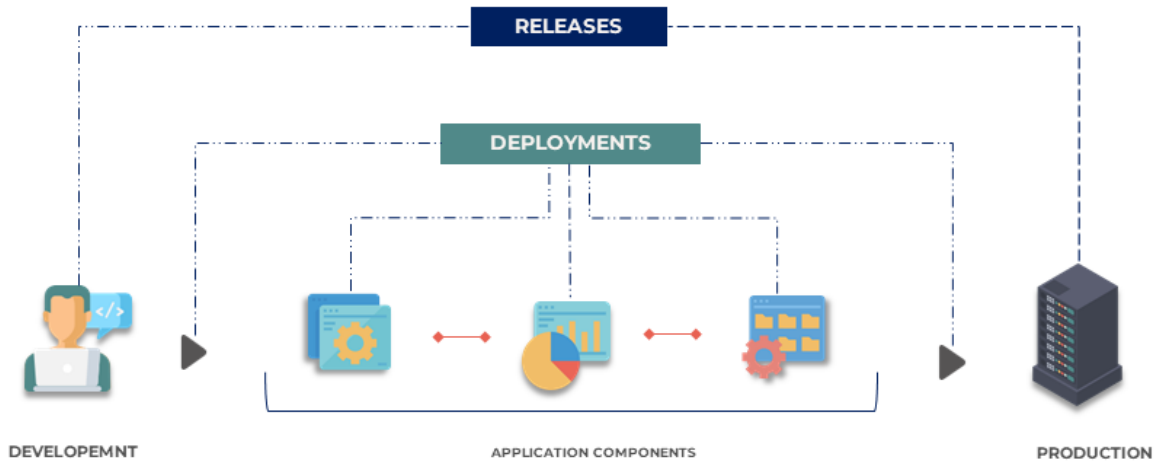


Figure 2.2: Différence entre une livraison et un déploiement

2.5 Services ou microservices

Les applications sont généralement créées de manière monolithique. Autrement dit, tous les éléments de l'application qui peuvent être déployés résident dans cette seule application, Un style architectural traditionnel et qui reste un bon choix pour de nombreuses application. L'inconvénient, c'est que plus celle-ci est volumineuse, plus il devient difficile de l'enrichir de fonctions et de traiter rapidement les problèmes qui surviennent. Avec une approche basée sur des microservices, il est possible de résoudre ces problèmes, d'améliorer le développement et de gagner en réactivité.
Les microservices également connus sous le nom d'architecture de microservice sont un style architectural qui structure une application comme une collection de services qui sont : [15]

- Hautement maintenables et testables
- Faiblement couplés
- Déployables de manière indépendante
- Organisés autour des capacités de l'entreprise
- Détenus par une petite équipe

L'architecture de microservices permet la livraison rapide, fréquente et fiable d'applications volumineuses et complexes. Il permet également à une organisation de faire évoluer sa pile technologique.

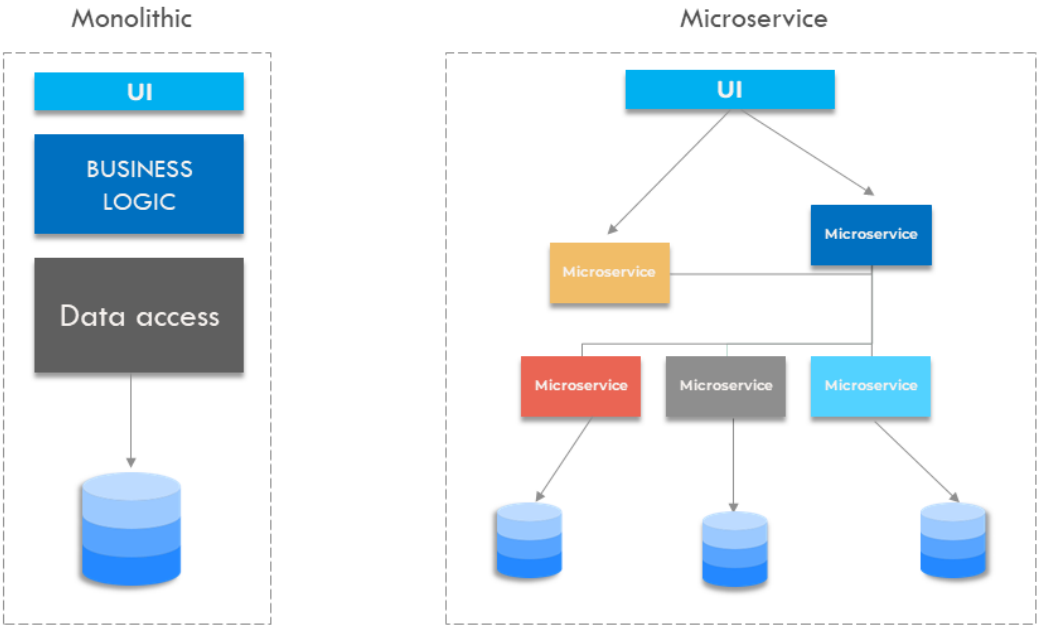


Figure 2.3: Architecture monolithique et microservice

2.6 Études des cas

Dans ce qui suit, nous présentons une étude de cas qui servira de scénario de déploiement pour le reste de ce document.

2.6.1 Scénario:

Dans le cadre des contrôles d'identité pour détecter les personnes recherchées, il est nécessaire de disposer de méthodes plus efficaces au-delà de l'utilisation de documents d'identification traditionnels tels que les permis de conduire et les passeports. Toutefois La reconnaissance faciale est l'une des technologies biométriques les plus répandue et performantes pour identifier un individu.

2.6.2 Secteurs exploitant la Reconnaissance facial

La reconnaissance faciale dans le monde réel, parmi ces secteurs on peut citer: [16]

1. **La Santé:** Afin de permettre aux médecins et aux hôpitaux d'inscrire plus facilement des patients sans avoir à s'aligner ou à interagir avec des formulaires.

2. **Les banques:** Commencent à utiliser la reconnaissance faciale comme outil de connexion sécurisé pour les clients utilisant leurs applications et sites Web bancaires en ligne

3. **Réseaux sociaux:** Comme DeepFace de facebook qui permet de reconnaître automatiquement les visages des personnes sur les photos téléchargées sur les réseaux sociaux.

4. **Éducation:** La numérisation du visage est utilisée dans certaines écoles pour rendre le campus plus sûr contre les menaces potentielles.

5. **Aéroport:** L'identification des passagers avec de faux documents tels que la reconnaissance faciale aux États-Unis a empêché un homme de tenter d'utiliser un faux passeport grâce aux proportions précises de l'appareil photo comparant des photos de visages de personnes enregistrées à partir de documents officiels du gouvernement.

2.6.3 Besoins liée au déploiement

Dans notre cas nous allons procéder au déploiement de la Solution cloud computing intelligent pour la reconnaissance en temps réel réalisé par le binôme Aitouakli Hichem et Mekhazni Fouad.
Une solution à grande échelle car ses fonctions de base nécessitent une grande puissance de calcul et une grande quantité de données qui évoluent en fonction des besoins des utilisateurs.

Parmi les fonctionnalités et besoins identifier, nous pouvons citer.

1. Détection des visages, sexe et objets en temps réel / Capture par webcam
2. Télécharger un modèle d'apprentissage automatique personnalisé
3. Performance
4. Portabilité
5. Scalabilité
6. Flexibilité
7. Sécurité
8. Robustesse

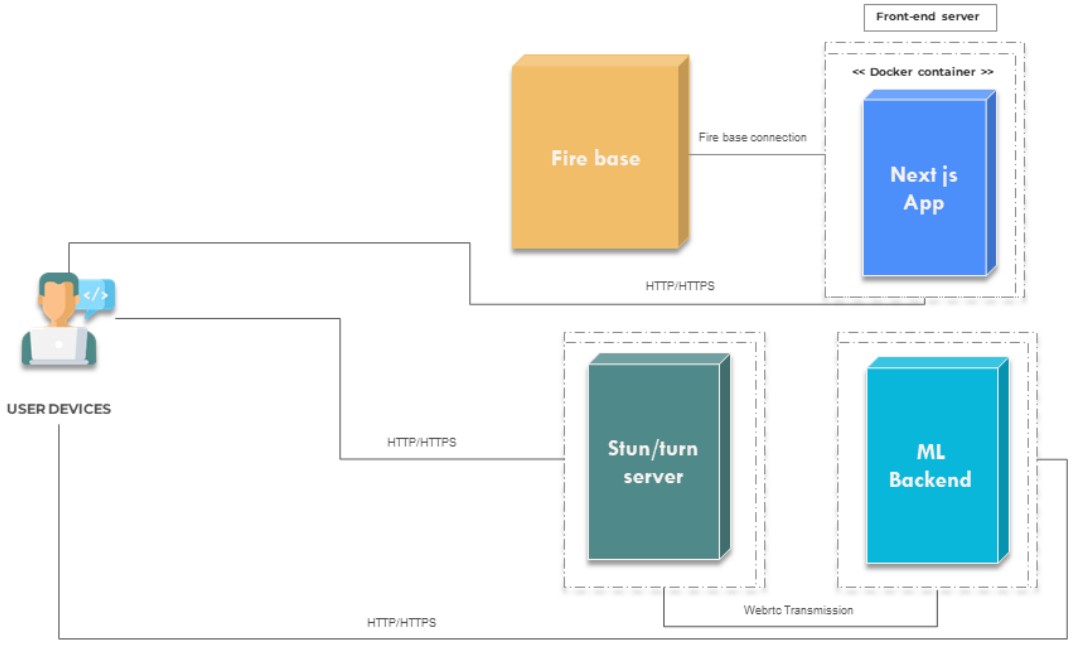


Figure 2.4: Schéma de l'application à deployer

2.6.4 solution proposée

La solution la plus attrayante que nous avons trouvé est le déploiement de ces applications via Cloud Computing.

2.7 Conclusion

Durant ce chapitre nous avons pu traiter la problématique autour de laquelle se construit notre travail, En expliquant les problèmes liés au déploiement complexe du machine Learning et à L'importance de sa mise en production.Quand au chapitre suivant, Celui-ci abordera certains éléments relatifs au cloud computing.

Chapter 3

Cloud computing

3.1 Introduction

Dans ce chapitre nous allons découvrir tous les outils que nous utiliserons au cours de notre travail, nous allons commencer par détailler le Cloud computing, puis nous suivrons avec la virtualisation et la conteneurisation, nous évoquerons aussi les outils relatifs a ces principes, docker et kubernetes, et enfin nous allons clore ce chapitre en décrivant la relation entre kubernetes et le ML ainsi que l'utilisation de kubeflow.

3.2 Cloud Computing

En termes simples, le cloud computing fournit des services informatiques via Internet (le "cloud") tels que des serveurs, du stockage, des bases de données, des réseaux, des logiciels, des analyses et des informations. Afin d'offrir une innovation plus rapide, des ressources flexibles et des économies d'échelle. En règle générale, on ne paye que pour les services en nuage que l'on utilise, ce qui aide à réduire les coûts d'exploitation et à gérer l'infrastructure plus efficacement.[17]

3.3 Modèles de services

Les modèles de services offerts par le cloud computing (voir figure 3.1) il y a: [17]

3.3.1 IaaS:

Infrastructure as a service, permet aux entreprises de louer des ressources informatiques (serveurs, réseaux, stockage, systèmes d'exploitation, etc...) Selon une tarification à l'usage.

3.3.2 PaaS:

Plateforme en tant que service, est un environnement de développement d'applications basé sur le cloud qui fournit aux développeurs tout ce dont ils ont besoin pour créer et déployer des applications. Cela inclut des outils de développement, des bibliothèques de code, des serveurs, des environnements de programmation et des composants d'application préconfigurés.

3.3.3 SaaS:

Software-as-a-service, le logiciel est hébergé sur un serveur distant et les clients peuvent y accéder à tout moment et en tout lieu, à partir d'un navigateur Web ou d'une intégration Web standard. Le fournisseur SaaS se charge des sauvegardes, de la maintenance et des mises à jour. Les solutions SaaS comprennent le progiciel de gestion intégré (ERP), la gestion de la relation client (CRM), la gestion de project, ect.

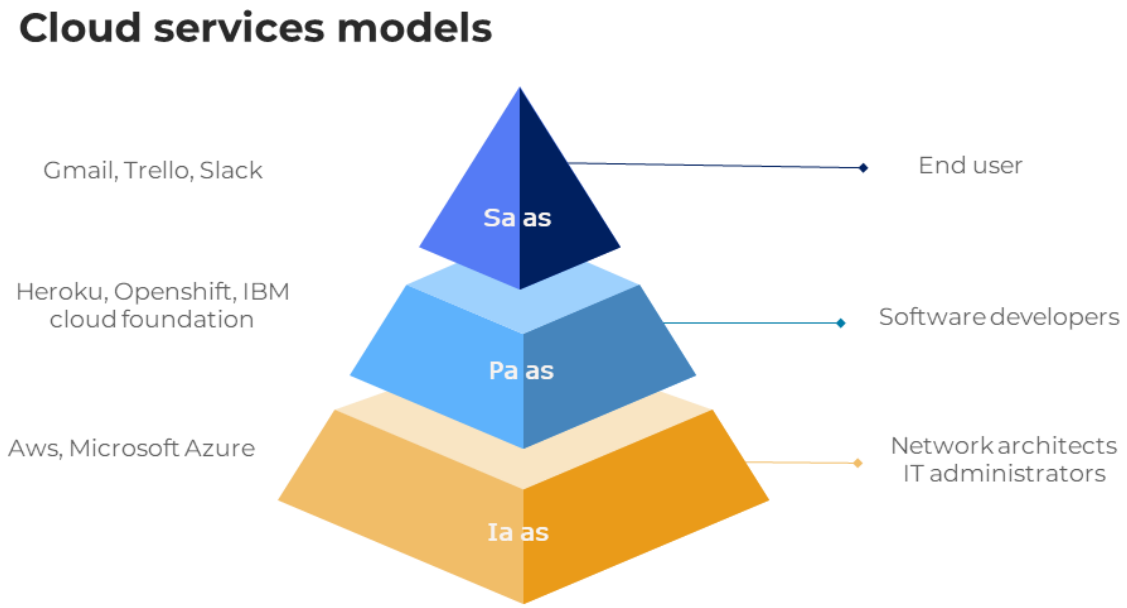


Figure 3.1: Cloud service models
[18]

3.4 Avantages du cloud computing

- **Coût:** Le Cloud Computing élimine les dépenses en capital liées à l'achat de matériel et de logiciels et à la configuration et à l'exploitation de centres de données sur site.
- **La rapidité:** Permet un déploiement rapide et une intégration facile. En effet, le déploiement de la solution cloud se fait en peu de temps et le service est disponible rapidement.
- **Sécurité et fiabilité:** De nombreux fournisseurs de cloud proposent un large éventail de politiques, de technologies et de contrôles qui renforcent globalement votre posture de sécurité, en aidant à protéger vos données, vos applications et votre infrastructure contre les menaces potentielles.
- **Flexibilité:** Les applications et services que vous utilisez dans le Cloud sont accessibles de n'importe où, tant que vous disposez d'un terminal et d'une connexion Internet. Le Cloud Computing s'adapte aux besoins évolutifs de votre entreprise en un temps-record.

- **Productivité:** Les plus grands services de cloud computing fonctionnent sur un réseau mondial de centres de données sécurisés, qui sont régulièrement mis à niveau vers la dernière génération de matériels informatiques rapide et efficace.

3.5 Virtualisation

Un mécanisme informatique qui se résume à faire fonctionner sur une même machine physique, plusieurs systèmes comme s'ils fonctionnaient sur des machines physiques distinctes. En d'autres termes la virtualisation est la création de plusieurs machines virtuelles (VM) à partir d'une seule machine physique à l'aide d'un logiciel appelé hyperviseur. La virtualisation permet donc une utilisation plus efficace du matériel informatique physique et offre un meilleur retour sur l'investissement matériel d'une organisation.

3.6 Machine virtuel et conteneur

3.6.1 Machine virtuel (VM)

Est une virtualisation au niveau matériel, Chaque VM exécute son propre système d'exploitation (OS) et se comporte comme un ordinateur indépendant, même si elle ne fonctionne que sur une partie du matériel informatique sous-jacent.

3.6.2 Conteneurisation

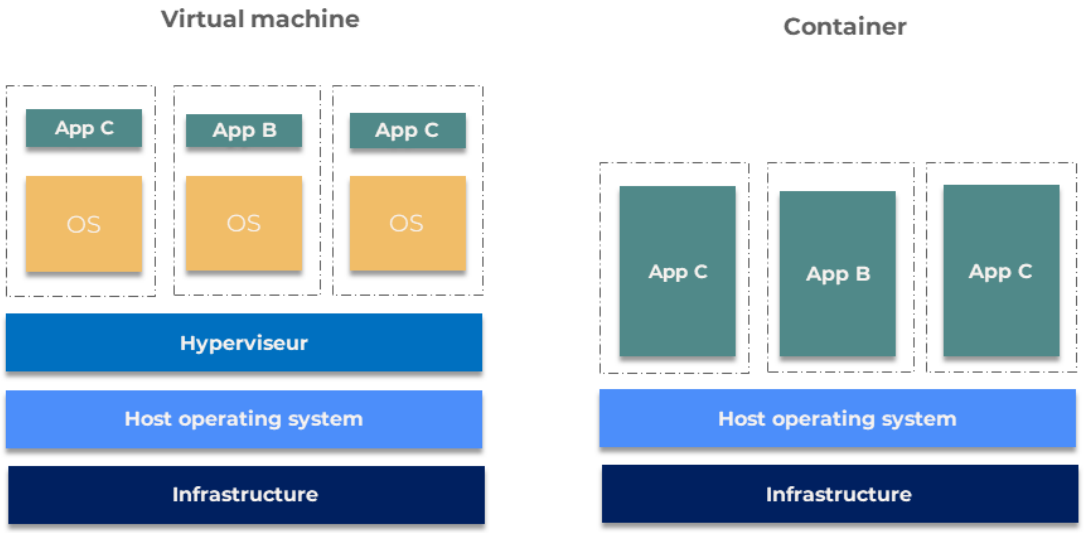
Un conteneur est similaire à une machine virtuelle mais la virtualisation s'effectue au niveau du système d'exploitation .Contrairement aux applications classiques, les applications conteneurisées partagent un environnement de système d'exploitation (noyau), elles utilisent donc moins de ressources que des machines virtuelles complètes, réduisent la pression sur la mémoire de l'hôte et sont compatibles avec les techniques émergents telles que le cloud computing, l'approche CI/CD et le DevOps.
Les conteneurs sont si rapides et petits que nous pouvons les utiliser pour diviser une application monolithique en plusieurs petits services qui communiquent entre eux. Chaque petit service peut ainsi fonctionner dans son propre conteneur, séparément des autres, afin qu'il puisse être entretenu ou réparé sans craindre de briser les autres parties.

Parmi les systèmes de conteneurisations les plus populaire et utilisée et celui que nous allons aborder dans notre travail est le soi-disant Docker.

3.6.3 Docker

Docker est une plate-forme de conteneurs open source sécurisée et économique qui facilite la création de conteneurs et d'applications basées sur des conteneurs.On peut créer, tester, déployer et exécuter des applications sur n'importe quel environnement à l'aide de Docker. Cela évite les conflits potentiels au niveau de chaque composant et résout ainsi

les problèmes de version, d'incompatibilité ou encore d'implémentation qui surviennent souvent lors du développement de solutions.[19]



[20]

Figure 3.2: machine virtuelle et conteneur

3.6.4 Composants Docker

Dans ce qui suit nous allons décrire les différents composants dockerDans ce qui suit nous allons décrire les différents composants docker [21]: (voir figure 3.3)

1. **Docker Engine:** Est le système qui fait tourner l'ensemble de Docker. Il s'agit d'un moteur qui suit une architecture client-serveur comprenant principalement trois composants :
   - **Serveur (docker daemon):** Il s'agit du démon docker appelé dockerd Il peut créer et gérer des images docker, c'est-à-dire des conteneurs, des réseaux.
   - **Rest API:** Il est utilisé pour indiquer au démon docker ce qu'il doit faire.
   - **Interface de ligne de commande (CLI):** C'est un client qui est utilisé pour saisir les commandes du docker.
2. **Image docker:** Un modèle qui contient toutes les instructions pour crée le conteneur Docker, Elle est composée de plusieurs couches empaquetant toutes les installations, dépendances, bibliothèques, processus et codes d'application nécessaires pour un environnement de conteneur pleinement opérationnel.
3. **Registre docker:** Stocke les images Docker. Docker Hub est un registre public que tout le monde peut utiliser, et Docker est configuré pour rechercher des images sur Docker Hub par défaut. Vous pouvez même exécuter votre propre registre privé.

4. **Conteneur :** Les conteneurs sont des environnements encapsulés dans lesquels les applications sont exécutées. Un conteneur est défini par l'image et les options de configuration.

3.6.5 Architecture docker

En se référant à la documentation officielle. Docker utilise une architecture client-serveur. Le client Docker communique avec le démon Docker, qui effectue le gros du travail de création, d'exécution et de distribution de vos conteneurs. Le client et le démon Docker peuvent s'exécuter sur le même système, ou vous pouvez connecter un client Docker à un démon Docker distant.

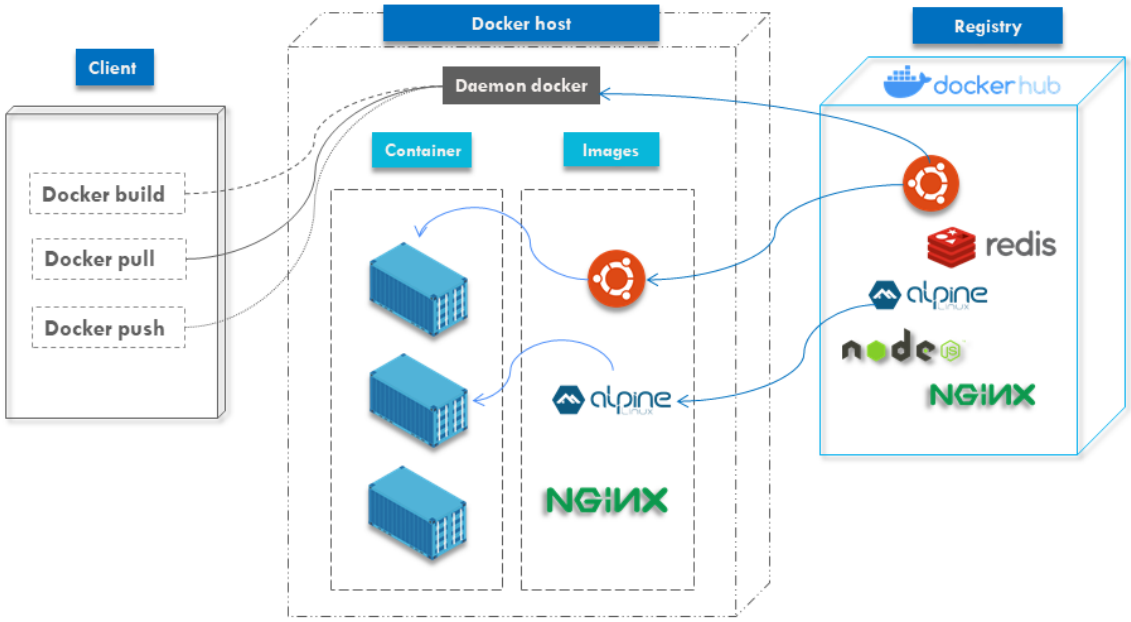


Figure 3.3: Architecture docker

3.7 Types de virtualisation

3.7.1 La virtualisation de systèmes d'exploitation:

Est une technique consistant à faire fonctionner en même temps, sur un seul ordinateur, plusieurs systèmes d'exploitation comme s'ils fonctionnaient sur des ordinateurs distincts.

1. **Virtualisation des serveurs:** Consiste à diviser un serveur physique en plusieurs serveurs virtuels uniques et isolés à l'aide d'un hyperviseur qui gère différents OS invités. Les multiples instances de serveur qui en résultent exécutent leurs propres systèmes d'exploitation indépendamment, s'exécutent en parallèle et ont des capacités différentes, mais exploitent le même matériel sous-jacent.Représentes de grands avantages aux entreprises : diminue le nombre de serveurs utilisés, optimises les ressources et réduit les coûts de matériel et maintenance.

2. **Virtualisation de poste de travail:** Comprends l'exécution de plusieurs systèmes d'exploitation de bureau, chacun sur sa propre machine virtuelle et sur le même ordinateur. L'un des principaux avantages de la virtualisation de bureau est que les utilisateurs peuvent accéder à tous leurs fichiers et applications personnels sur n'importe quel PC, ce qui signifie qu'ils peuvent travailler de n'importe où sans avoir besoin d'utiliser leur propre ordinateur de travail. Cela réduit également les coûts de licence et de mise à jour logicielle. La maintenance et la gestion sont simples car tous les postes de travail virtuels sont hébergés au même endroit.
   Il existe deux types de virtualisation de postes :

   - **VDI:** Virtual desktop infrastructure, La VDI consiste à abstraire et à héberger plusieurs sessions de bureau virtualisées sur un serveur principal centralisé. Ces bureaux virtuels sont ensuite mis à la disposition des utilisateurs accessibles via des terminaux clients légers.
   - **La virtualisation de poste local**: Permet à l'utilisateur d'exécuter un ou plusieurs systèmes d'exploitation supplémentaires sur cet ordinateur et de passer d'un système d'exploitation à un autre selon les besoins, sans rien changer au système d'exploitation principal

3.7.2 La virtualisation d'application:

Permet d'exécuter des logiciels d'application sans installation directe sur le système d'exploitation de l'utilisateur. Cette méthode diffère de la virtualisation complète du poste de travail (mentionnée ci-dessus) car seule l'application s'exécute dans un environnement virtuel, le fonctionnement du système d'exploitation de l'appareil de l'utilisateur final étant inchangé.

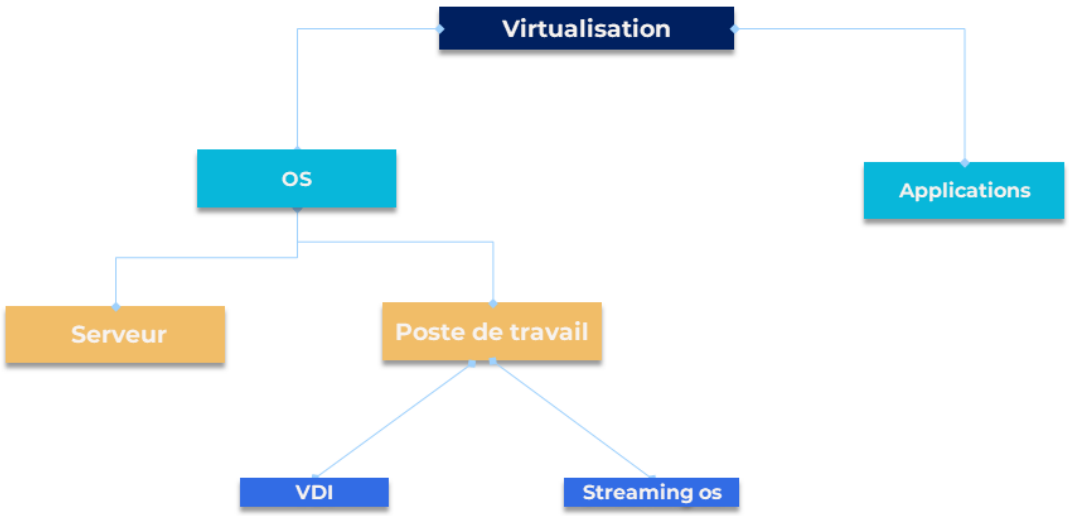


Figure 3.4: Virtualisation

3.8 Kubernetes (orchestration)

3.8.1 Définition

Également connu sous le nom de K8s, est un système open source permettant d'automatiser le déploiement, la mise à l'échelle et la gestion des applications conteneurisées. conçu pour orchestrer des architectures de clusters de conteneurs logiciels de manière extrêmement agile, en donnant la possibilité de gérer l'allocation des ressources machine sous-jacentes à la volée.
Crée par Google, Développé en tirant les leçons de l'écriture et de l'exploitation du Project Borg pendant plus de 15 ans. un système de clustering utilisé par le géant américain depuis des années. Google Search, Maps, Gmail, Youtube, etc. Tous ses services reposent sur des clusters de centaines de containers pilotées par Borg, Par la suite Google en fait don à la CNCF (Cloud Computing Foundation). [22]

3.8.2 Fonctionnalités

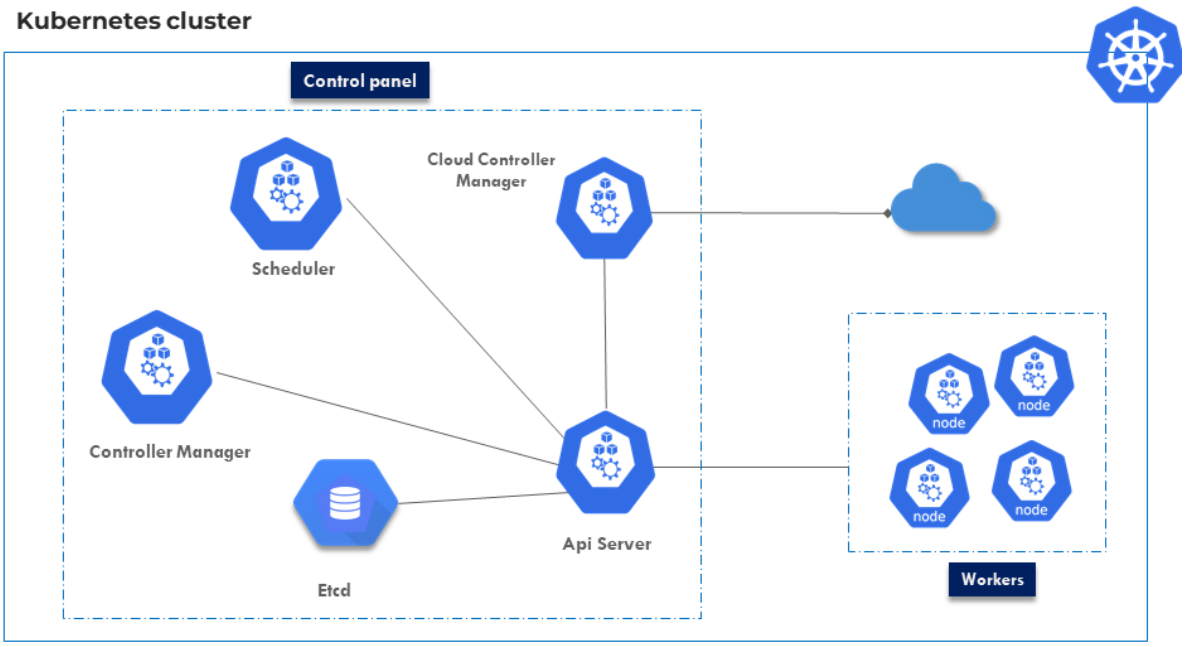
Parmi les fonctionnalités de bases de K8s, nous citons:

1. Orchestrer des conteneurs sur plusieurs hôtes
2. Optimiser l'utilisation de votre matériel afin de maximiser les ressources requises pour l'exécution de vos applications d'entreprise
3. Contrôler et automatiser les déploiements et mises à jour d'applications
4. Monter et ajouter des systèmes de stockage pour exécuter des applications avec état
5. Mettre à l'échelle des applications conteneurisées et leurs ressources à la volée.
6. Gérer des services de façon déclarative et garantir ainsi que les applications déployées s'exécutent toujours de la manière dont vous les avez déployées
7. Vérifier l'intégrité de vos applications et les réparer automatiquement grâce au placement, au démarrage, à la réplication et à la mise à l'échelle automatique

3.8.3 Architecture

Un environnement exécutant Kubernetes comprend les composants de base suivants (voir figure 3.5) 20:

- **Le nœud master (master node):** Exécute le plan de contrôle Kubernetes qui contrôle l'ensemble du cluster , composé d'un api-server, un Schedule, un etcd, un Controller managé et un Cloud Controller manager.
- **Le nœud de travail (worker node):** Se charge de l'exécution et de la mise en réseau des applications conteneurisées. Il effectue toutes les actions déclenchées via l'API Kubernetes, qui s'exécute sur le nœud maître. Se compose d'un servie kubelet, container runtime et d'un kube proxy.
- **Api-server:** Interface utilisée pour gérer, créer et configurer les clusters Kubernetes. C'est le moyen d'interaction entre les utilisateurs et le cluster Kubernetes.



[23]

Figure 3.5: Architecture kubernetes

- **Cluster store (etcd):** Base de données de Kubernetes ou informations de configuration, d'exécution et d'état y sont stockées. C'est le seul composant " stateful " du master, il est crucial pour le bon fonctionnement de Kubernetes.
- **Scheduler:** Planificateur qui affecte les conteneurs à des nœuds en prenant en compte le taux de charge du noeud, sa capacité et les besoins en ressource.
- **Controller manager:** Exécutes les contrôleurs qui surveillent en permanence l'état de votre cluster, puis effectuent ou demandent des modifications si nécessaire. Chaque contrôleur essaie de rapprocher l'état actuel du cluster de l'état souhaité.
- **Kubelet:** Permet la communication en worker et master node.
- **Container runtime:** Un environnement dans lequel les conteneurs peuvent être exécutés
- **Kubeproxy:** Chargé d'activer la communication entre les services au sein du cluster. [22]

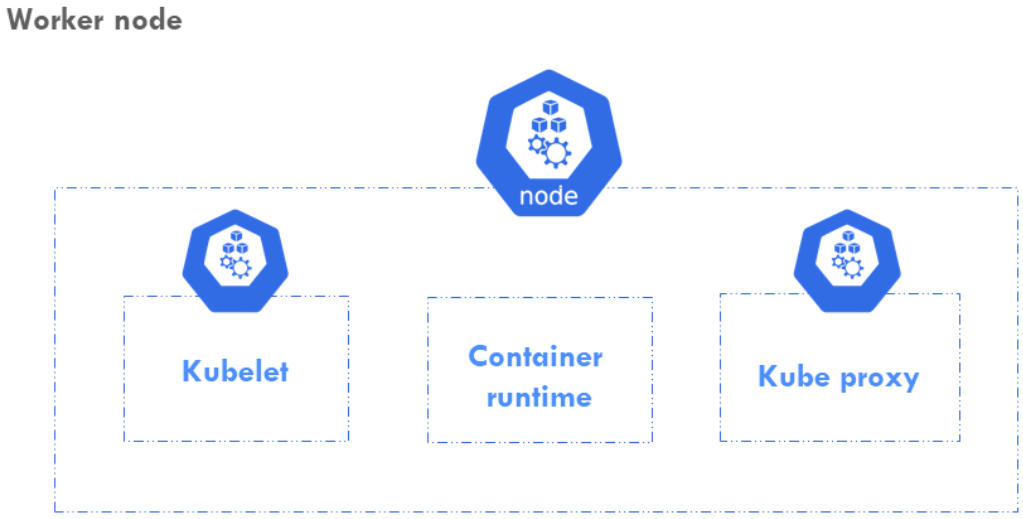


Figure 3.6: Worker node

3.9 Pods et services

Nous décrirons certains des concepts suivants concernant les k8 (voir figure 3.7)

3.9.1 Pods et nodes

1. **Pod:** Il s'agit de la plus petite unité de base d'une application Kubernetes, un pod peut contenir un unique conteneur (cas plus courant) ou plusieurs conteneurs (cas plus avancé), Au sein du système Kubernetes, les conteneurs d'un même pod partagent les mêmes ressources de calcul. Ces ressources de calcul sont regroupées pour former des clusters.

2. **Node:** Une machine de travail qui exécute des charges de travail Kubernetes. Il peut s'agir d'une machine physique ou d'une machine virtuelle (VM). Chaque nœud peut héberger un ou plusieurs pods.

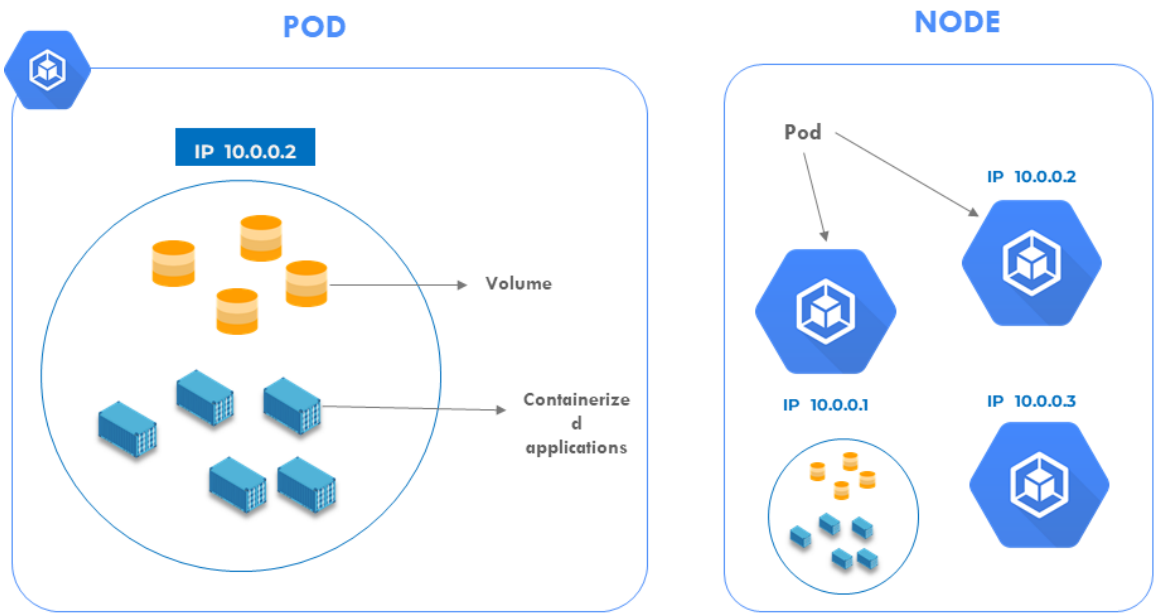


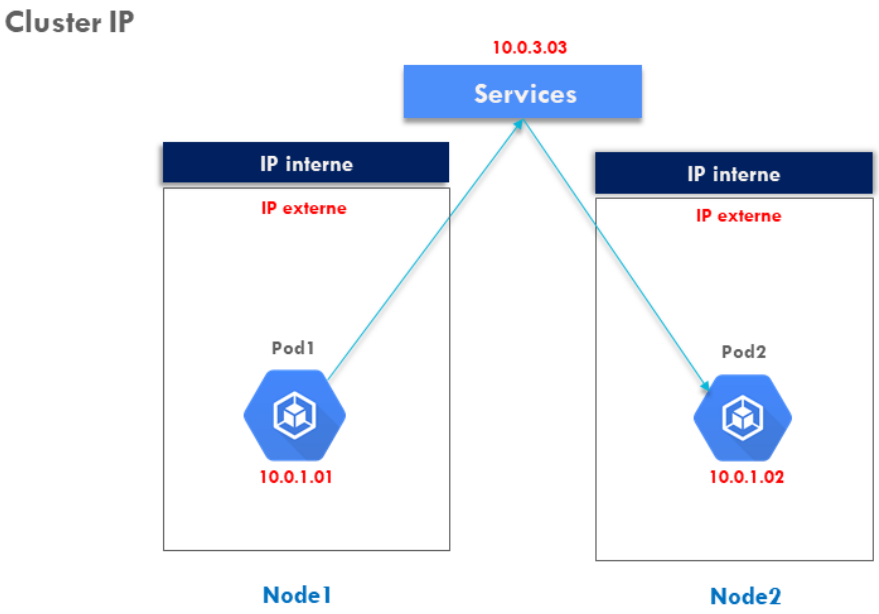
Figure 3.7: Pods et nodes

3.9.2 Services

Un service est chargé d'autoriser l'accès au réseau à un ensemble de pods. Il peut être défini comme une abstraction sur le dessus du pod qui fournit une adresse IP et un nom DNS uniques par lesquels les pods sont accessibles. Accorde aux applications de recevoir du trafic et peuvent être exposés de différentes manières et permet de gérer la configuration de l'équilibrage de charge.

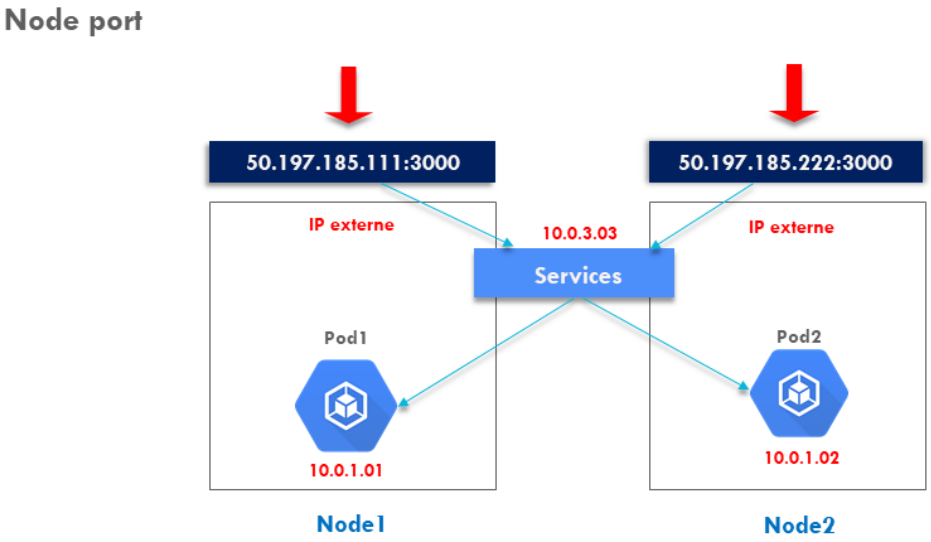
Les Types possibles[24]:

- **Cluster IP:** Est le service Kubernetes par défaut. expose l'IP interne du cluster Kubernetes (voir figure 3.8).
- **NodePort:** Expose le Service sur l'IP de chaque nœud sur un port statique (voir figure 3.9).
- **LoadBalancer:** Expose le Service à l'extérieur en utilisant l'équilibreur de charge d'un fournisseur de cloud computing (voir figure 3.10)
- **Ingress:** Permet d'accéder à vos services depuis l'extérieur du cluster Kubernetes. IL est le plus utile si vous souhaitez exposer plusieurs services sous la même adresse IP (voir figure 3.11).



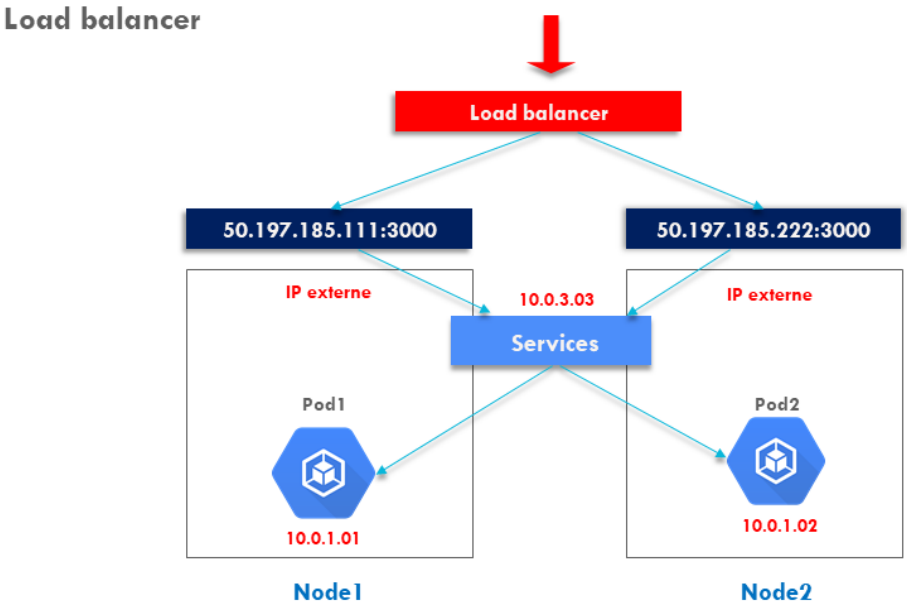
[24]

Figure 3.8: Cluster IP



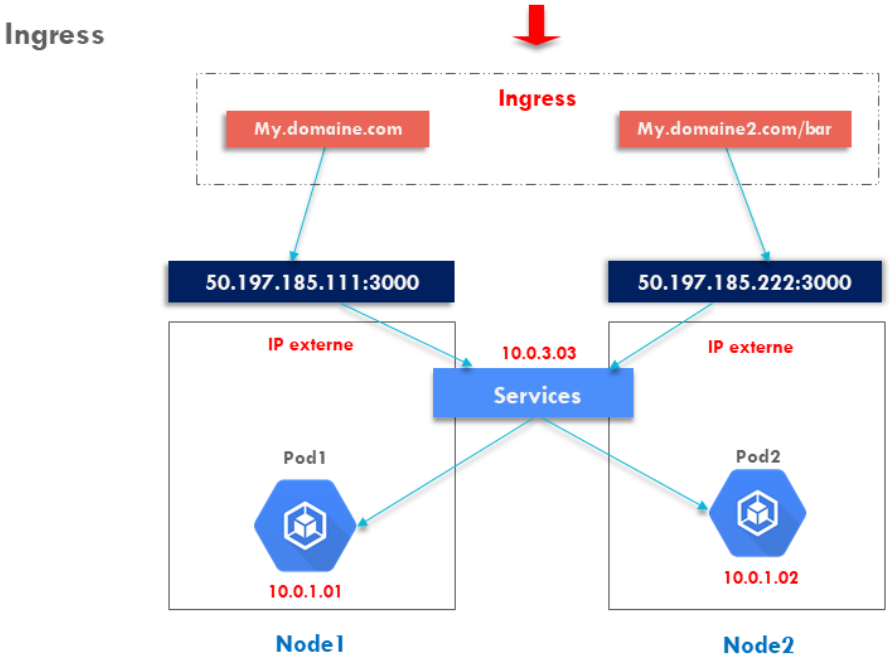
[24]

Figure 3.9: Node port



[24]

Figure 3.10: Load balancer



[24]

Figure 3.11: Ingress

3.10 ML et kubernetes

h Un des objectifs de notre projet est de déployer efficacement des modèles d'apprentissage automatique, les mettre à l'échelle, les distribuer sur des clusters de serveurs.
Kubernetes est l'un des moyens les plus populaires qui gère les modèles d'apprentissage automatique dans des conteneurs. Ces modèles peuvent être facilement mis à l'échelle et planifiés lorsqu'ils sont conteneurisés, la gestion des performances de la charge de travail peut être automatisée. À l'aide de Kubernetes, les organisations peuvent intégrer des flux de travail d'apprentissage automatique de bout en bout dans des conteneurs.
Une boîte à outils spécifique appeler Kubeflow a également été développées pour simplifier le processus de déploiement de modèles d'apprentissage automatique sur Kubernetes.
[25] [26]

3.10.1 Kubeflow

Sur la base des objectifs décrits précédemment, nous avons choisi Kubeflow comme plateforme pour nos services (voir figure 3.2).
Kubeflow est un projet open source dédié à rendre les déploiements de flux de travail d'apprentissage automatique (ML) sur Kubernetes simples, portables et évolutifs. Il ne vise pas à recréer d'autres services, mais à fournir un moyen simple de déployer les meilleurs systèmes open source pour le ML sur diverses infrastructures. Ainsi, partout où Kubernetes s'exécute, Kubeflow peut s'exécuter.
Kubeflow s'appuie sur la conteneurisation et le découplage en micro-services pour former une plateforme portable pouvant s'exporter sur n'importe quel environnement capable de déployer un cluster Kubernetes (notamment les principaux cloud providers).[27]
Kubeflow est livré avec de nombreux composants que l'on peut choisir d'installer ou non, et donc chacun a un rôle spécifique : [28]

- **Kubeflow pipline:** Permet de construire, exécuter et administrer des pipelines ML automatisés.
- **Nootbook server:** Permet de créer des serveurs Jupyter à la demande pour chaque utilisateur.
- **KFserving:** Utilisé pour exposer des modèles de Machine Learning.
- **Katib:** Utilisé pour optimiser les hyper-paramètres des modèles de Machine Learning en profitant de la puissance de calcul et du caractère distribué de Kubernetes.

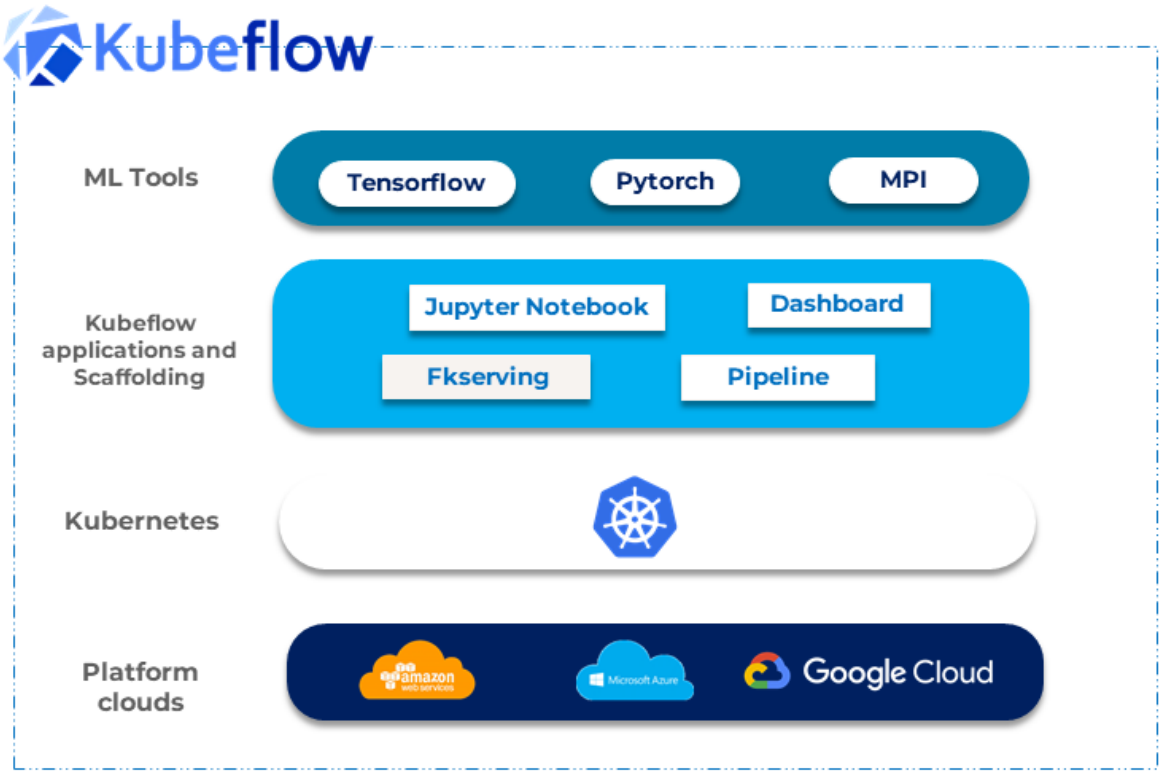


Figure 3.12: Architecture kubeflow

3.11 Conclusion

Jusqu'à présent, nous avons vu la plupart des concepts nécessaires liés aux outils que nous utiliserons dans ce travail, puis nous avons détaillé ces outils comme docker, kubernetes et kubeflow, et expliqué leurs composants. Dans le chapitre suivant, nous détaillerons la procédure de déploiement local via kubernetes.

Chapter 4

Déploiement local

4.1 Introduction

Dans ce chapitre, Nous allons démontrer une implémentation locale avec un seul nœud utilisant minikube et avec plusieurs nœuds utilisant kubeadm tout en précisant les problèmes rencontrés et les solutions adaptées.

4.2 Déploiement avec un seul nœud (Minikube)

Pour démarrer avec kubernetes, Minikube est l'outil idéal qui permet d'exécuter un cluster Kubernetes à nœud unique dans une machine virtuelle (VM) sur un ordinateur portable. Nous allons commencer par le déploiement d'une simple application appelé (getting-started) Qui s'agit d'une simple liste de taches.

1. **Installation:**

   Nous allons procéder a l'installation de MInikube, docker et Kubectl qui est l'outil de ligne de commande Kubernetes qui permet d'exécuter des commandes sur des clusters Kubernetes.

```
$ choco install kubernetes-cli
$ choco install minikube
$ choco install Docker-cli
$ choco install Docker
```

2. **Création de l'image de notre application:**

   Le processus de création d'image kubernetes consiste a créé une image docker puis la pousser dans un registre ensuite la référencier depuis un pod.

(a) **Crée un dockerfile:** Simple document texte qui contient une série de commandes que Docker utilise pour créer une image.

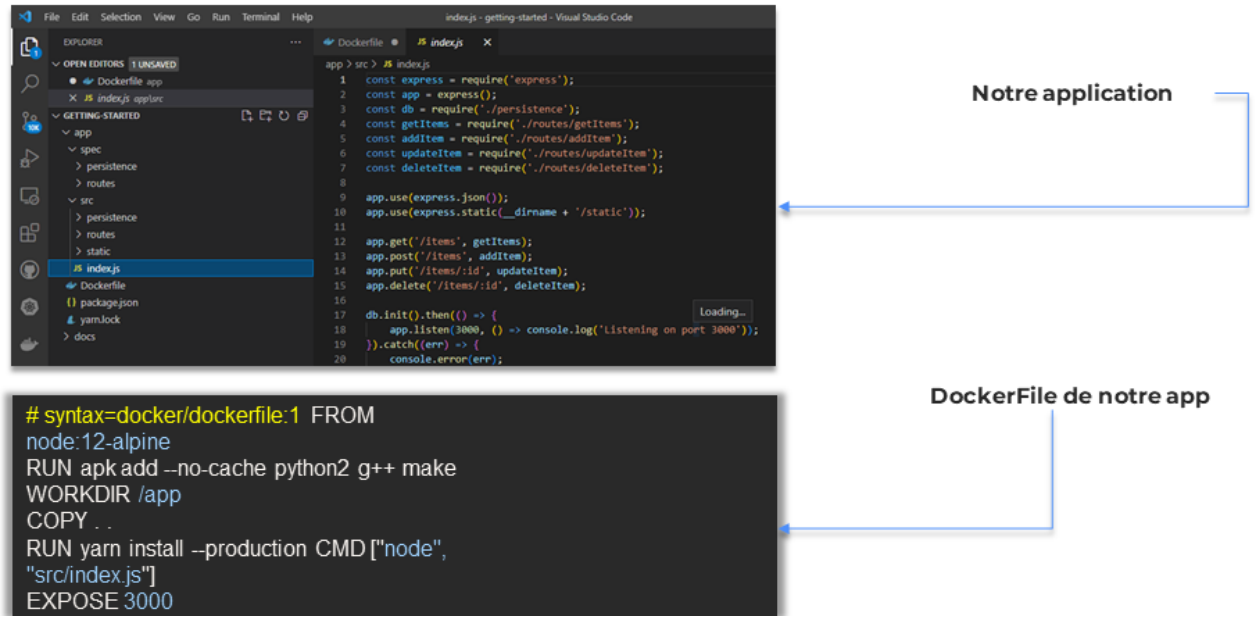


Figure 4.1: DockerFile

(b) **Crée l'image du conteneur et Mettre l'image dans le regitre docker**

```
$ docker build -t nom-de-l'image
$ docker build -t getting-started
```

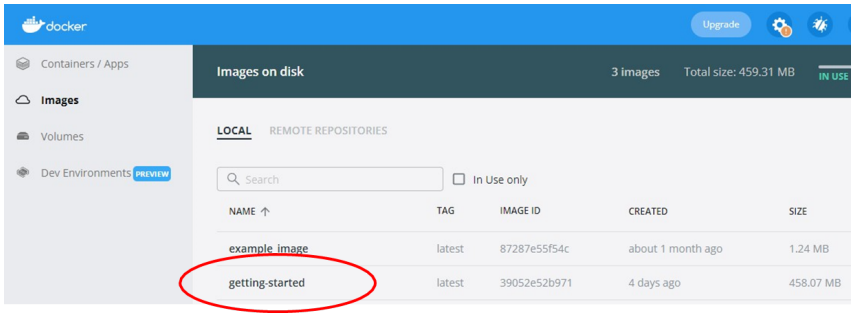


Figure 4.2: Création de l'image

(c) **Deployer l'application avec docker:** Nous allons lancer un conteneur docker afin de tester le deploiement de notre application

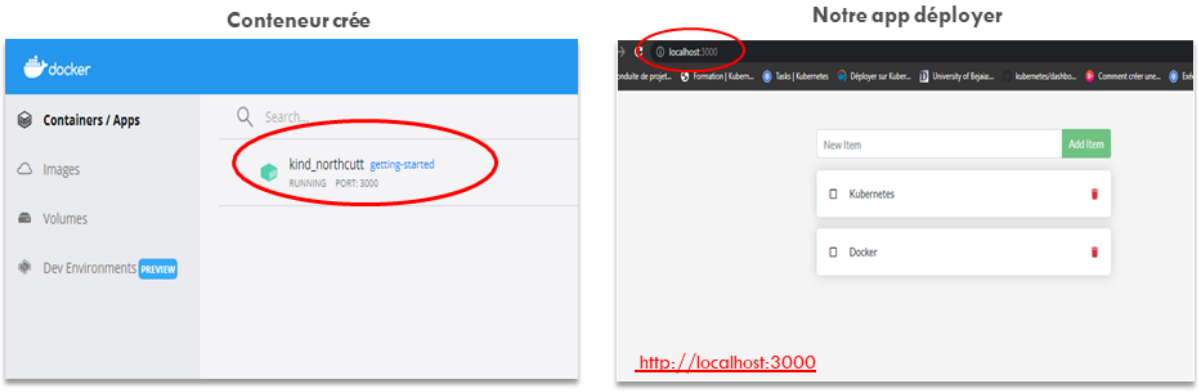


Figure 4.3: Déploiement avec Docker

(d) **Deployer l'application avec kubernetes:** Nous devons tout d'abord lancer un cluster kubernetes pour ce faire, nous pouvons utiliser sois Minikube sois l'environnement kubernetes complet intégré dans docker desktop.

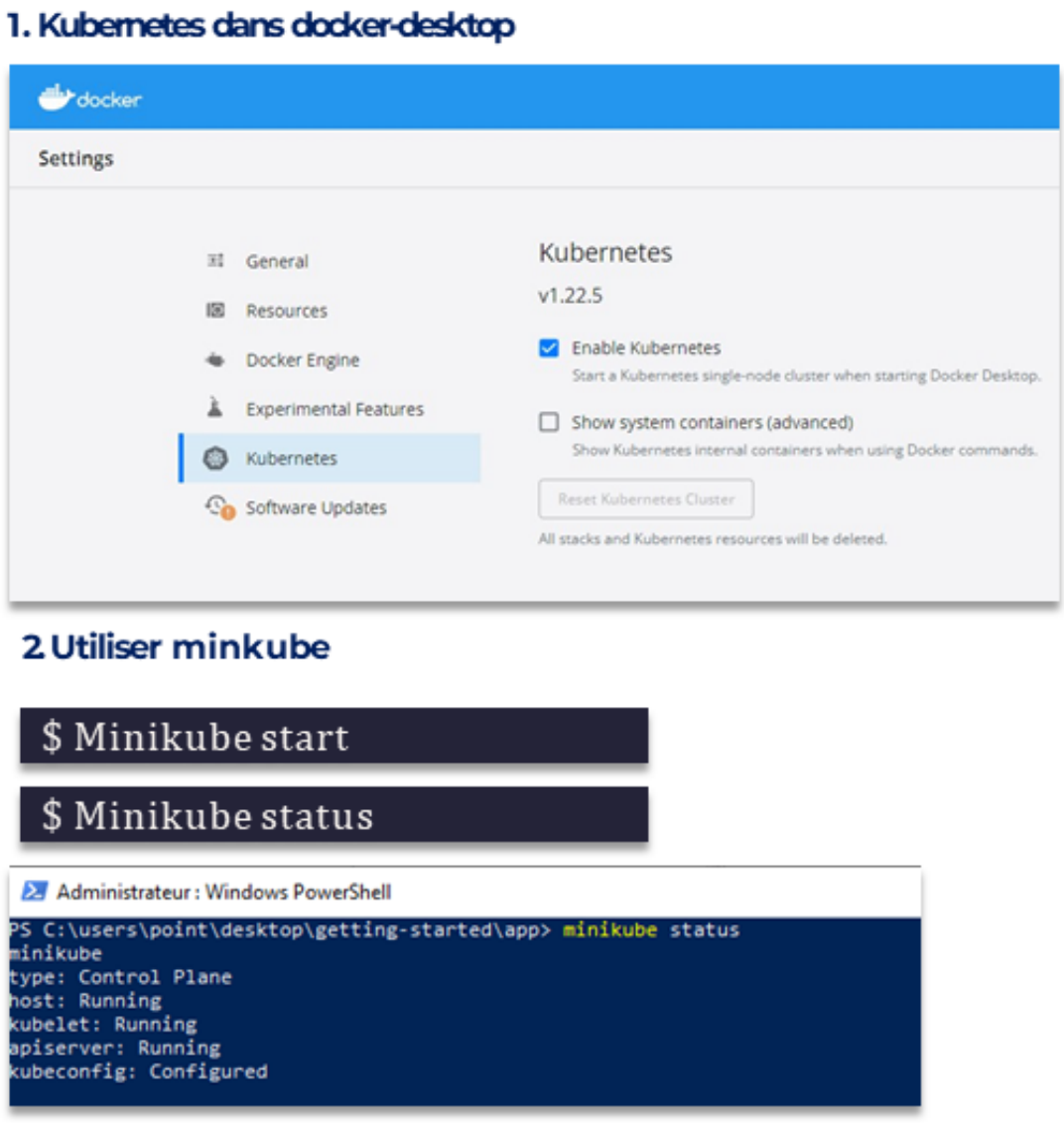


Figure 4.4: Déploiement avec Kubernetes

Ensuite nous allons créer un fichier yaml qui décrit tous les composants et configurations de pour notre application.

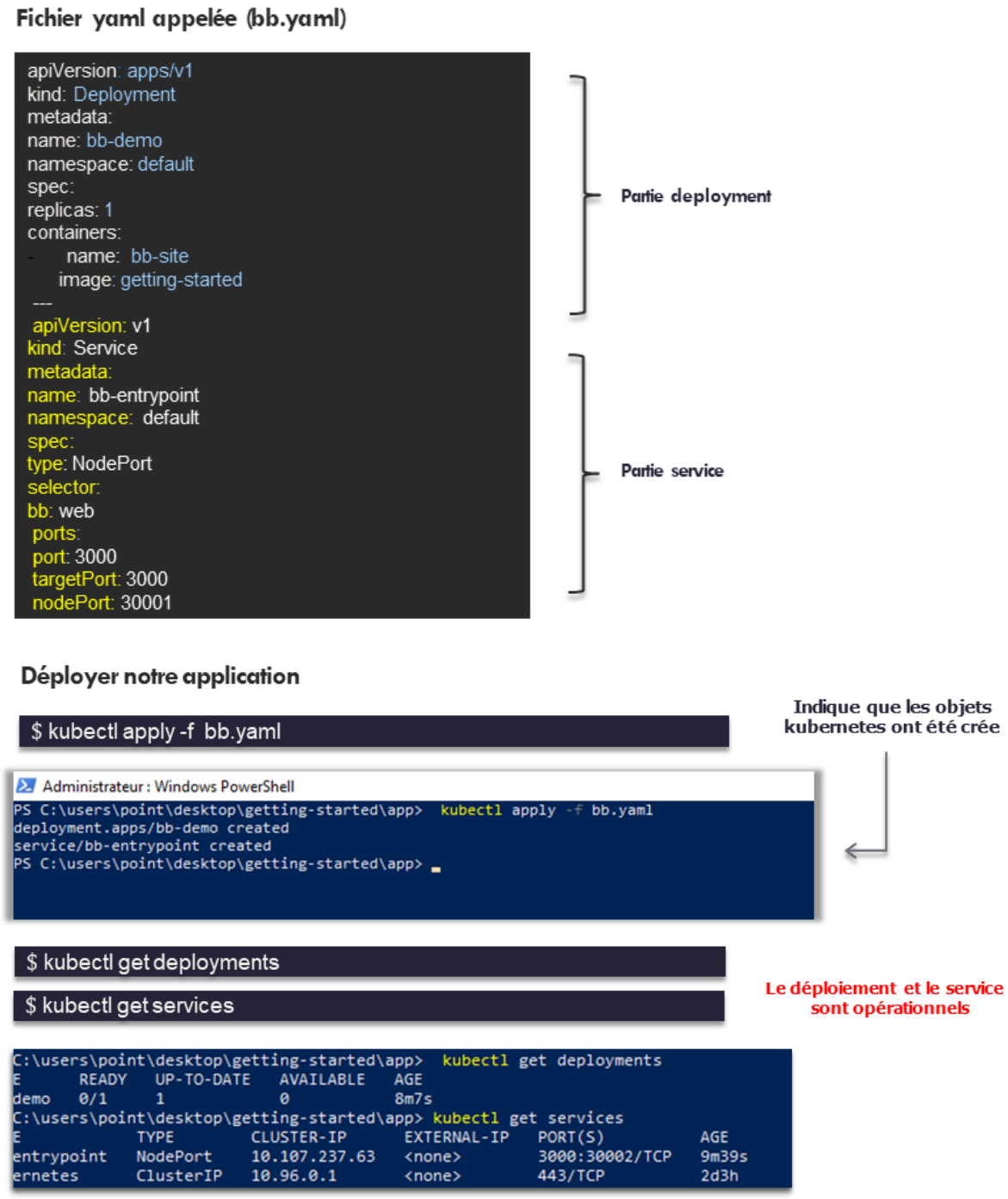


Figure 4.5: Déploiement avec Kubernetes

3. **Erreur du déploiement**
   Nous avons rencontré l'erreur ImagePullBackoff qui signifie que Le pod n'arrive pas a extraire l'image du conteneur.

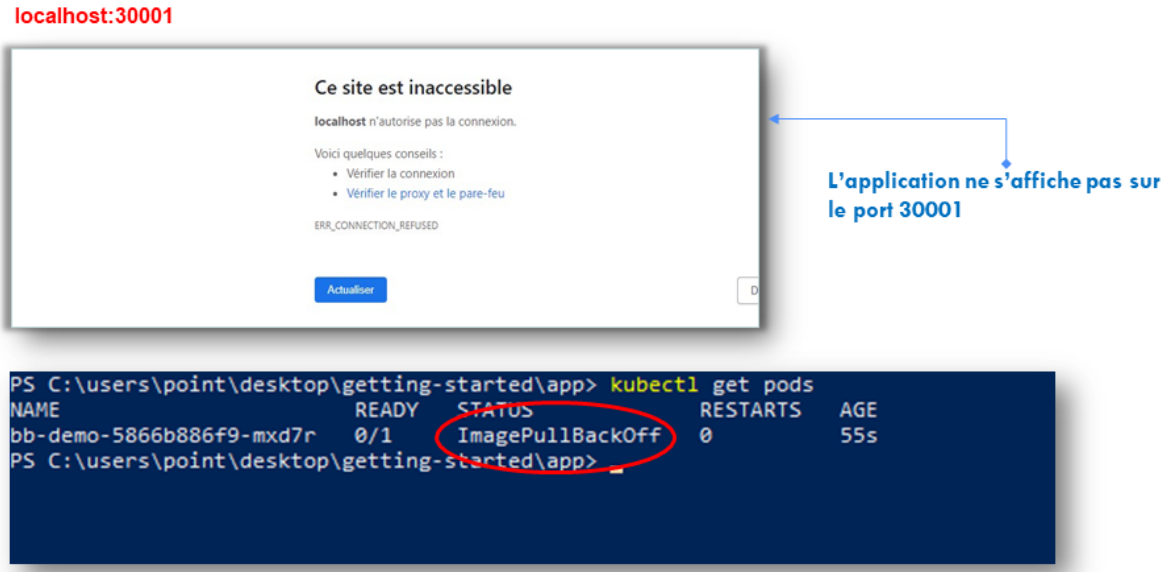


Figure 4.6: Erreur

4. **Solutions**
   Dans le but de savoir si le problème provient de l'application. Nous avons testé ces deux images officielles : la simple image hello world et le serveur web open source Nginx

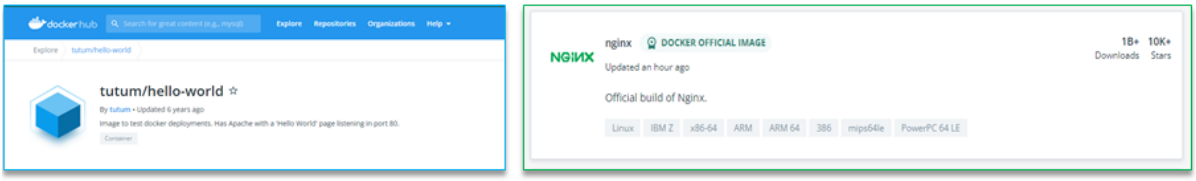
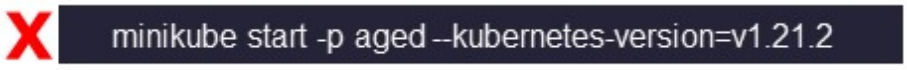


Figure 4.7: Deux images officiel

Nous avons remarqué la persistance du même problème ce qui nous a pousser a chercher plus loin pour en déduire que c'étais un soucis de version et du container rutime.
Pour que les ponds puissent s'exécuter il est nécessaire d'installer un environnement d'exécution de conteneur tel que docker.
Le composant dockershim de Kubernetes permettait d'utiliser Docker comme runtime de conteneur, Mais Dockershim a été supprimer complétement de k8s depuis la version 1.20
Par conséquent, nous avons utilisé deux méthodes pour résoudre cette erreur.

i. **Utiliser le runtime docker**

- **Downgrade kubernetes:** Nous avons essayé d'utiliser une version inférieure à 1.20, Qui inclue dockershim.



- **Docker-cri:** Est la solution pour installer et utiliser dockershim, nous avons donc essayé de l'installer conformément aux instructions de la documentation officielle.

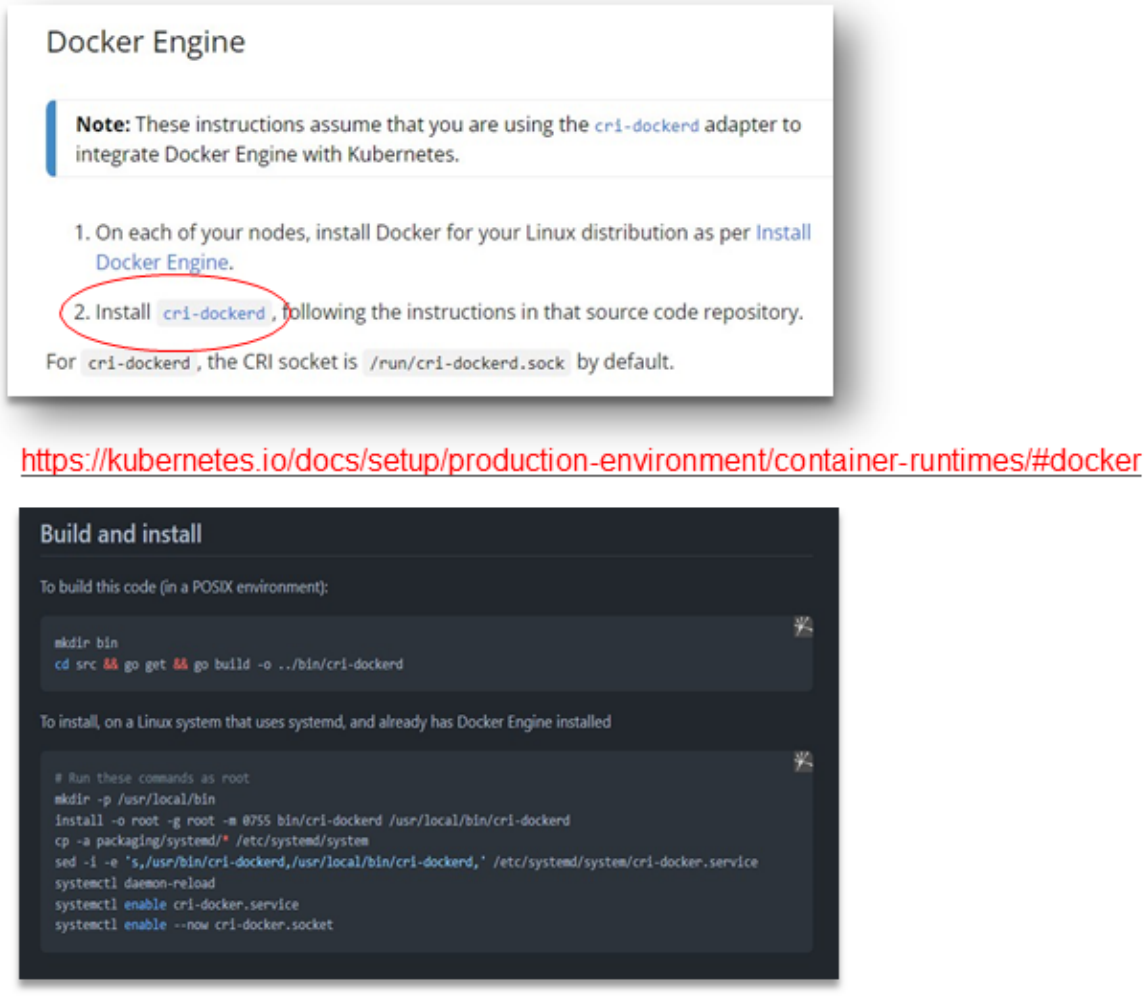


Figure 4.8: Docker Engine

Étant donné que la solution proposée est sous l'environnement Linux, nous avons essayé d'utiliser la distribution Linux sous Windows.

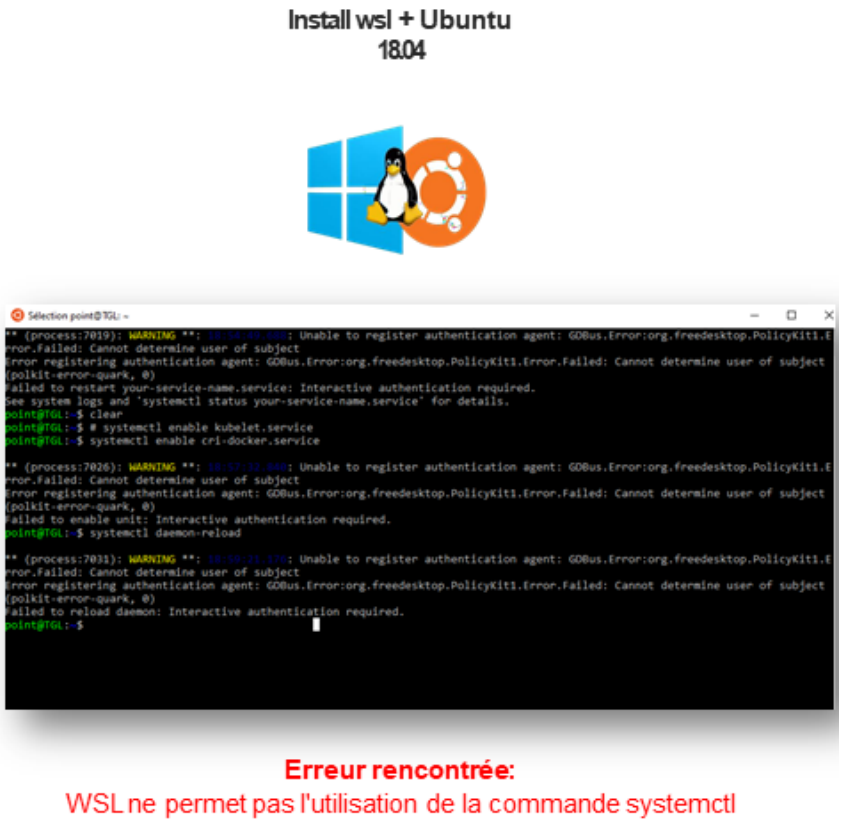


Figure 4.9: Distribution linux sous windows

ii. **Containerd:** Nous avons essayé de modifier le runtime CRI en choisissant containerd qui possède la plupart des fonctionnalités Docker pour l'exécution des conteneurs, la gestion de la mémoire et la gestion des images.

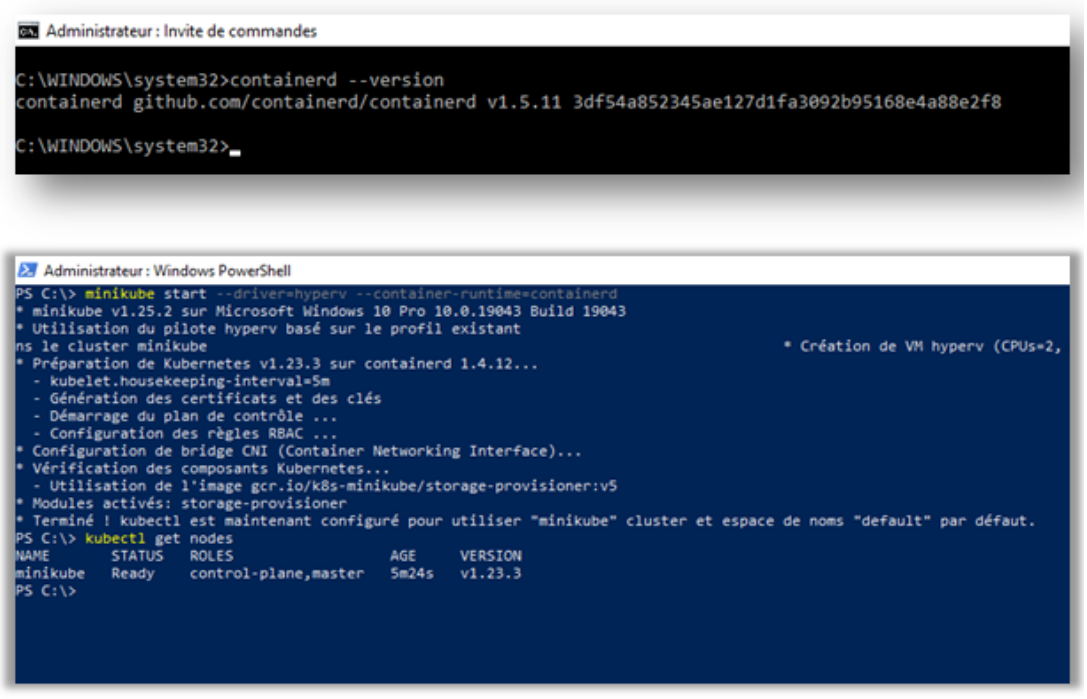
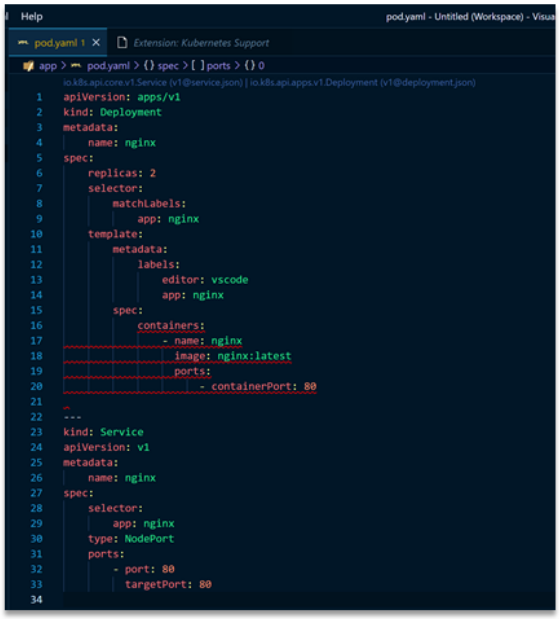


Figure 4.10: CRI containerd

iii. **Continuer le déploiement avec kubernetes**

Revenons aux deux précédentes images Hello world et Nginx que nous avons essayé d'implémenter.

Nous allons créer deux fichiers yaml pour chacune puis nous procédons à leur déploiement.



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
    name: nginx
spec:
    replicas: 2
    selector:
        matchLabels:
            app: nginx
    template:
        metadata:
            labels:
                editor: vscode
                app: nginx
        spec:
            containers:
                - name: nginx
                  image: nginx:latest
                  ports:
                      - containerPort: 80
---
kind: Service
apiVersion: v1
metadata:
    name: nginx
spec:
    selector:
        app: nginx
    type: NodePort
    ports:
        - port: 80
          targetPort: 80
```

Pod.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
    name: tutum-deployment
spec:
    replicas: 1
    selector:
        matchLabels:
            app: tutum
    template:
        metadata:
            labels:
                editor: vscode
                app: tutum
        spec:
            containers:
                - name: tutum
                  image: tutum/hello-world:latest
                  ports:
                      - containerPort: 80
---
kind: Service
apiVersion: v1
metadata:
    name: tutum-service
spec:
    selector:
        app: tutum
    type: NodePort
    ports:
        - port: 5500
          targetPort: 80
```

Simple.yaml

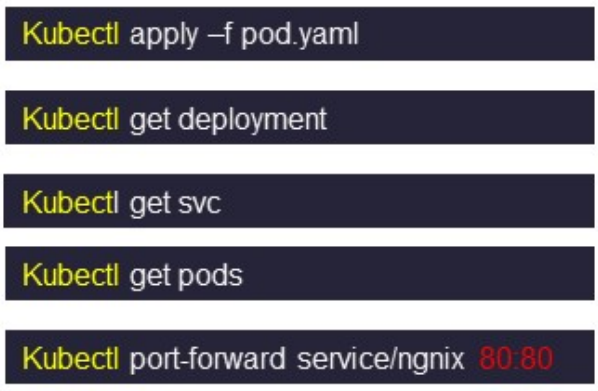
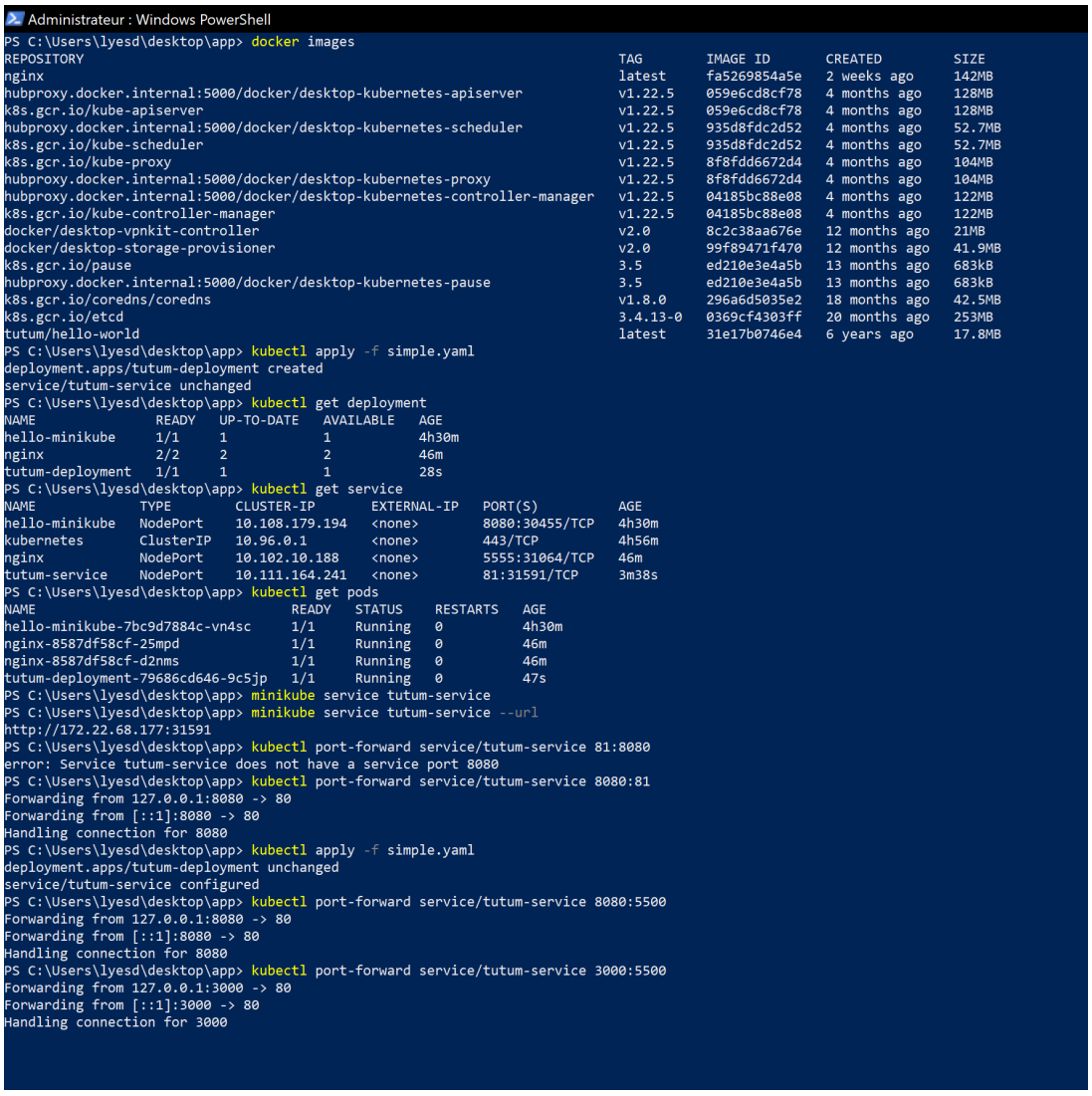
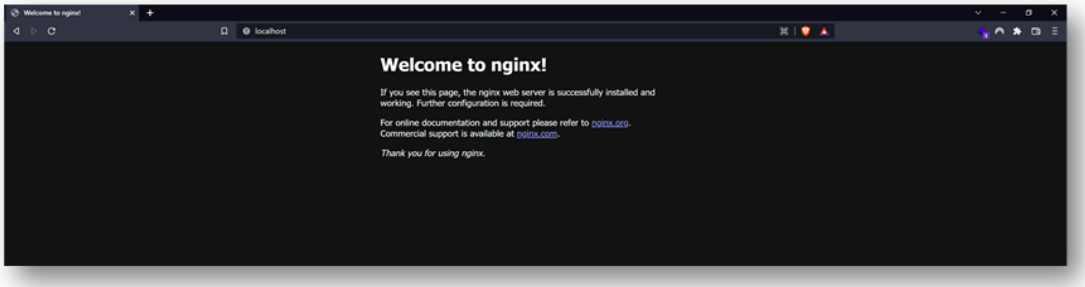


Figure 4.11: Fichiers yaml des deux images et lancement du déploiement





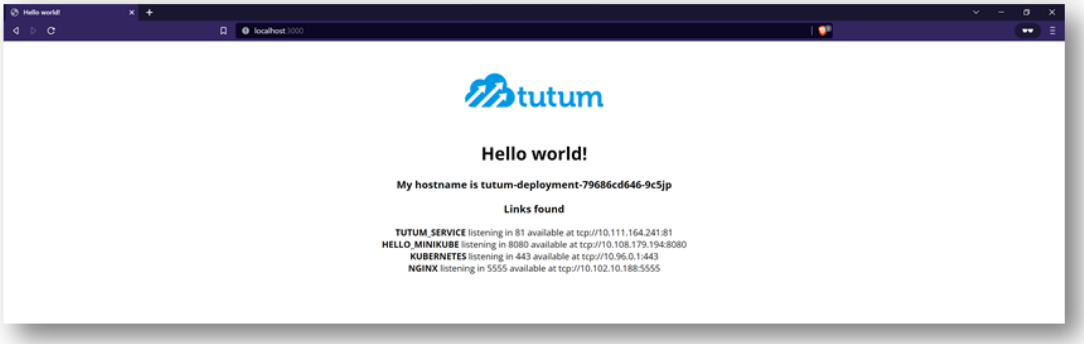
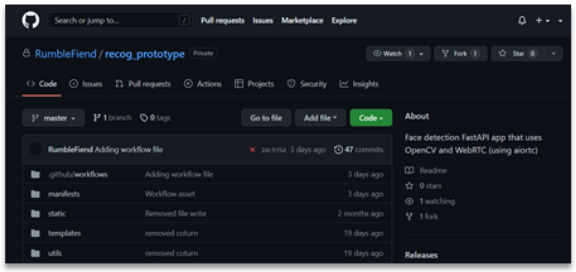
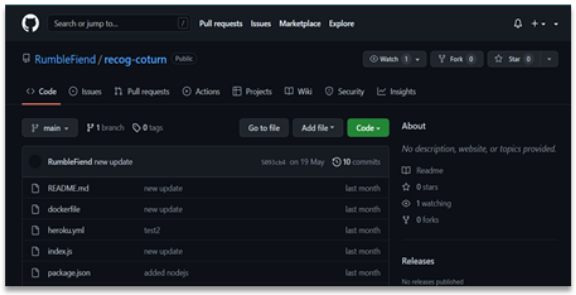


Figure 4.12: Resultat

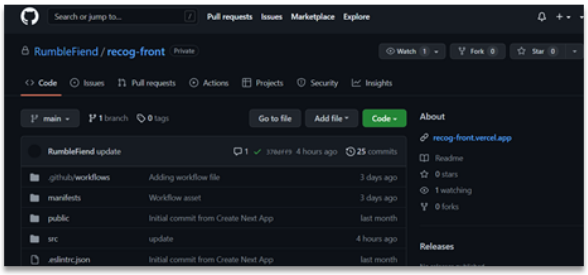
5. **Déploiement du système avec Minikube:** La solution cloud computing Intelligent pour la reconnaissance en temps Réel dispose d'une architecture microservices composée de trois parties : Frontend, Coturn et Backend ML.
   Nous mettrons donc en place trois services.



Frontend



Coturn



BackendML

[29] [30] [31]

Figure 4.13: les repos git

- **Déploiement du front-end:** Nous allons passer par les mêmes étapes précédentes, c'est-à-dire que nous allons construire notre image puis la taguer pour l'inclure dans le registre docker, puis nous configurons le fichier yaml et déployons l'image du frontend.

Tag and push notre image sur docker hub dans le repo « po001int »

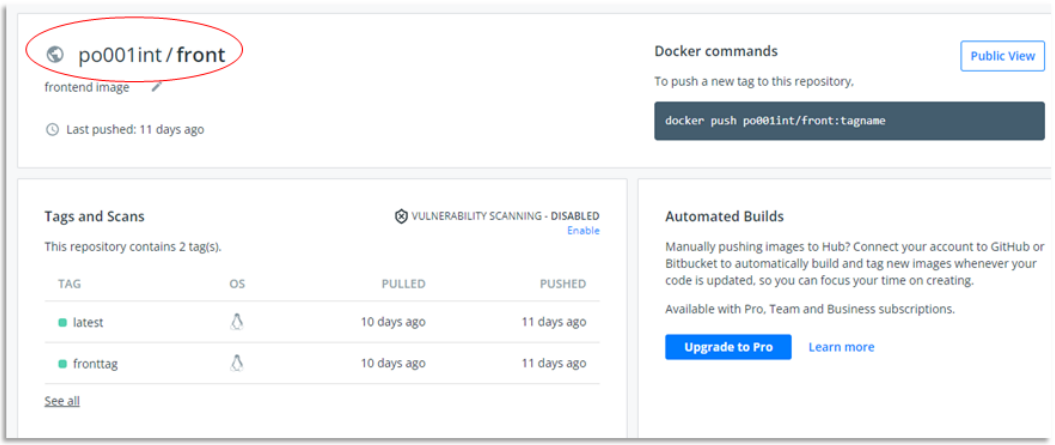


Figure 4.14: Image de l'application

Avec yaml manuellement

```
Administrateur : Windows PowerShell
PS C:\users\point\desktop\front-recog> kubectl apply -f front.yaml
deployment.apps/front unchanged
service/front unchanged
PS C:\users\point\desktop\front-recog> kubectl get deployment
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
front   1/1     1            1           7m40s
PS C:\users\point\desktop\front-recog> kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
front        NodePort    10.111.234.19   <none>        5555:32702/TCP   7m43s
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP          3h2m
PS C:\users\point\desktop\front-recog> kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
front-76ff5dc9c6-ws1sn   1/1     Running   0          7m47s
PS C:\users\point\desktop\front-recog> kubectl port-forward service/front 3000:5555
Forwarding from 127.0.0.1:3000 -> 3000
Forwarding from [::1]:3000 -> 3000
```

Avec CLI

```
PS C:\users\point\desktop\front-recog> kubectl create deployment front --image=po001int/front
deployment.apps/front created
PS C:\users\point\desktop\front-recog> kubectl expose deployment front --type=NodePort --port=3000
service/front exposed
PS C:\users\point\desktop\front-recog> kubectl get deployment
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
front   1/1     1            1           53s
PS C:\users\point\desktop\front-recog> kubectl get svc
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
front        NodePort    10.103.2.26    <none>        3000:32030/TCP   15s
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP          3h6m
PS C:\users\point\desktop\front-recog> kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
front-88574f45f-176nz    1/1     Running   0          64s
PS C:\users\point\desktop\front-recog> kubectl port-forward service/front 3000:5555
error: Service front does not have a service port 5555
PS C:\users\point\desktop\front-recog> kubectl port-forward service/front 5555:3000
Forwarding from 127.0.0.1:5555 -> 3000
Forwarding from [::1]:5555 -> 3000
Handling connection for 5555
Handling connection for 5555
Handling connection for 5555
PS C:\users\point\desktop\front-recog>
```

Résultat

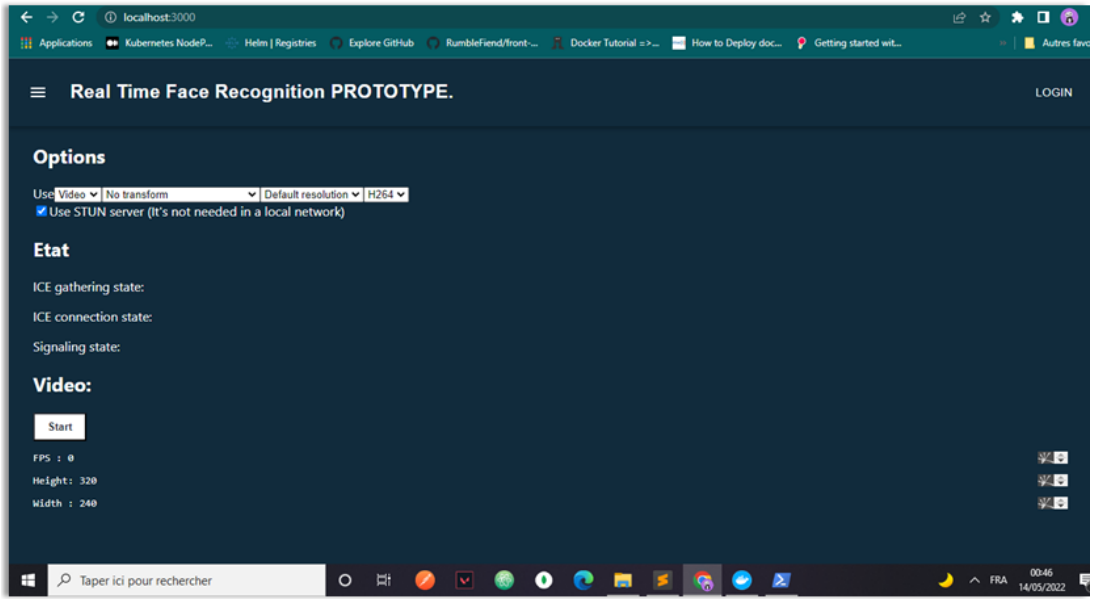
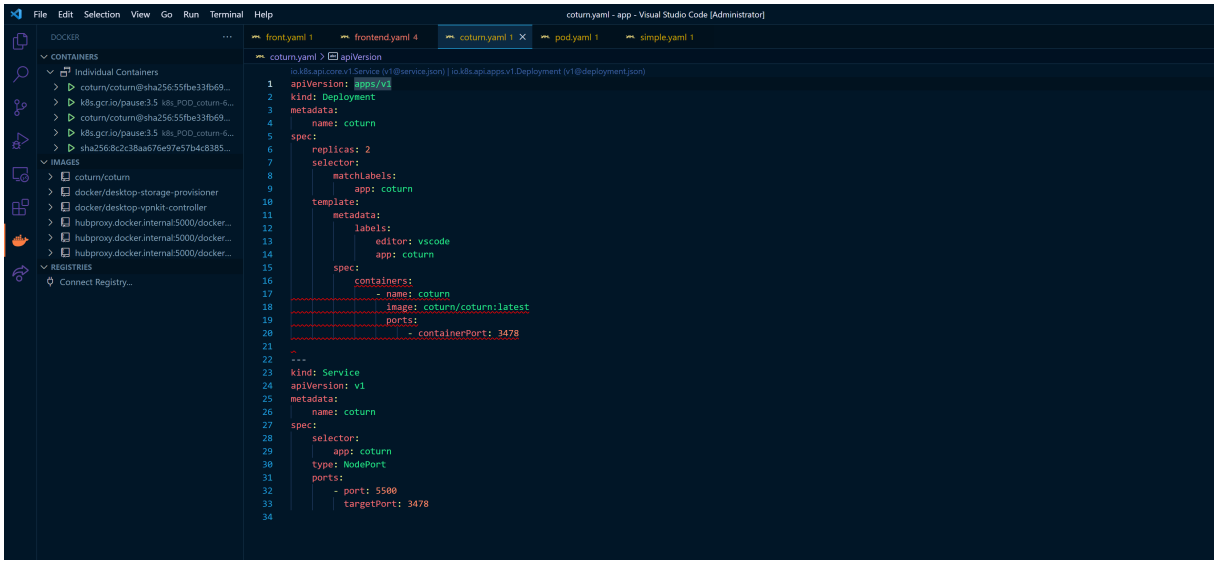


Figure 4.15: Image de l'application

- **Déploiement du Coturn**



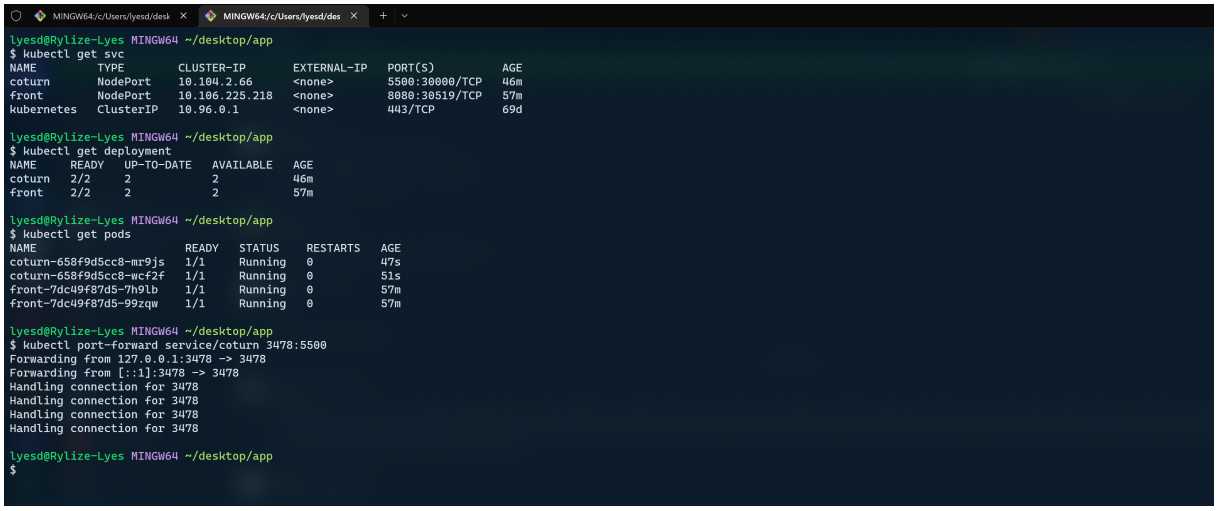


Figure 4.16: Coturn

6. **Déploiement du BackendML**
   c'est là que nous utiliserons kubeflow, pour cela nous utiliserons MicroK8 qui est un K8 léger et concentré.
   Avant de commencer, nous devons considérer les prérequis suivants : Ubuntu 20.04 ou version ultérieure, au moins 16Go de mémoire libre et 20Go d'espace disque.

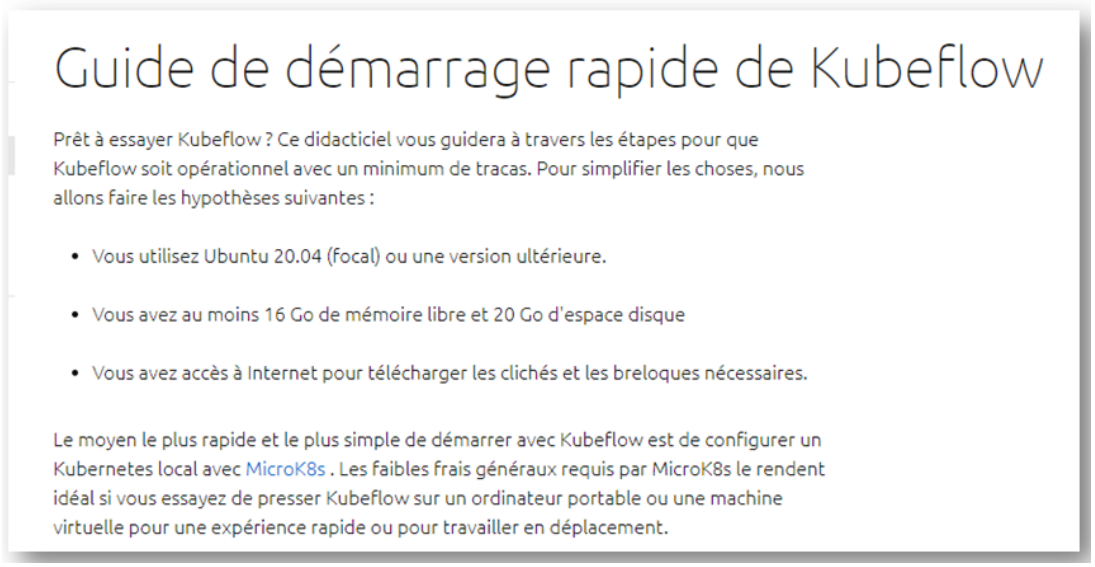


Figure 4.17: Kubeflow prérequis.

Pour cela, nous utiliserons une session d'un vps (virtual private server) qu'on a pu se procurer et qui est doté d'une capacité conforme aux exigences.

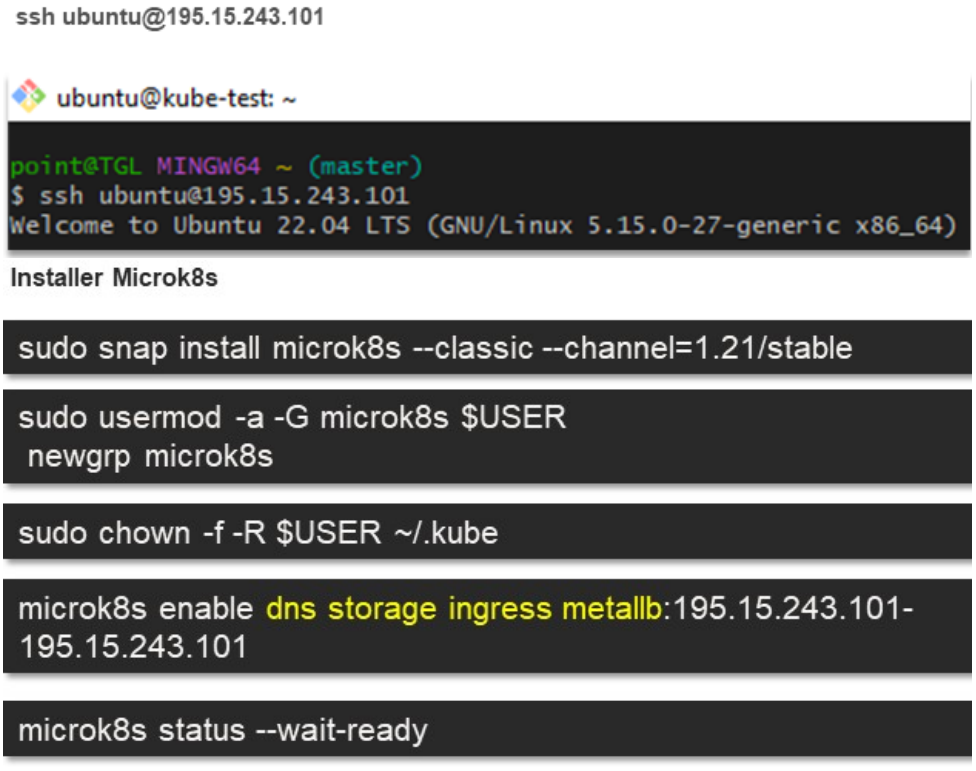
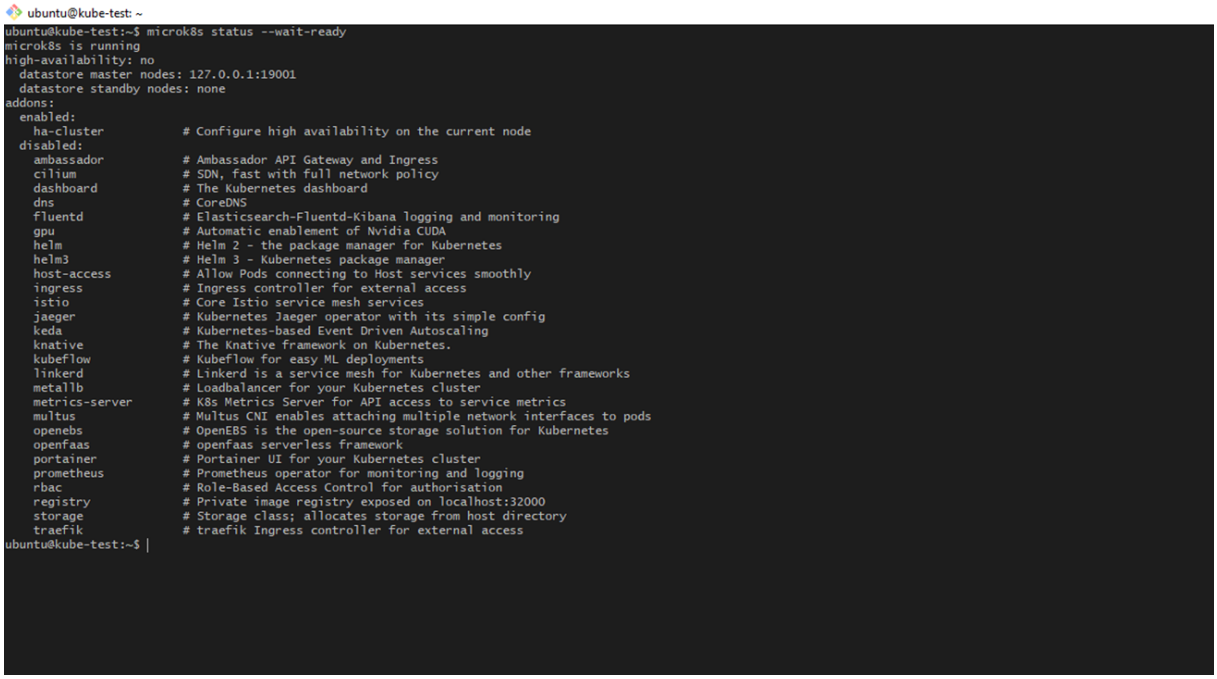


Figure 4.18: vps et microk8s



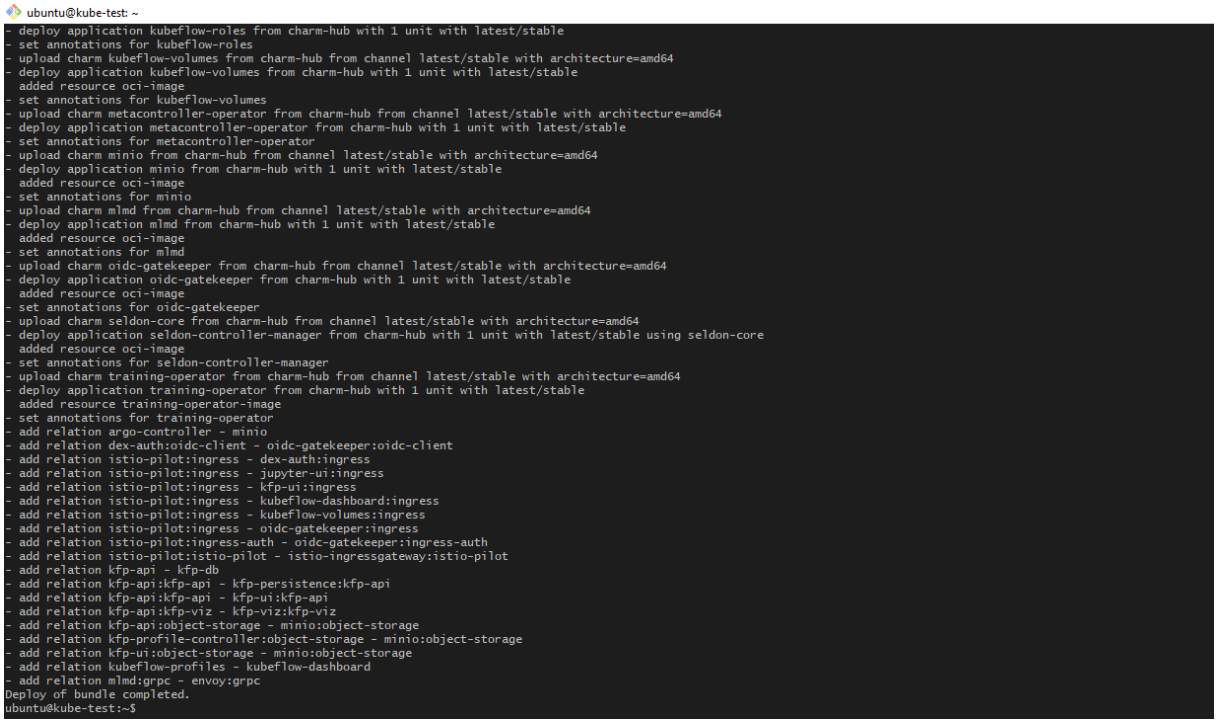


Figure 4.19: Installation de microk8s

Installer Juju

Utiliser pour déployer et gérer les composants qui composent Kubeflow.

```
sudo snap install juju --classic
juju bootstrap microk8s
juju add-model kubeflow
```

Déployer kubeflow

```
juju deploy kubeflow-lite --trust
watch -c juju status --color
```

Configuration

```
juju config dex-auth static-username=admin
juju config dex-auth static-password=admin
```

Access to Dashboard

```
exit
ssh -D 9999 ubuntu@195.15.243.101
```

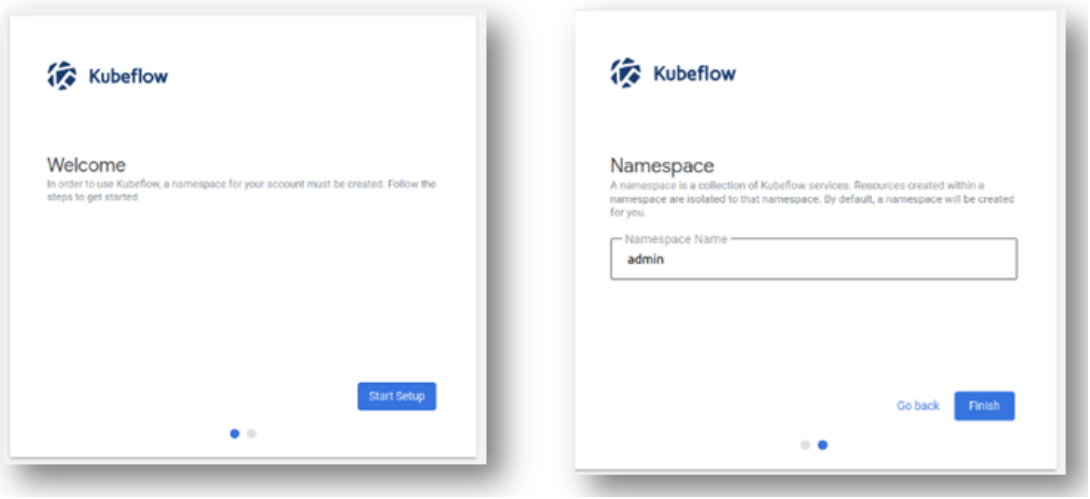


Figure 4.20: Déploiement de kubeflow

http://1195.15.243.101.nip.io

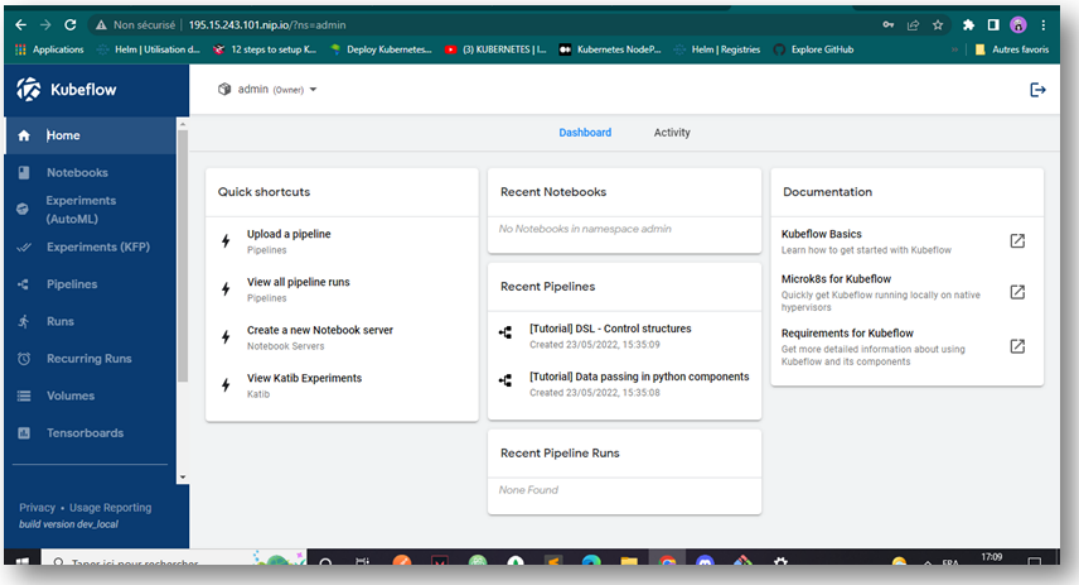


Figure 4.21: Interface kubeflow

4.3 Déploiement avec trois nœuds physiques

Pour créer un cluster k8s multi-nœuds, nous utiliserons kubeadm. Dans notre cas,nous allons opérer sur trois nœuds: Master node, Worker1 et Worker2.

Installer et configurer kubeadm

```
sudo apt update
```

```
sudo sysctl --system
```

```
sudo apt install -y docker docker.io containerd
```

```
sudo usermod -aG docker $USER
```

```
sudo newgrp docker
```

```
echo "configuring docker to use systemd«
cat << EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"]
} EOF
```

```
" Restart docker "
sudo systemctl daemon-reload
sudo systemctl restart docker
```

```
" Restart containerd "
sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl enable docker
```

```
" Installing kubernetes debendencies "
sudo apt install -y apt-transport-https curl
curl -s
https://packages.cloud.google.com/apt/doc/apt-
key.gpg | sudo apt-key add -
```

```
echo "adding apt repository for kubernetes ..."
cat << EOF | sudo tee
/etc/apt/sources.list.d/kubernetes.list deb
https://apt.kubernetes.io/ kubernetes-xenial
main EOF-y apt-transport-https curl
```

```
sudo apt update
```

Figure 4.22: Installation et configuration de Kubeadm

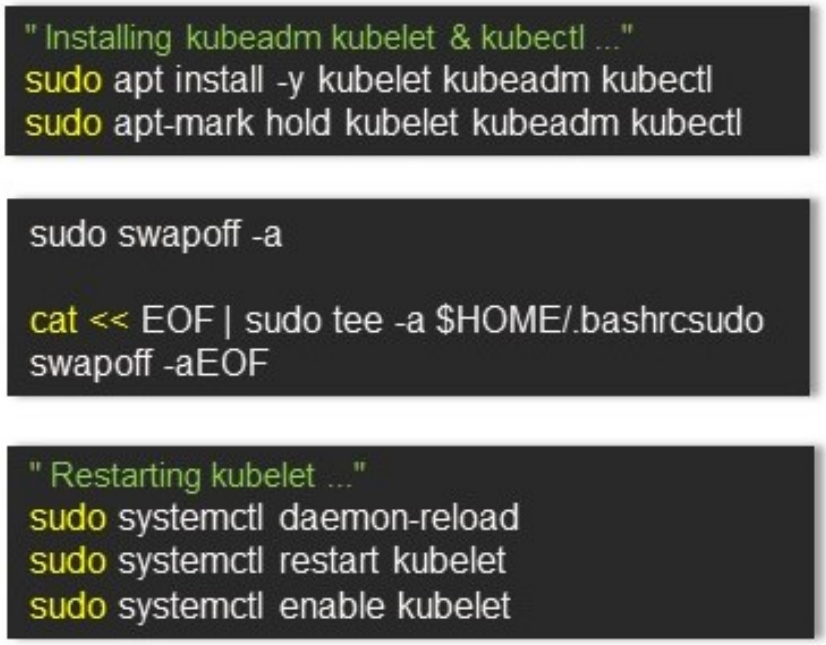


Figure 4.23: Installation et configuration de Kubeadm

Nous allons initialiser un cluster sur le master par la suite nous joindrons les deux workers.

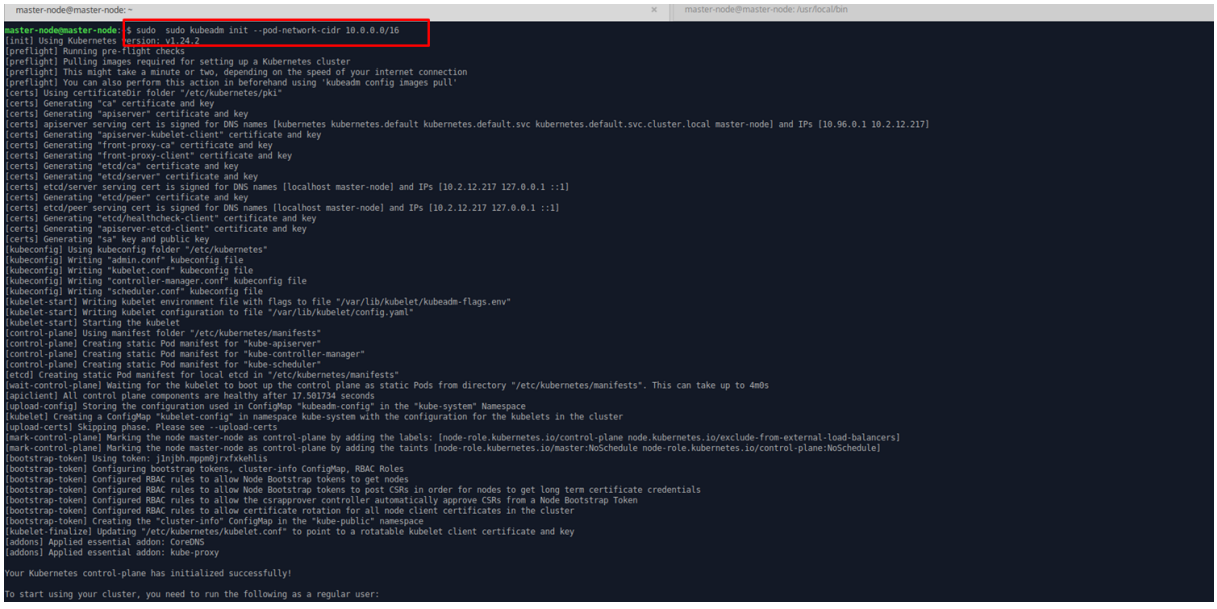


Figure 4.24: Kubeadm

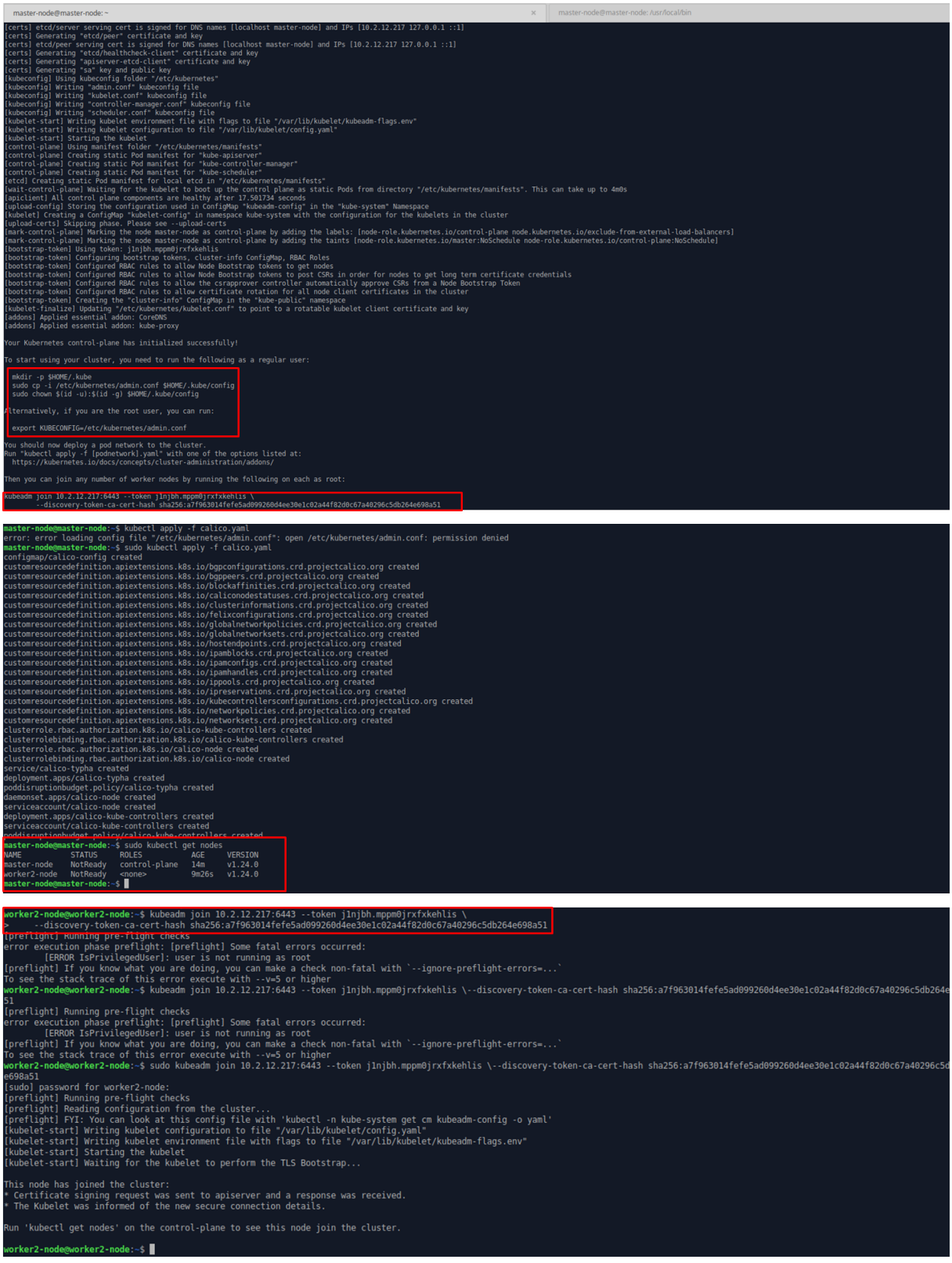


Figure 4.25: Kubeadm

Nous aurons besoin de Helm, le gestionnaire de packets pour Kubernetes dans le but d'ajouter des packets nécessaires. dans notre cas nous allons déployer avec le dahboard k8s.

1- Installation de Helm

```
choco install kubernetes-helm
```

2- Initialiser un dépôt de charts Helm

```
helm repo add stable https://charts.helm.sh/stable
```

3- Installer un package pour tester

```
helm install happy-panda bitnami/wordpress
```

les dépôts publiques de charts Helm sont dans ArtifactHUB

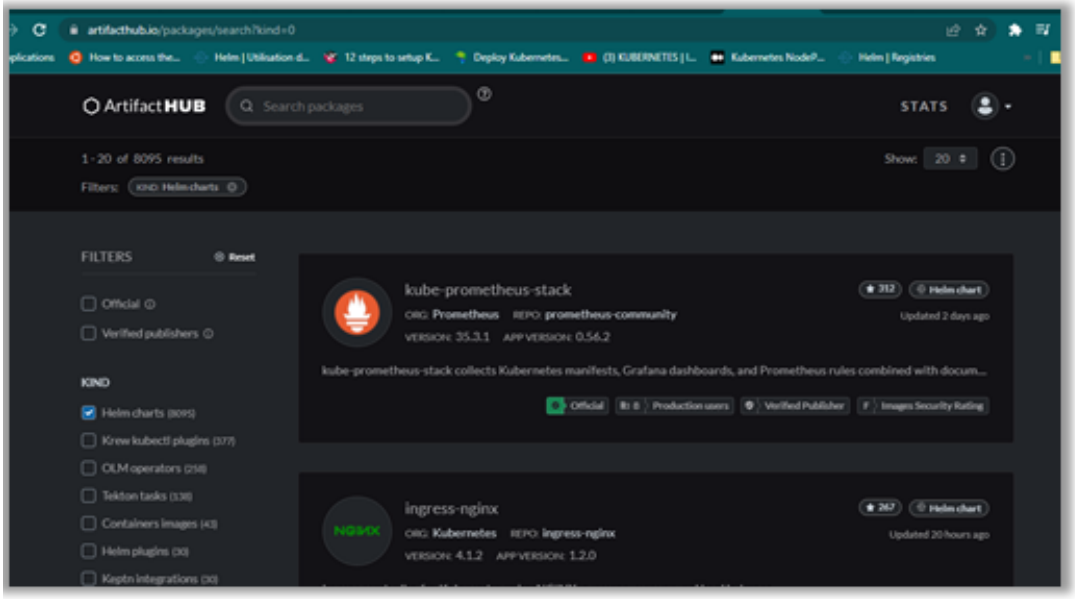


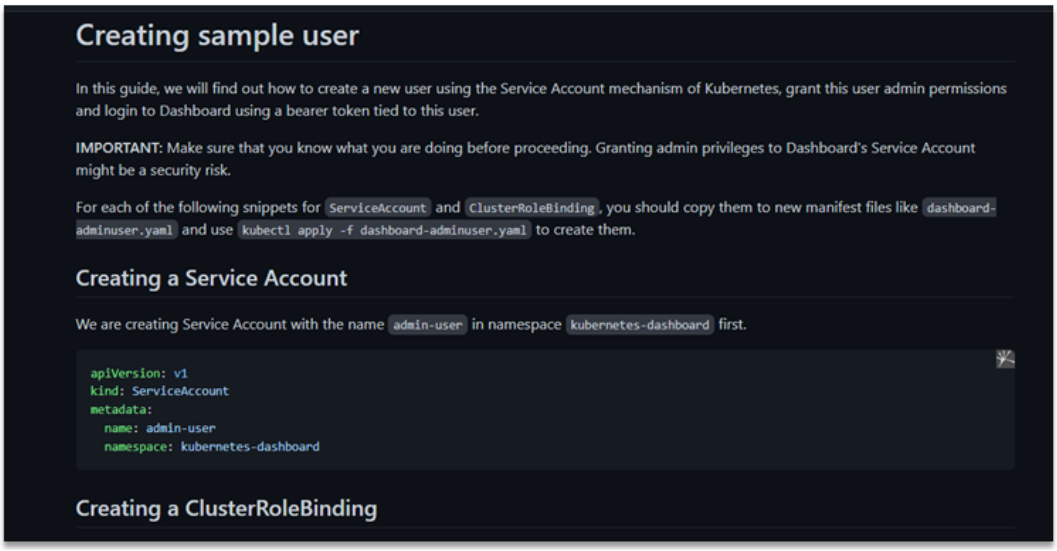
Figure 4.26: Helm


```
# Add kubernetes-dashboard repository
helm repo add kubernetes-dashboard https://kubernetes.github.io/dashboard/
# Deploy a Helm Release named "kubernetes-dashboard" using the kubernetes-dashboard chart
helm install kubernetes-dashboard kubernetes-dashboard/kubernetes-dashboard
```

```
helm repo add kubernetes-dashboard https://kubernetes.github.io/dashboard/
helm install kubernetes-dashboard kubernetes-dashboard/kubernetes-dashboard
```

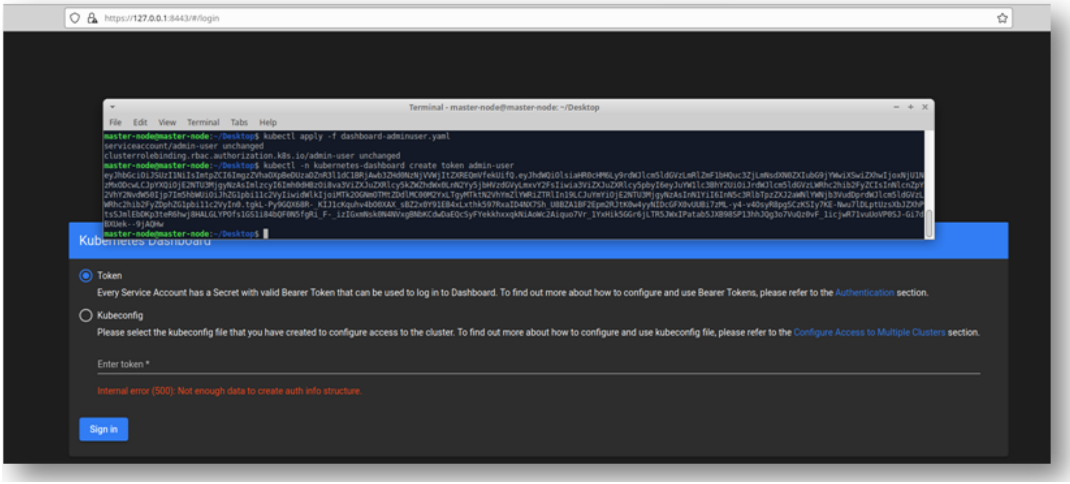

Figure 4.27: Installer Helm

Crée un utilisateur



Trouver le jeton que nous pouvons utiliser pour nous connecter.

```
kubectl -n kubernetes-dashboard create token admin-user
```



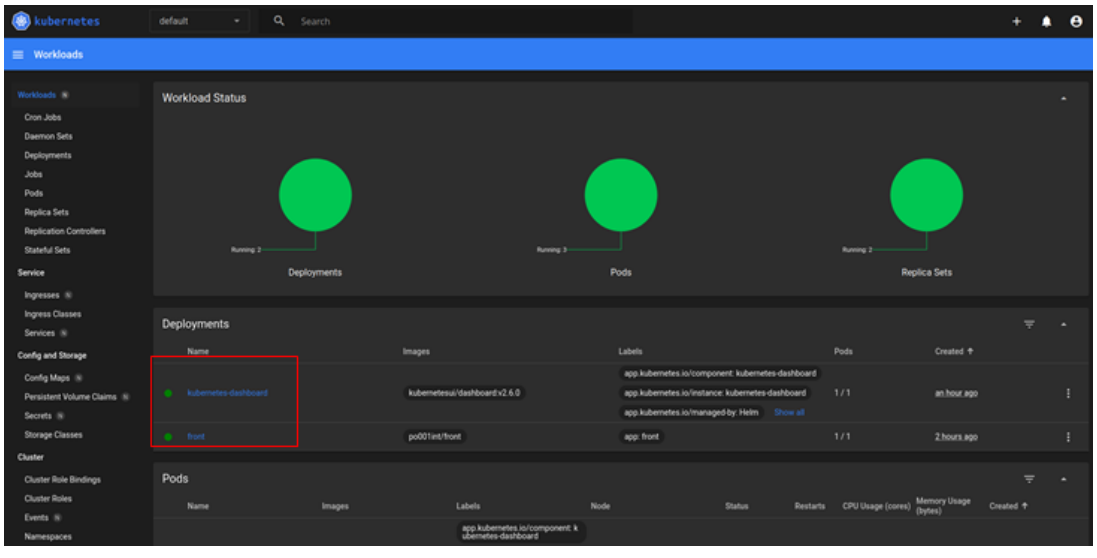


Figure 4.28: Déployer le dashboard kubernetes

Déployer le frontend

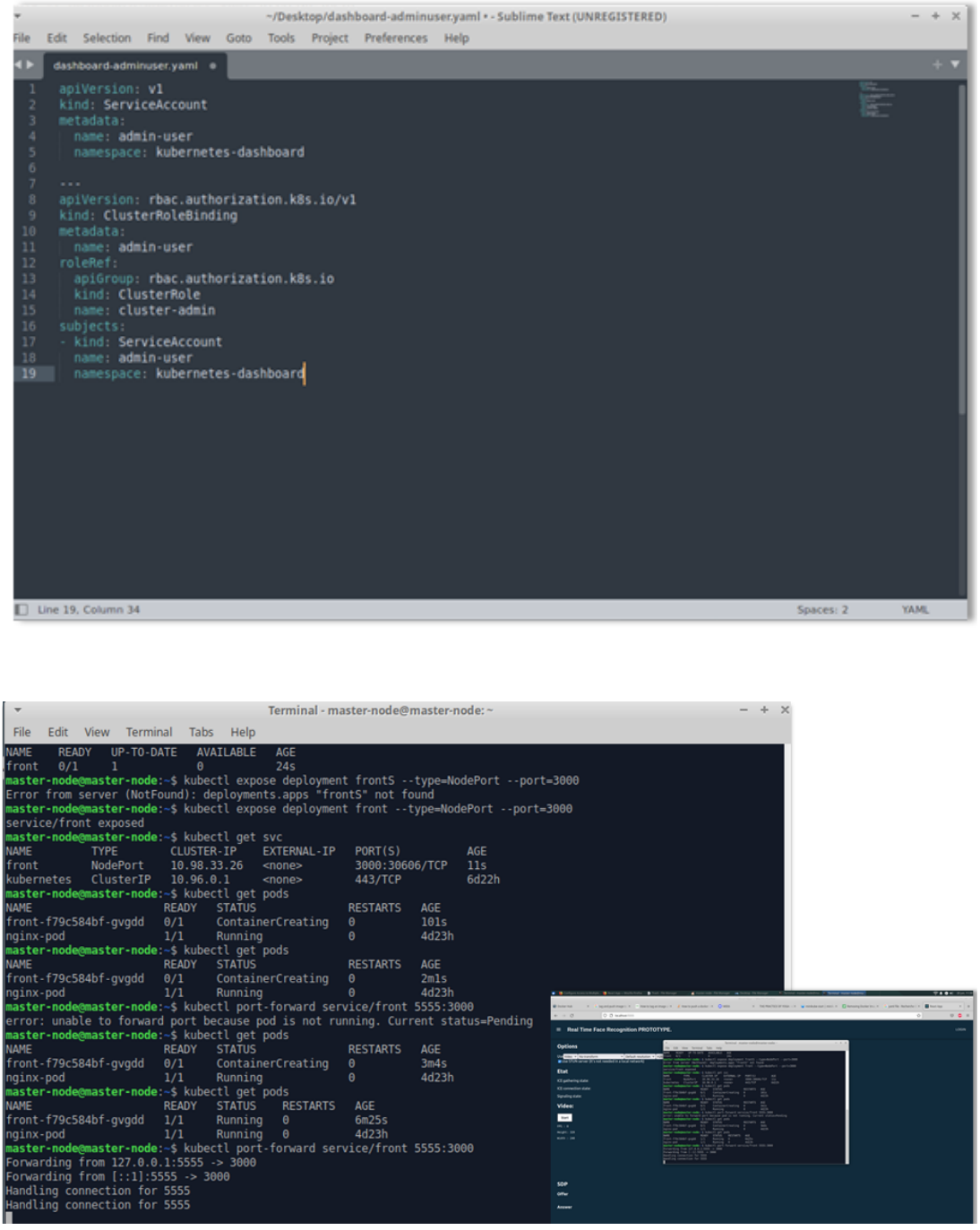


Figure 4.29: Déployer le dashboard kubernetes

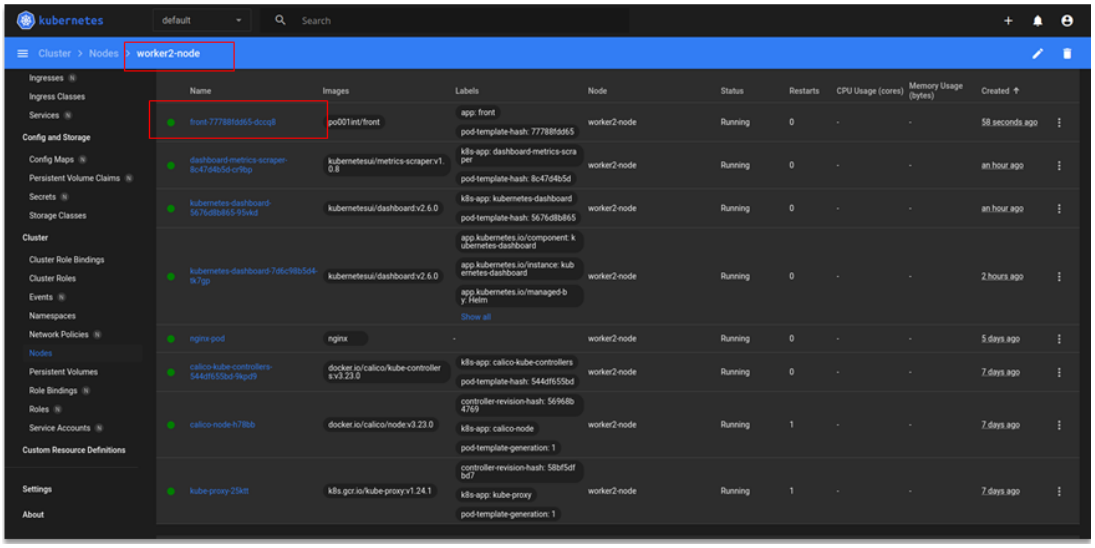


Figure 4.30: Dashboard kubernetes

4.4 Conclusion

Dans ce chapitre, nous avons pu implémenter les trois microservices qui composent notre système : Frontend, coturn et BackendML, en utilisant minikube, kubeadm et kubeflow qui proposent des services différents afin de réaliser un déploiement local.

Chapter 5

Déploiement cloud

5.4 Conclusion

Dans ce chapitre, nous avons montré comment utiliser et configurer des infrastructures cloud, notamment AWS et Azure, pour créer des clusters, des rôles, des groupes de ressources, connecter des clusters à une machine et déployer différents systèmes (images simples, applications, modèles machine learning).

5.1 Introduction

Dans cette section, nous allons démontrer les différentes étapes de déploiement cloud que nous avons prises sur deux fournisseurs de services cloud : Amazon web services et Microsoft Azure.

5.2 Avec amazon web services (AWS)

AWS est le principal fournisseur de cloud sur le marché, offrant plus de 200 services complets et fournissant une large gamme d'opérations à la demande telles que la puissance de calcul, le stockage de base de données, la diffusion de contenu, etc. Pour aider les entreprises à se développer et à prospérer.[32]
Aws nous propose deux façons de démarrer, d'exécuter et de mettre à l'échelle Kubernetes:

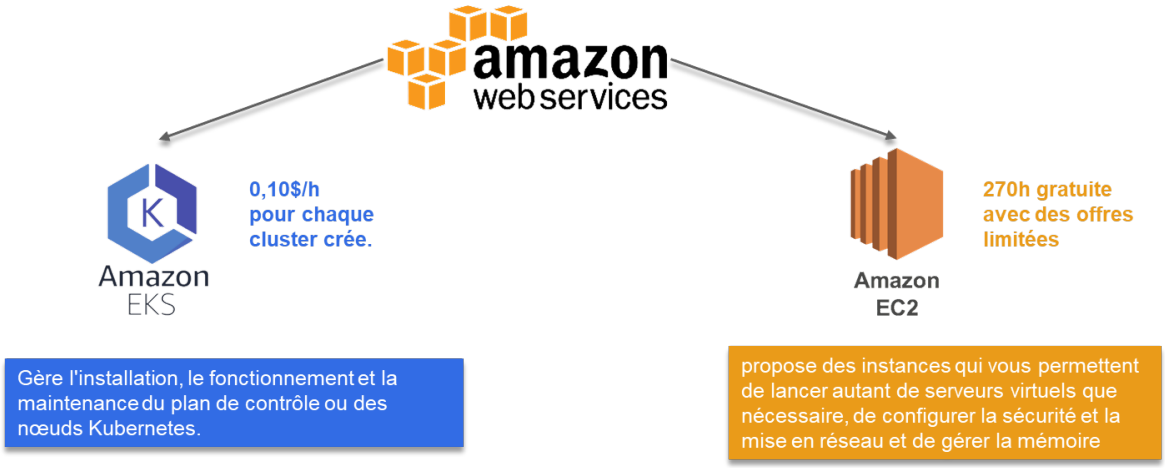


Figure 5.1: AWS Kubernetes

1. **EKS:** vous permet d'exécuter Kubernetes sans avoir à installer, utiliser et maintenir votre propre plan de contrôle ou nœuds Kubernetes.

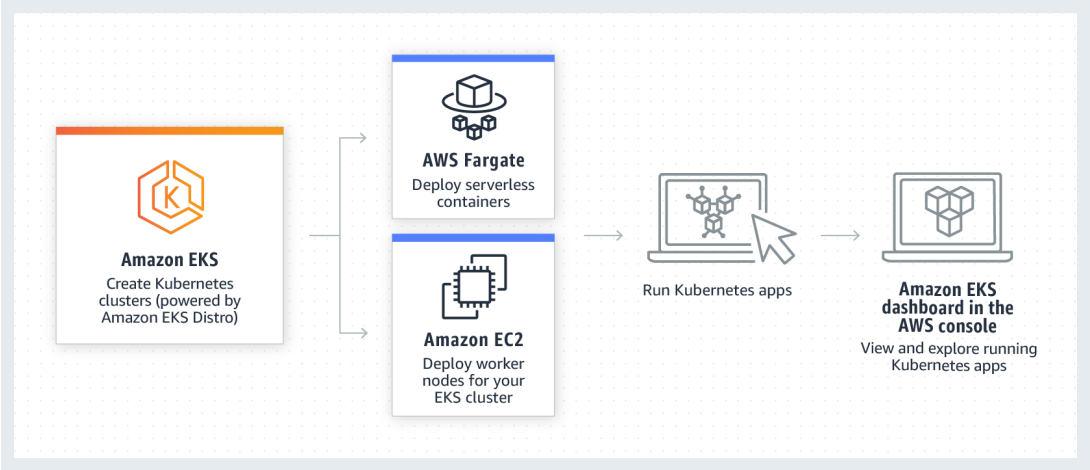


Figure 5.2: Amazon Eks

- **Creation d'un cluster Eks:** Pour créer un cluster Eks, vous devez d'abord créer un VPC (Amazon Virtual Private Cloud) qui vous permet de lancer des ressources AWS dans un réseau virtuel que vous définissez. Ensuite vous crée un rôle IAM qui sera utilisé pour accéder au cluster.

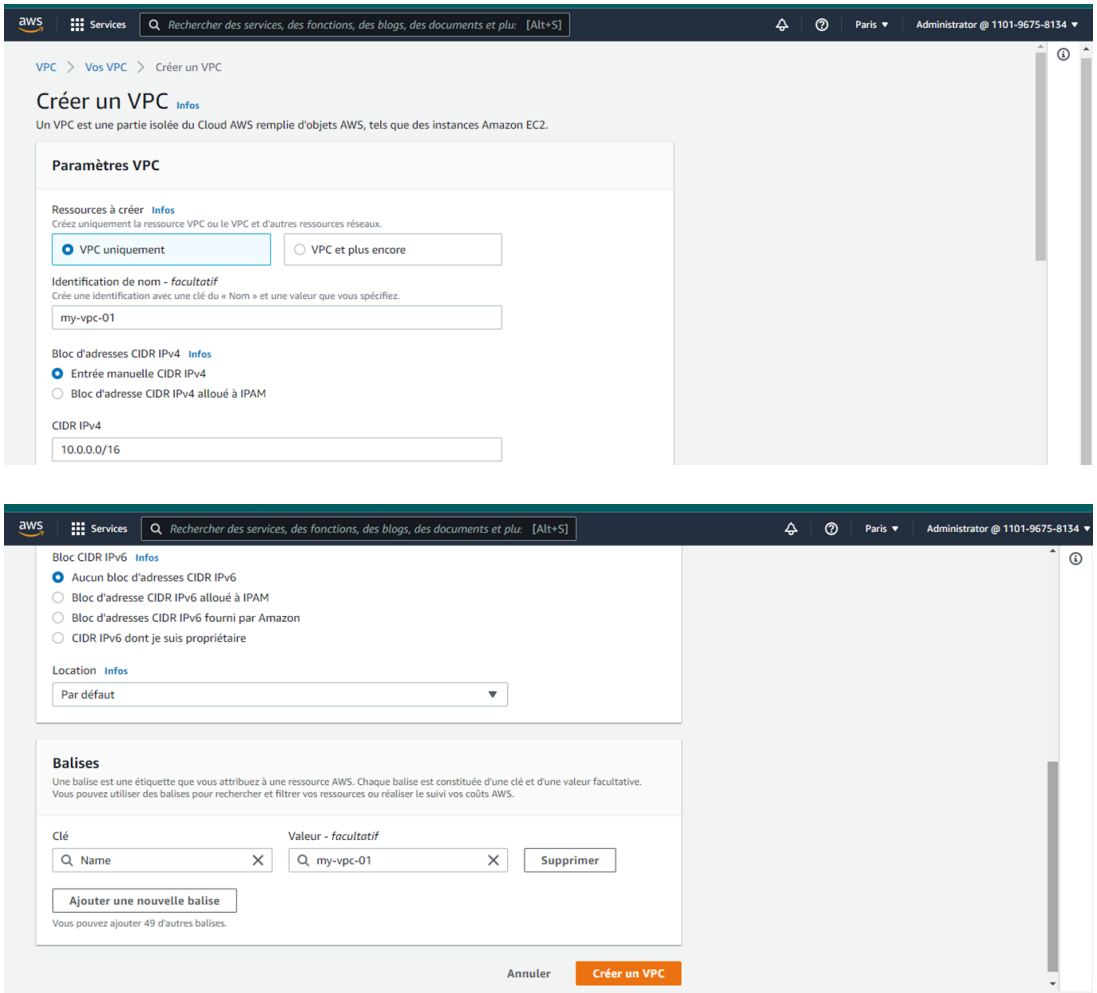


Figure 5.3: Crée un vpc

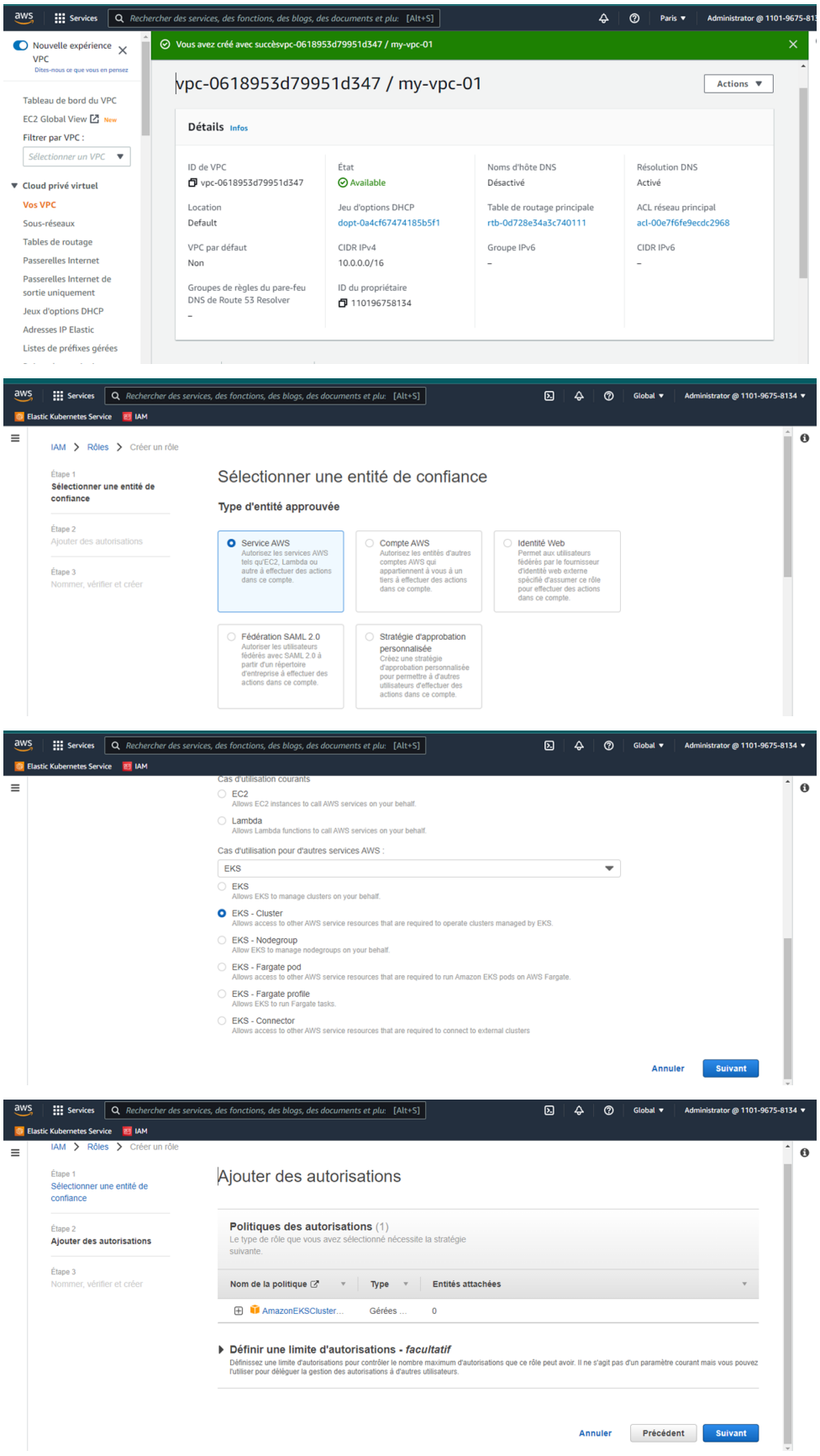


Figure 5.4: Crée un vpc

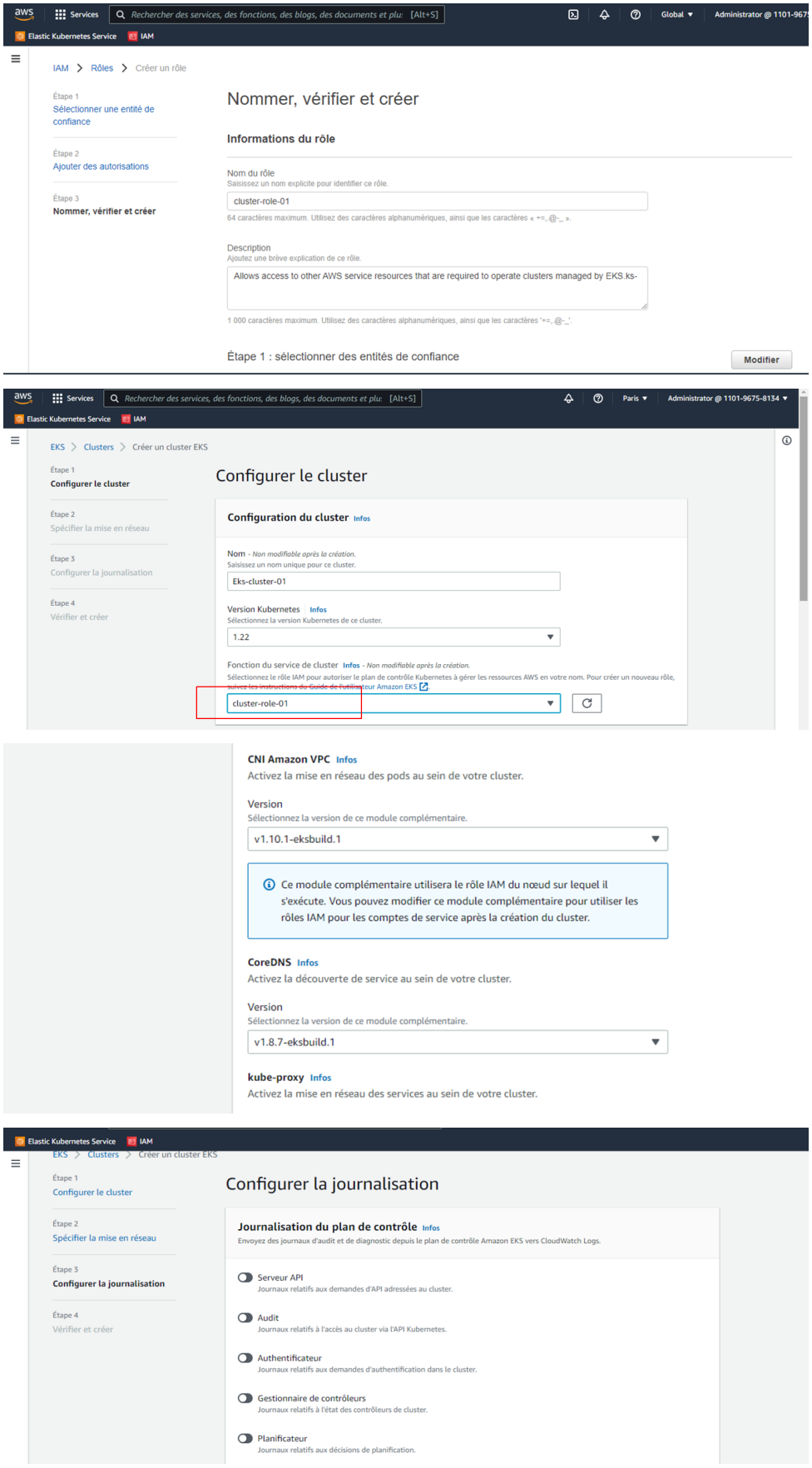


Figure 5.5: Créer un cluster

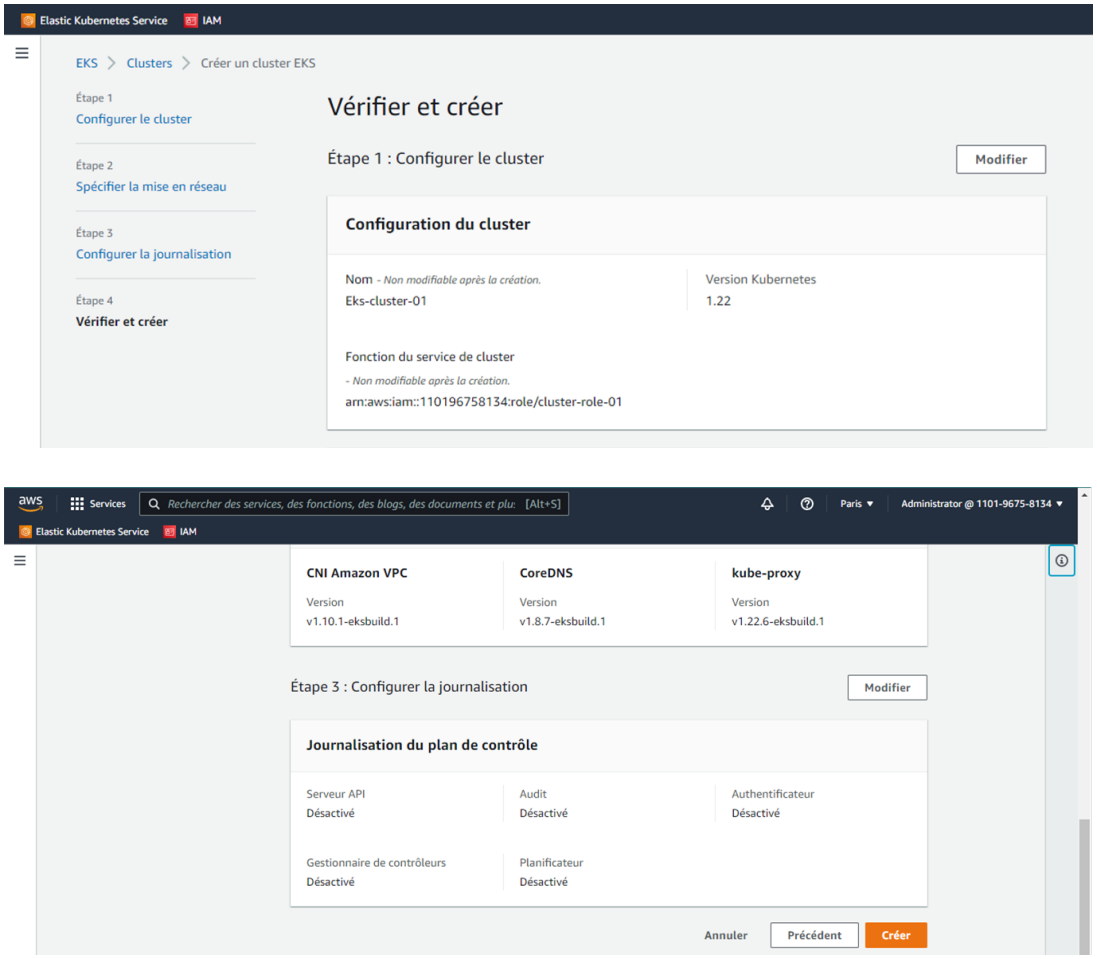


Figure 5.6: Créer un cluster

2. **EC2:** Vous permet de Gérer entièrement votre déploiement.on fournissant des instances de calcul qui hébergent vos nœuds Kubernetes.

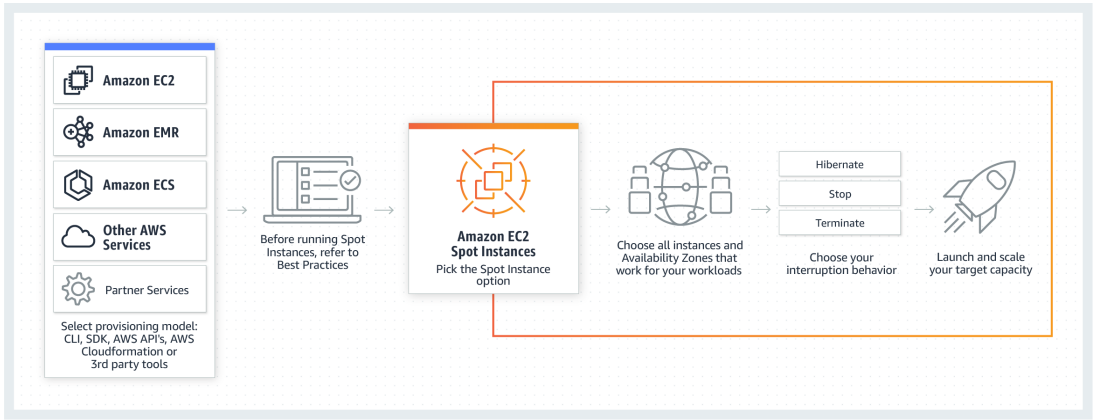


Figure 5.7: Amazon EC2

- **Création d'un utilisateur IAM:** Afin de réaliser toute action sur AWS, il est obligé de s'identifier auprès d'IAM qui est un service de gestion des identités et des accès, il gère les autorisations et les interdictions des actions selon les droits accordés par l'administrateur du compte.

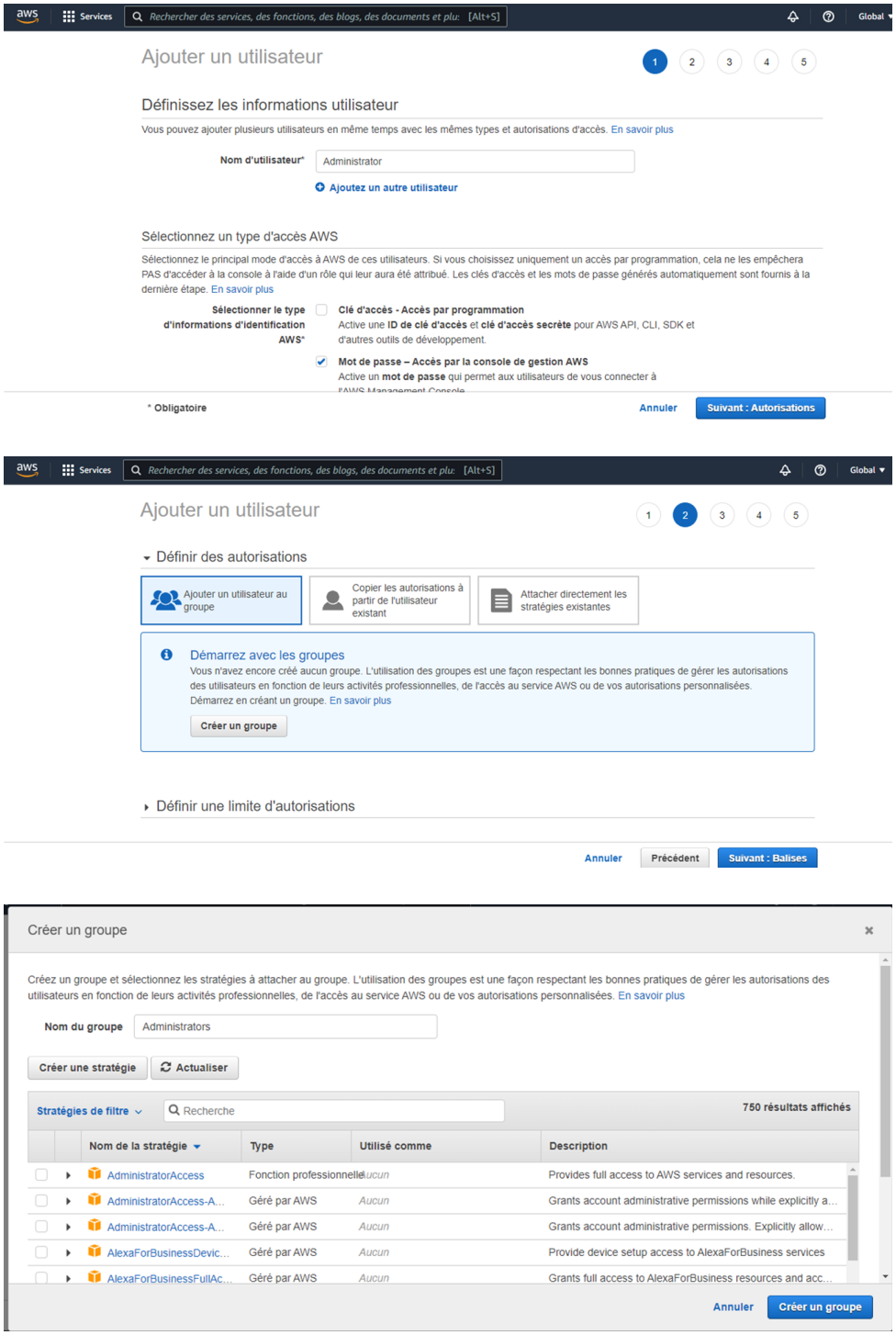


Figure 5.8: IAM

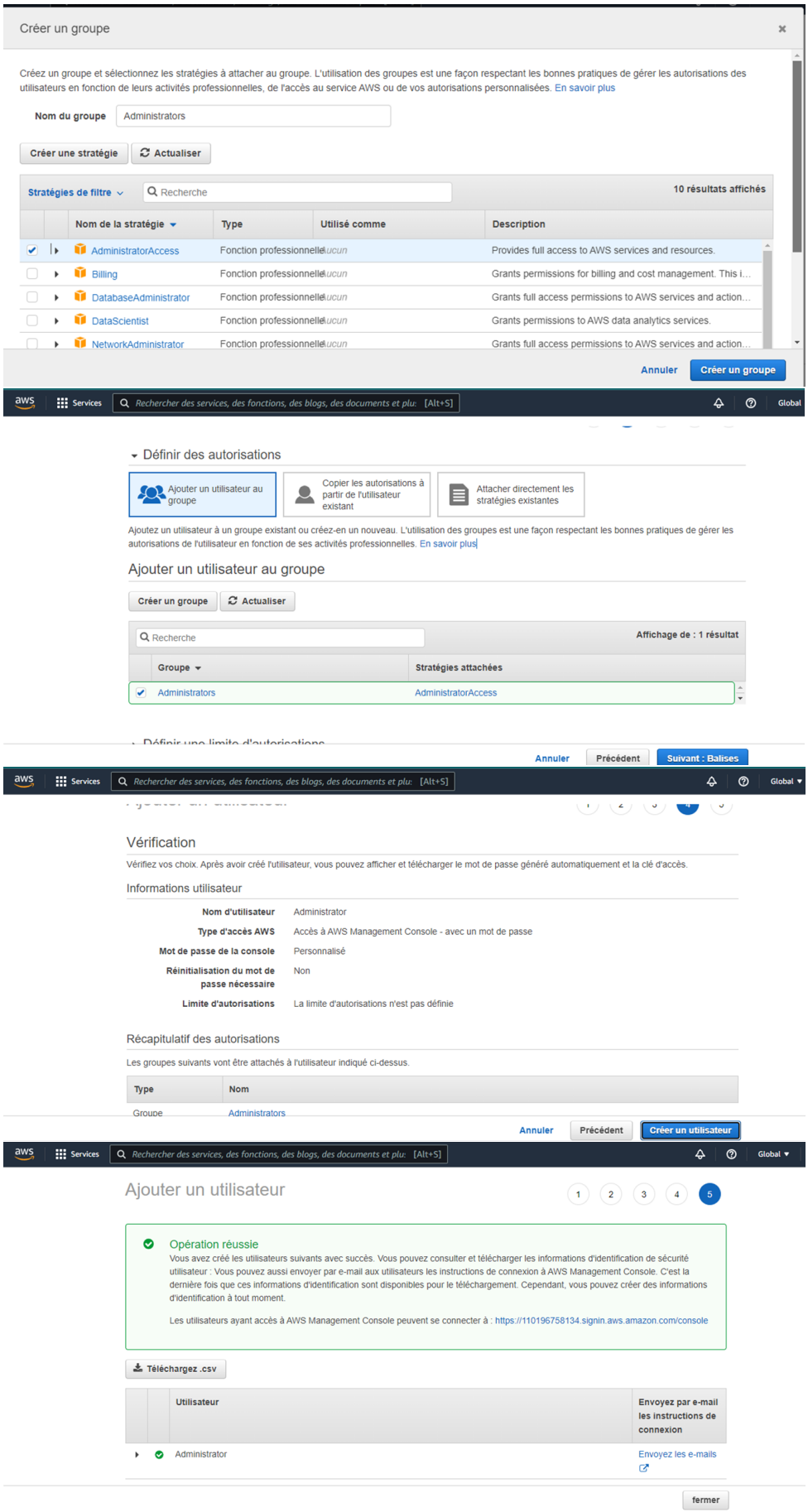


Figure 5.9: IAM

- **Création d'instances EC2:** Une instance EC2 est un serveur virtuel hébergé dans Elastic Compute Cloud (EC2) pour exécuter des applications sur AWS.

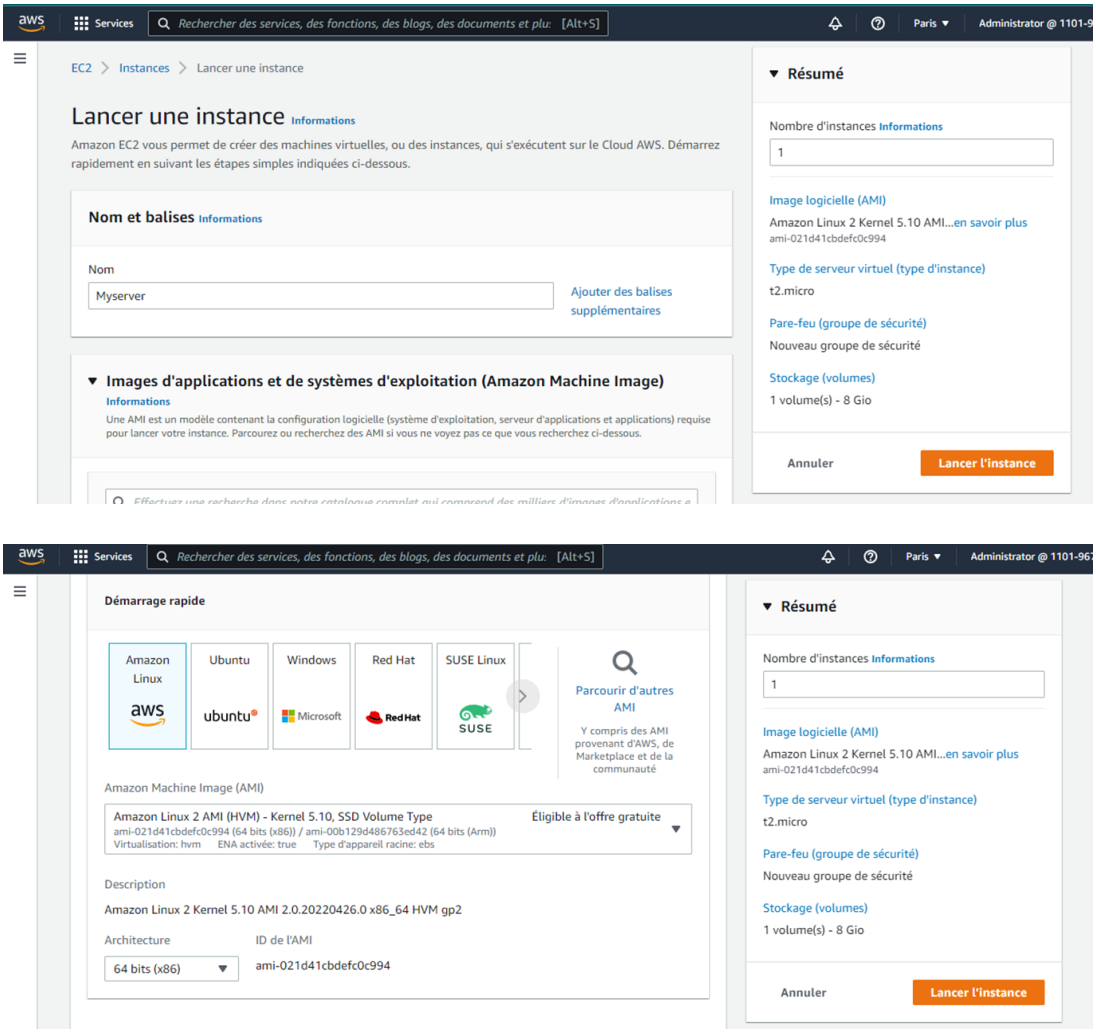


Figure 5.10: Instance EC2

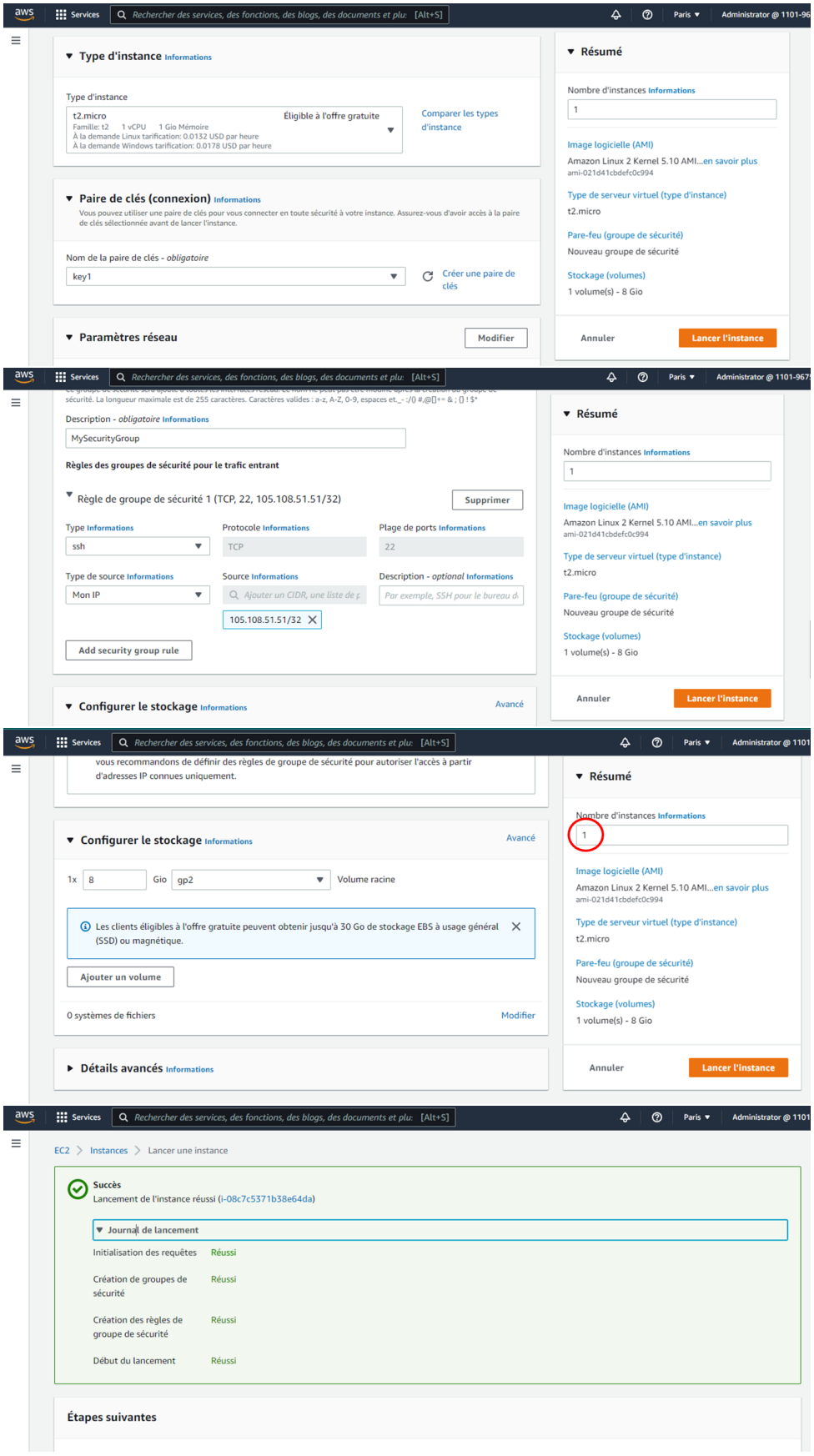


Figure 5.11: Instance EC2

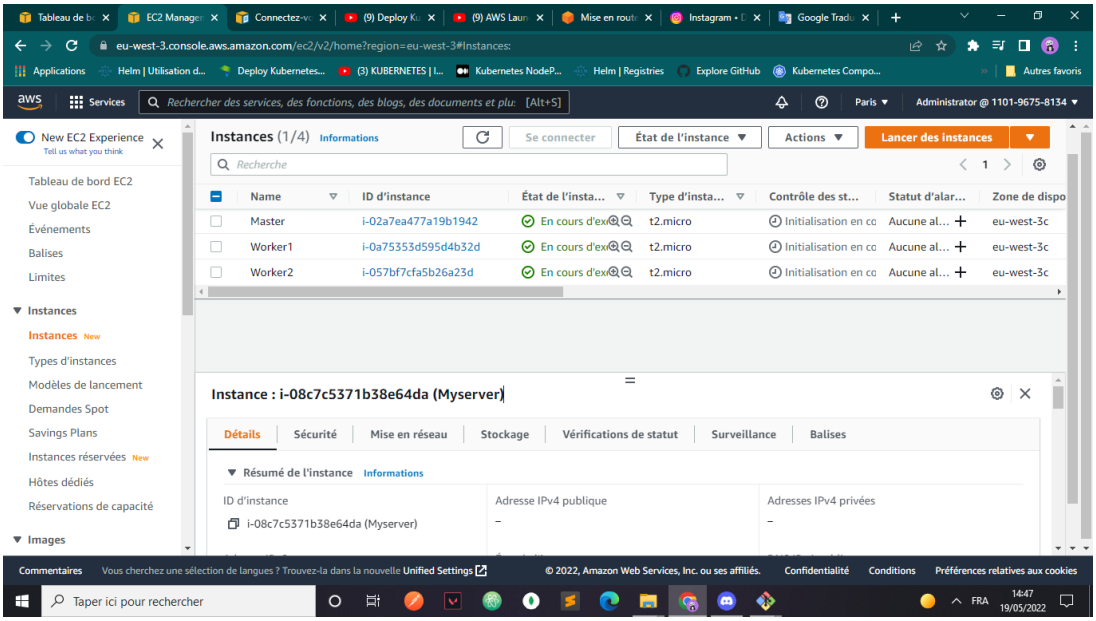


Figure 5.12: Les trois instances crée

- **Installation:** Nous procéderons à l'installation des outils nécessaires sur les trois instances créées.

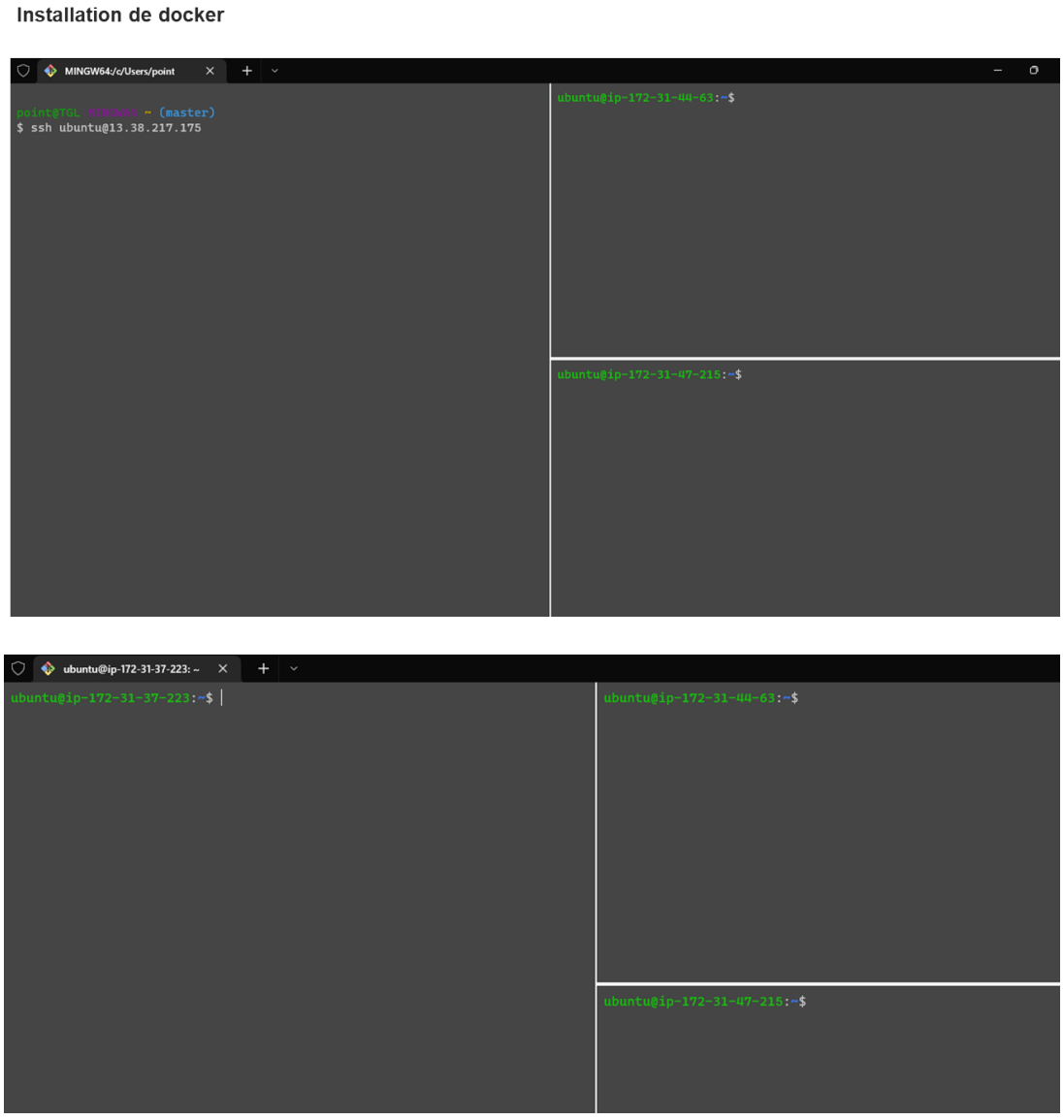


Figure 5.13: Installation sur les trois instances

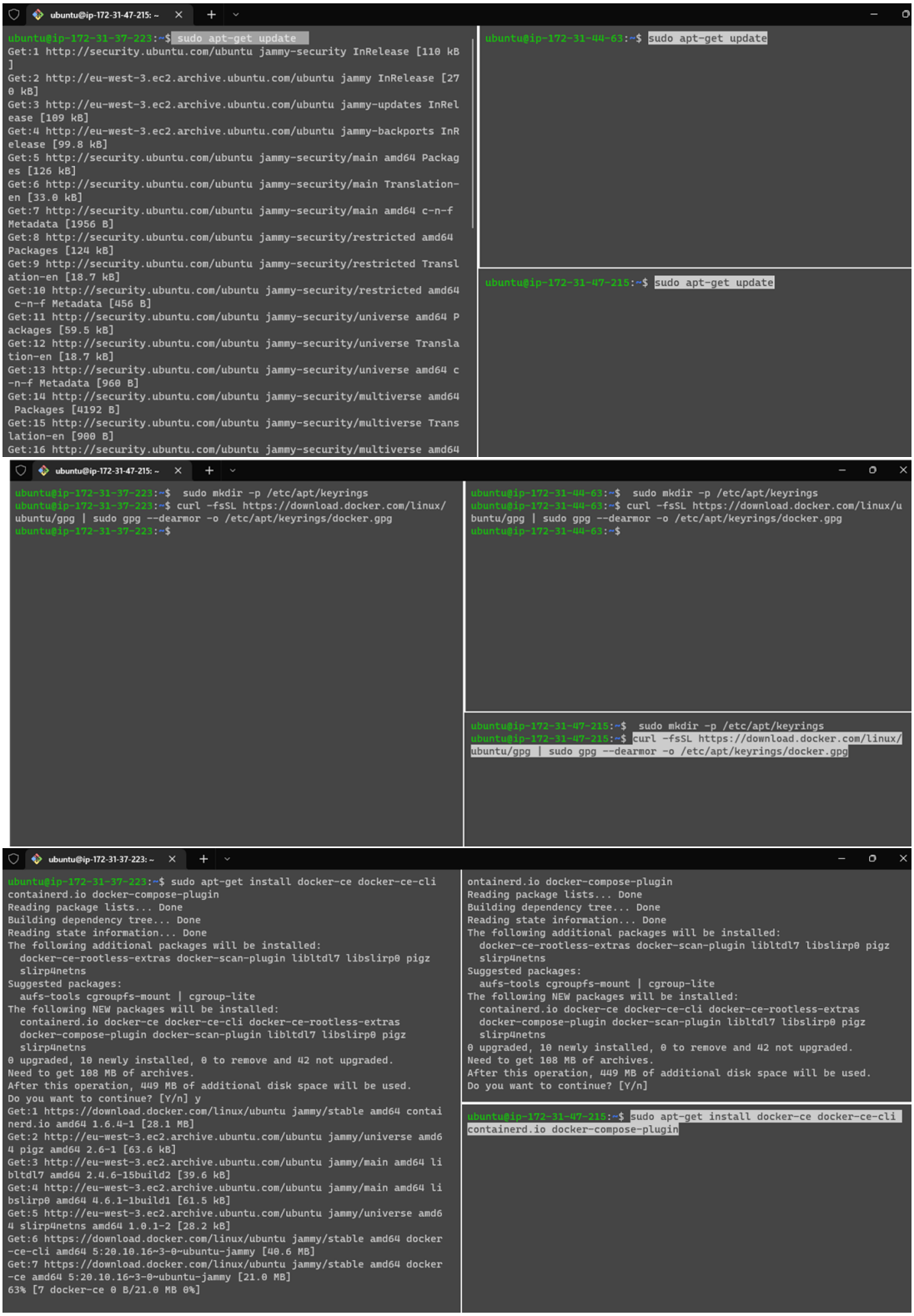
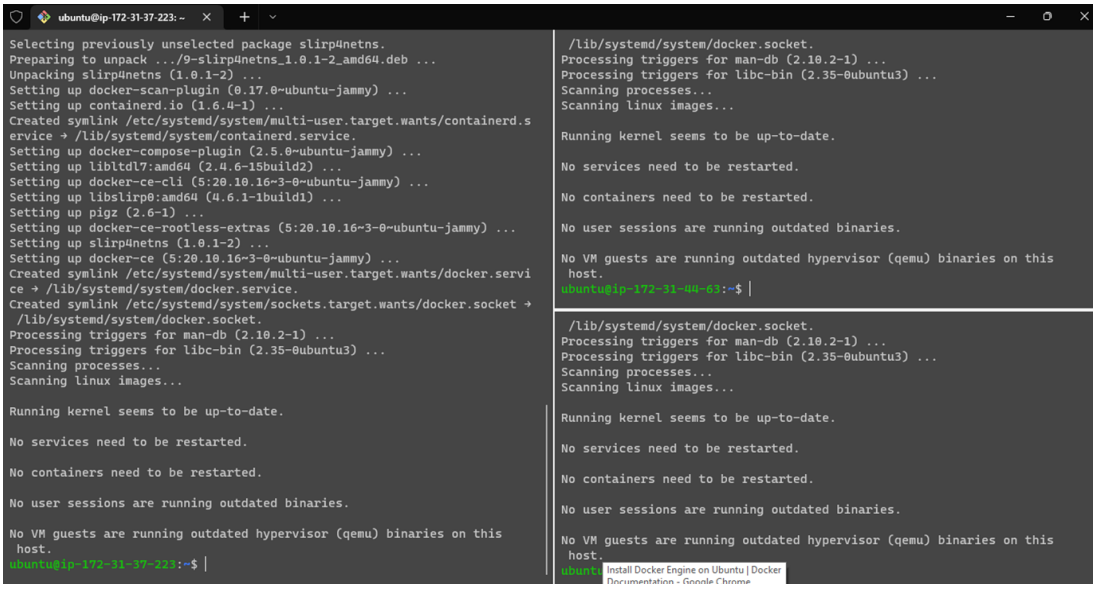
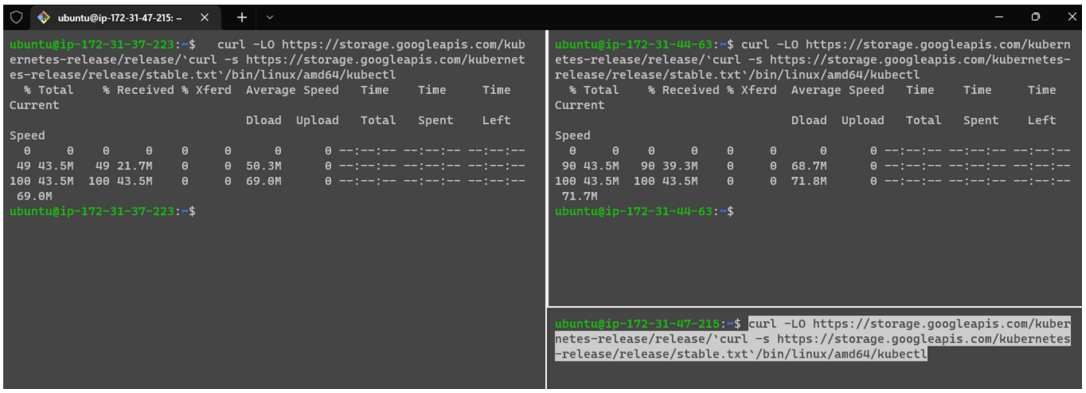


Figure 5.14: Installation sur les trois instances



Installation de kubectl



Installation de kubeadm

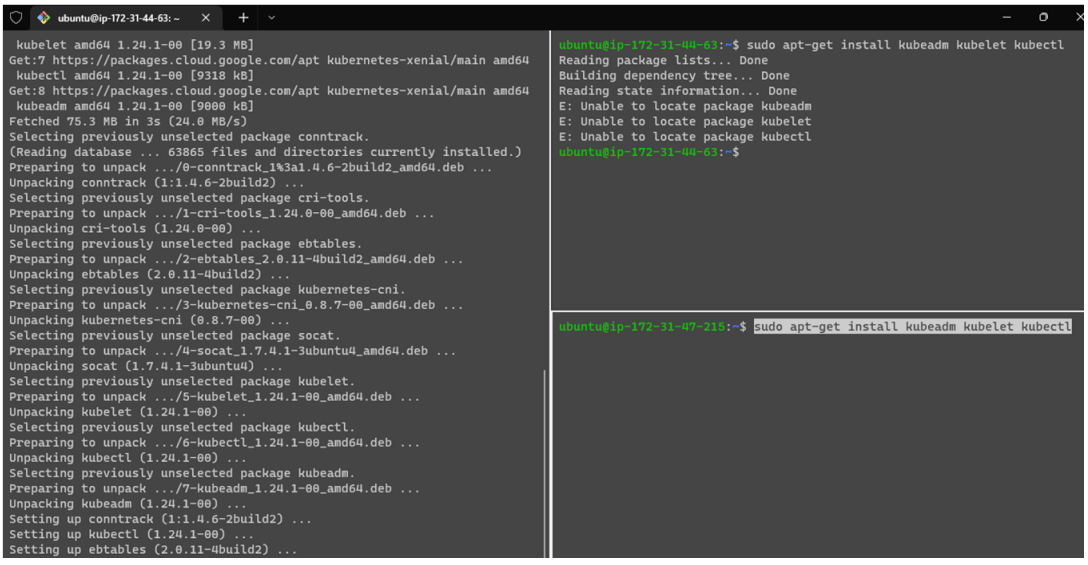
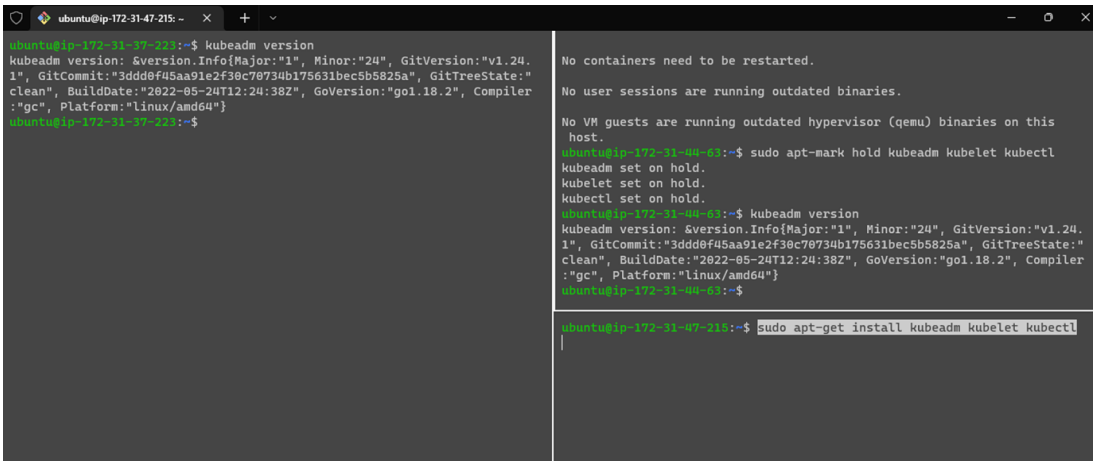


Figure 5.15: Installation sur les trois instances



Initialisation de kubeadm

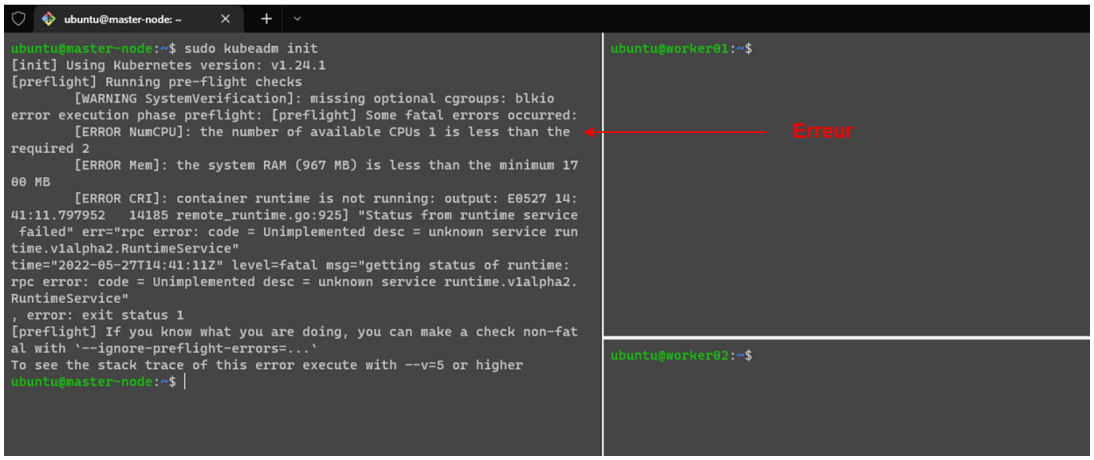
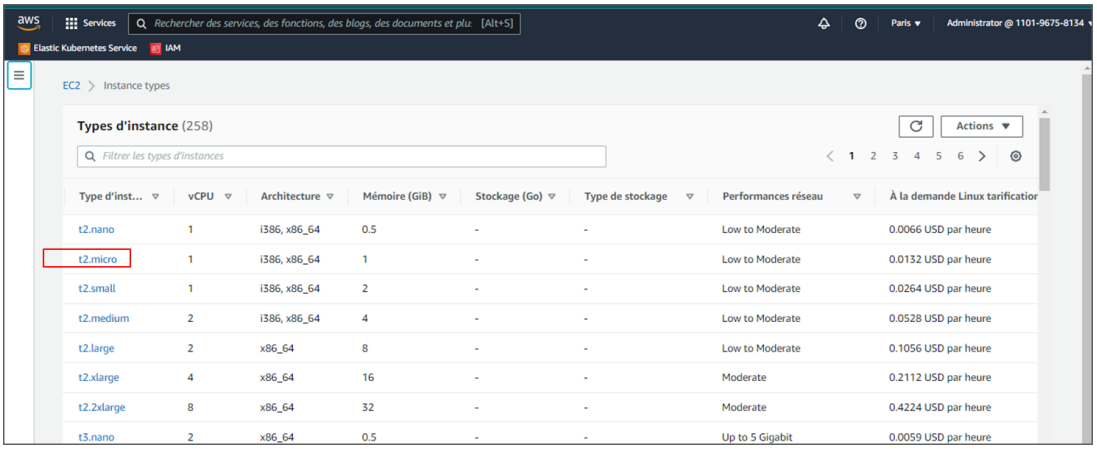


Figure 5.16: Installation sur les trois instances



RQ : le type d'instance inclus dans l'offre gratuite est t2.micro qui possède une capacité limitée qui ne répondait pas à nos besoins

Figure 5.17: Types d'instances

5.3 Avec microsoft azure

Azure est la plate-forme cloud de Microsoft, rassemblant plus de 200 services cloud différents, notamment le calcul, l'analyse, le stockage et la mise en réseau.
Azure Kubernetes Service (AKS) offre le moyen le plus rapide de commencer à développer et à déployer des applications cloud natives.

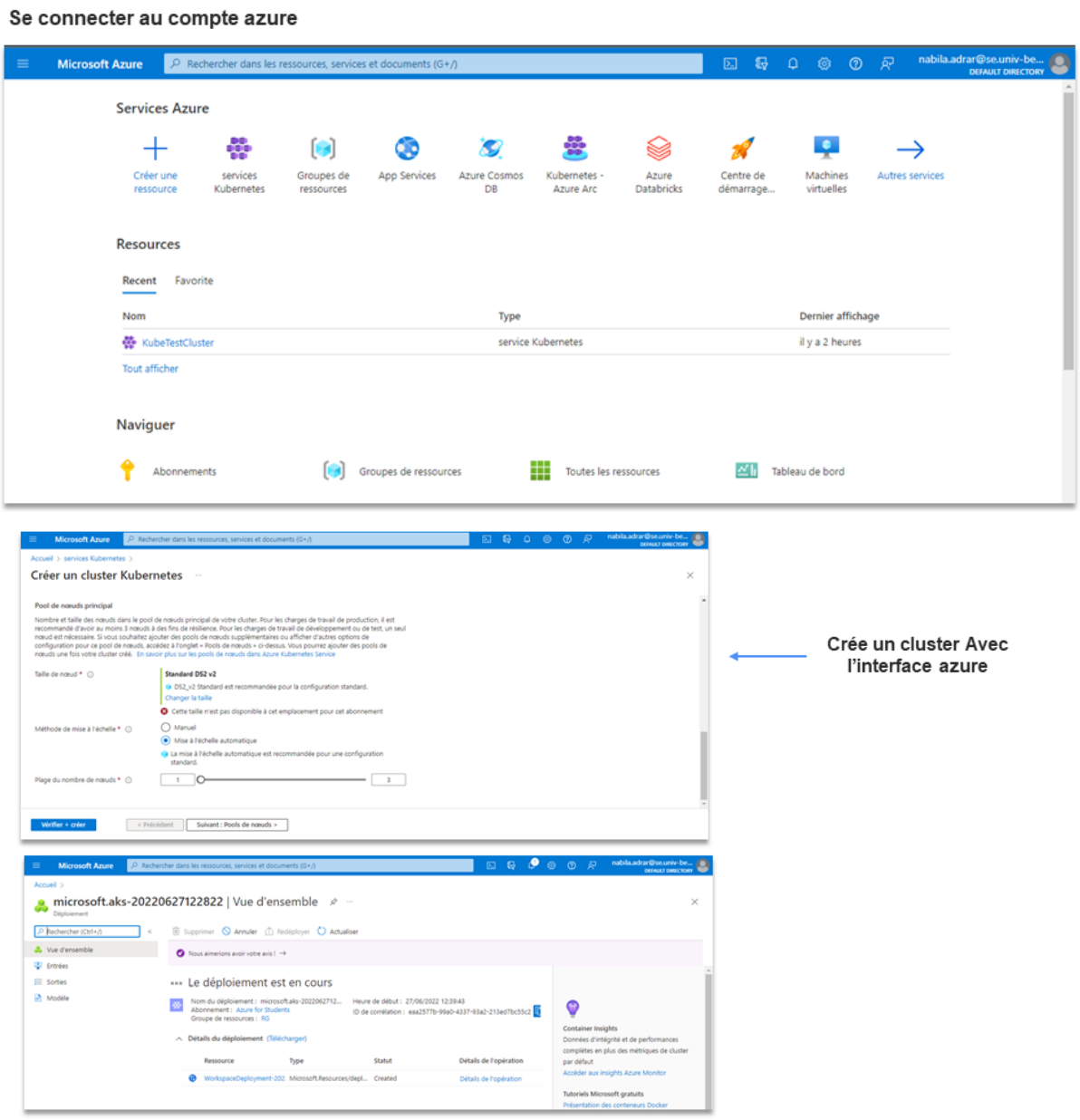


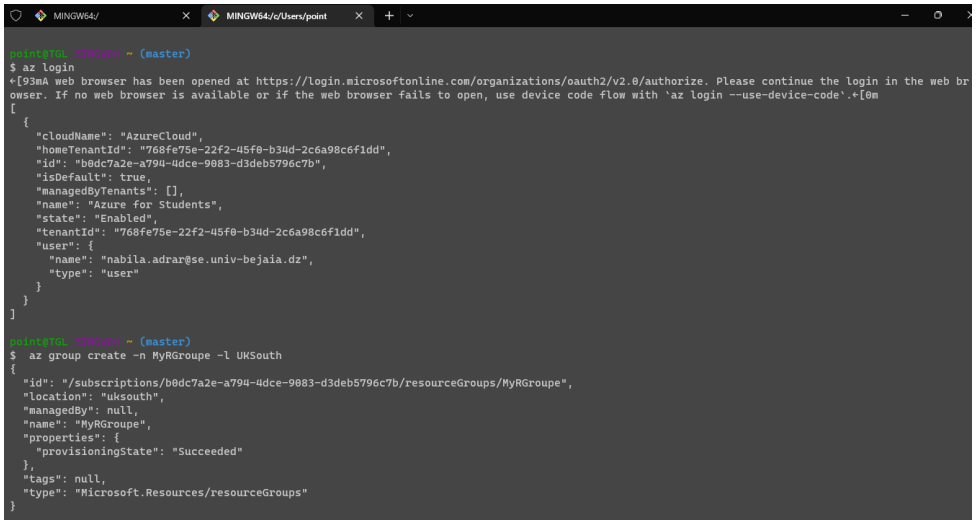
Figure 5.18: Crée un cluster avec l'interface Azure

Crée un cluster avec Azure CLI

```
az login
az account set –subscription b0dc7a2e-a794-4dce-9083-d3deb5796c7
```

```
az group create -n MyRGoupe -l uksouth
```

```
az aks create -g MyRGoupe -n KubeTestCluster -s Standard_D4s_v3 -c
3 -l uksouth --generate-ssh-keys
```



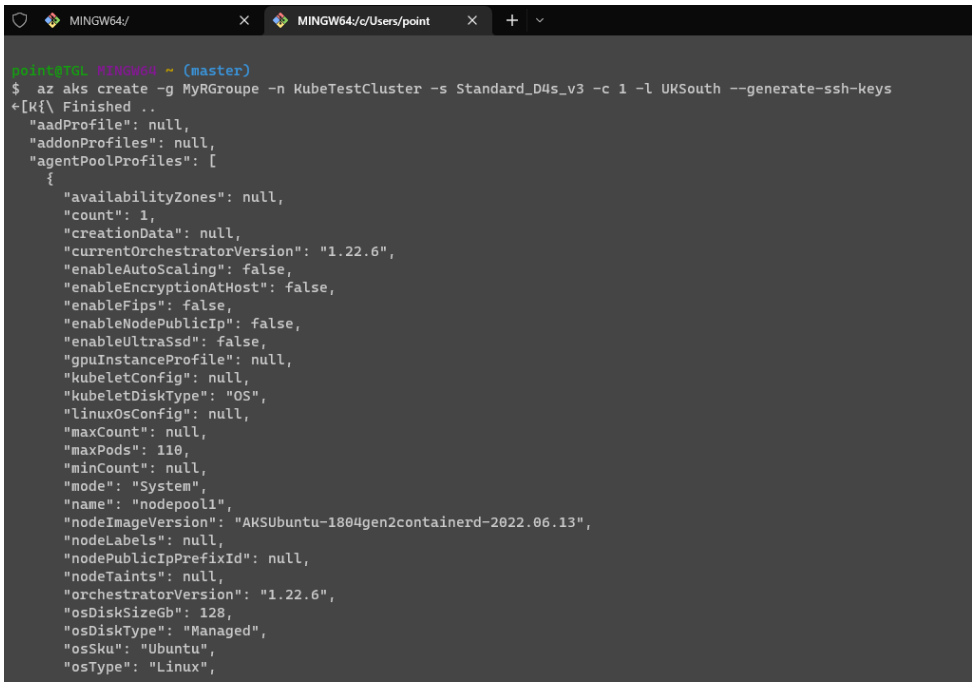


Figure 5.19: Crée un cluster avec Azure CLI

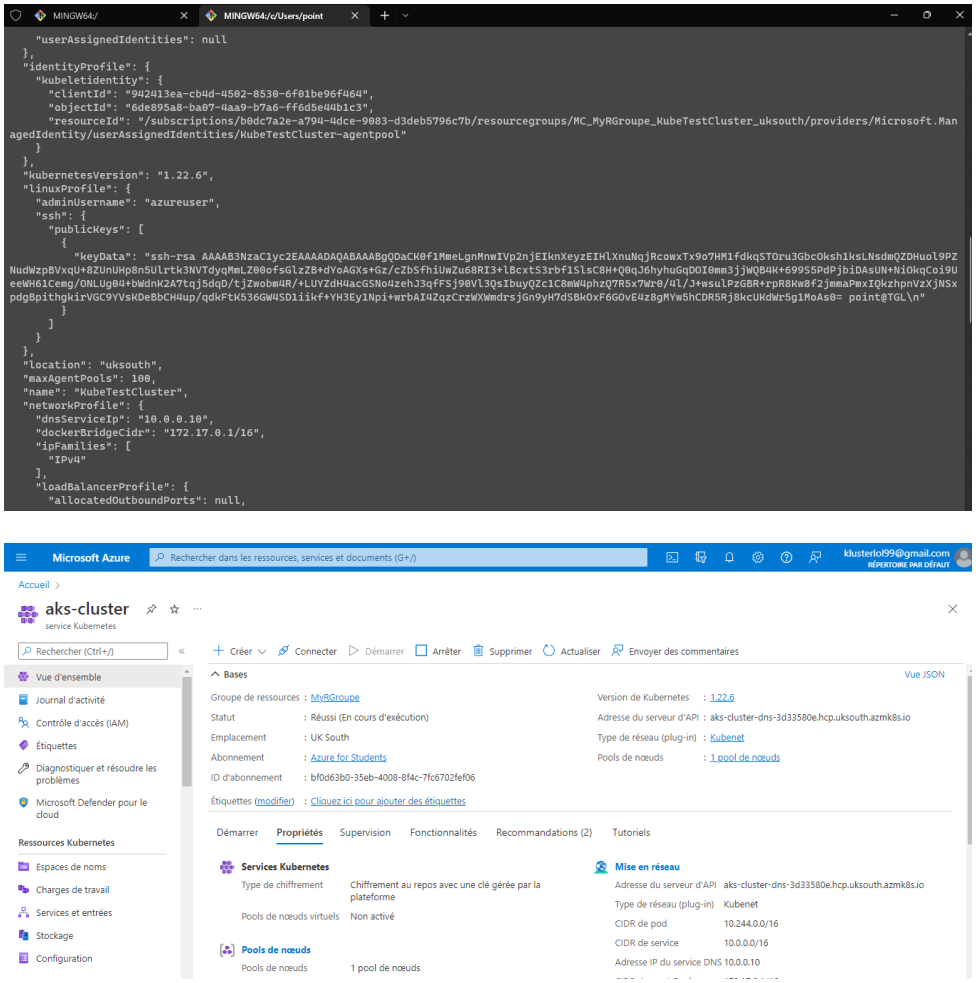


Figure 5.20: Le cluster crée

Connecter au cluster avec Azure CLI

```
az account set --subscription b0dc7a2e-a794-4dce-9083-d3deb5796c7b
```

```
az aks get-credentials --resource-group RG --name AKS-Cluster
```

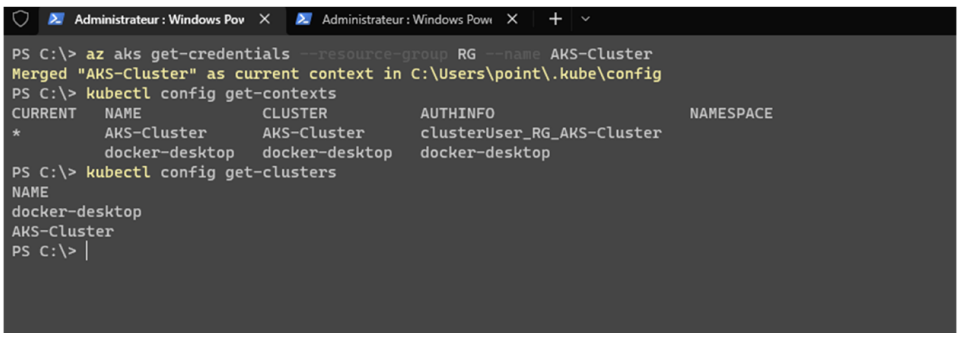


Figure 5.21: Se connecter au cluster

Pour commencer, nous allons tester les deux méthodes que nous propose azure: créer une application de base et créer une application d'image unique.

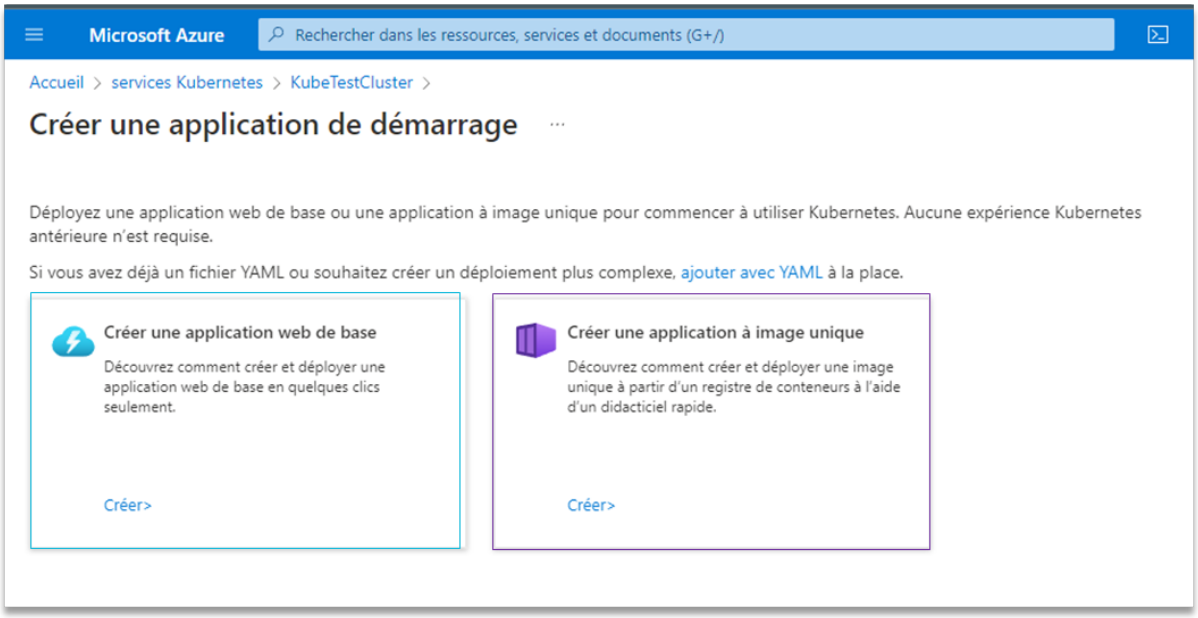


Figure 5.22: Deux méthodes de déploiement

1. **créer une application de base:** Pour bien comprendre le fonctionnement de cette étape, nous avons implémenté une image simple fournie par Azure, située dans mcr.microsoft.com, qui est le registre d'artefacts de Microsoft.

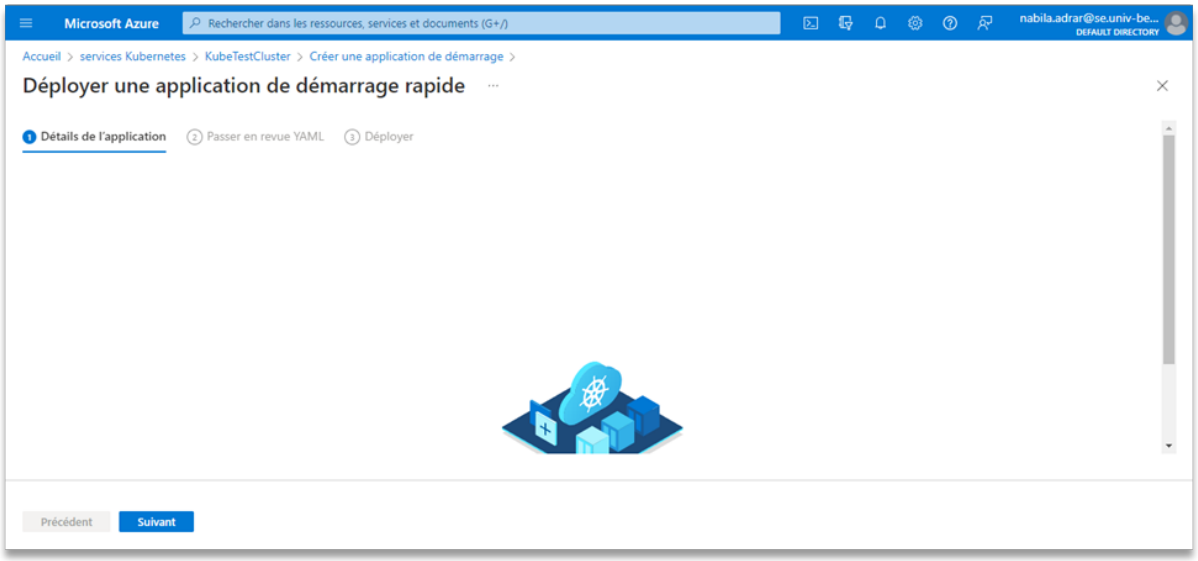


Figure 5.23: Méthode une

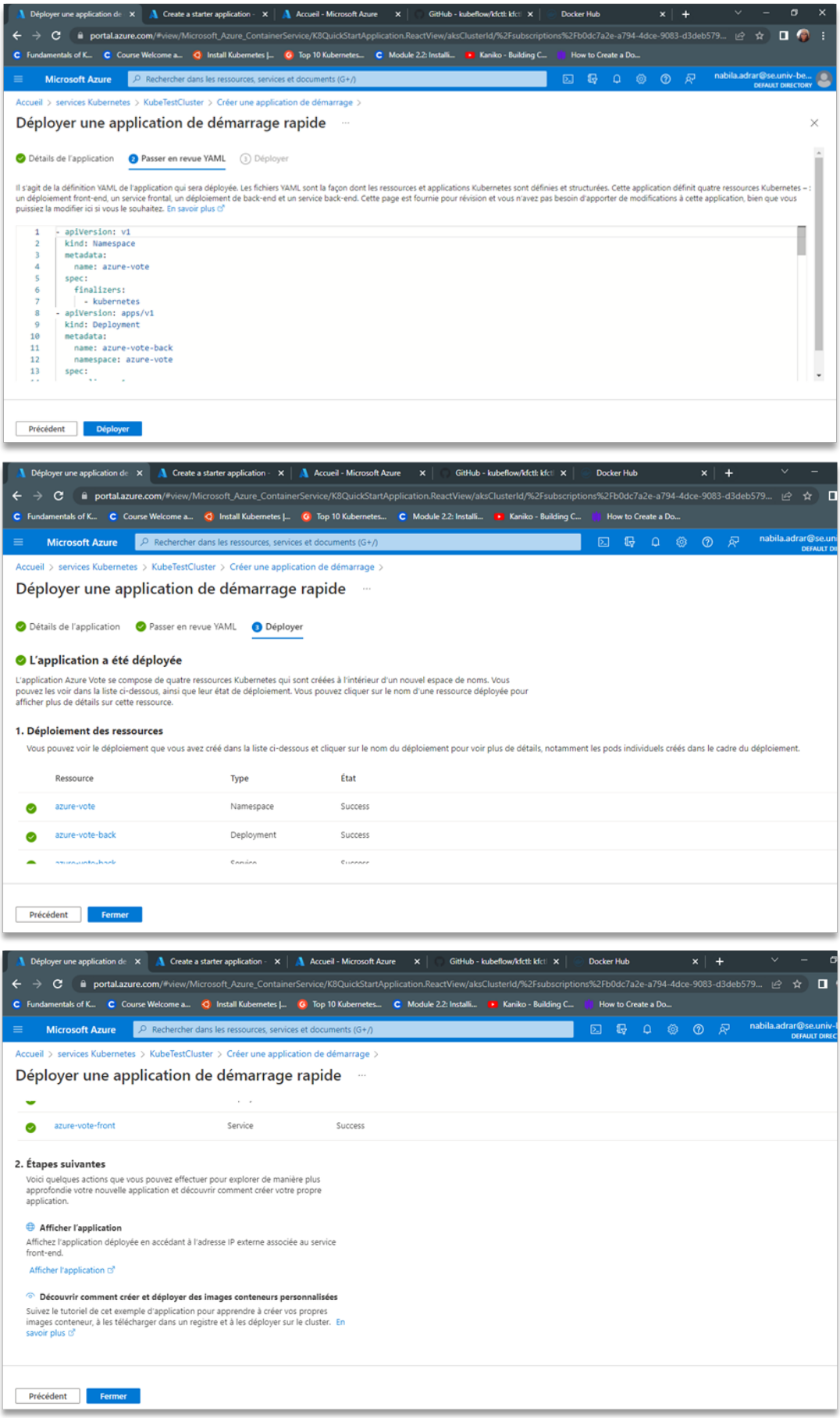


Figure 5.24: Méthode une étapes

2. **créer une application d'image unique:**
   Ici, nous allons utiliser le registre de conteneurs Azure, pour cela Nous allons créer un registre Azure puis nous taguerons nos images locales puis nous pousserons les images vers le registre créé pour enfin les déployer.

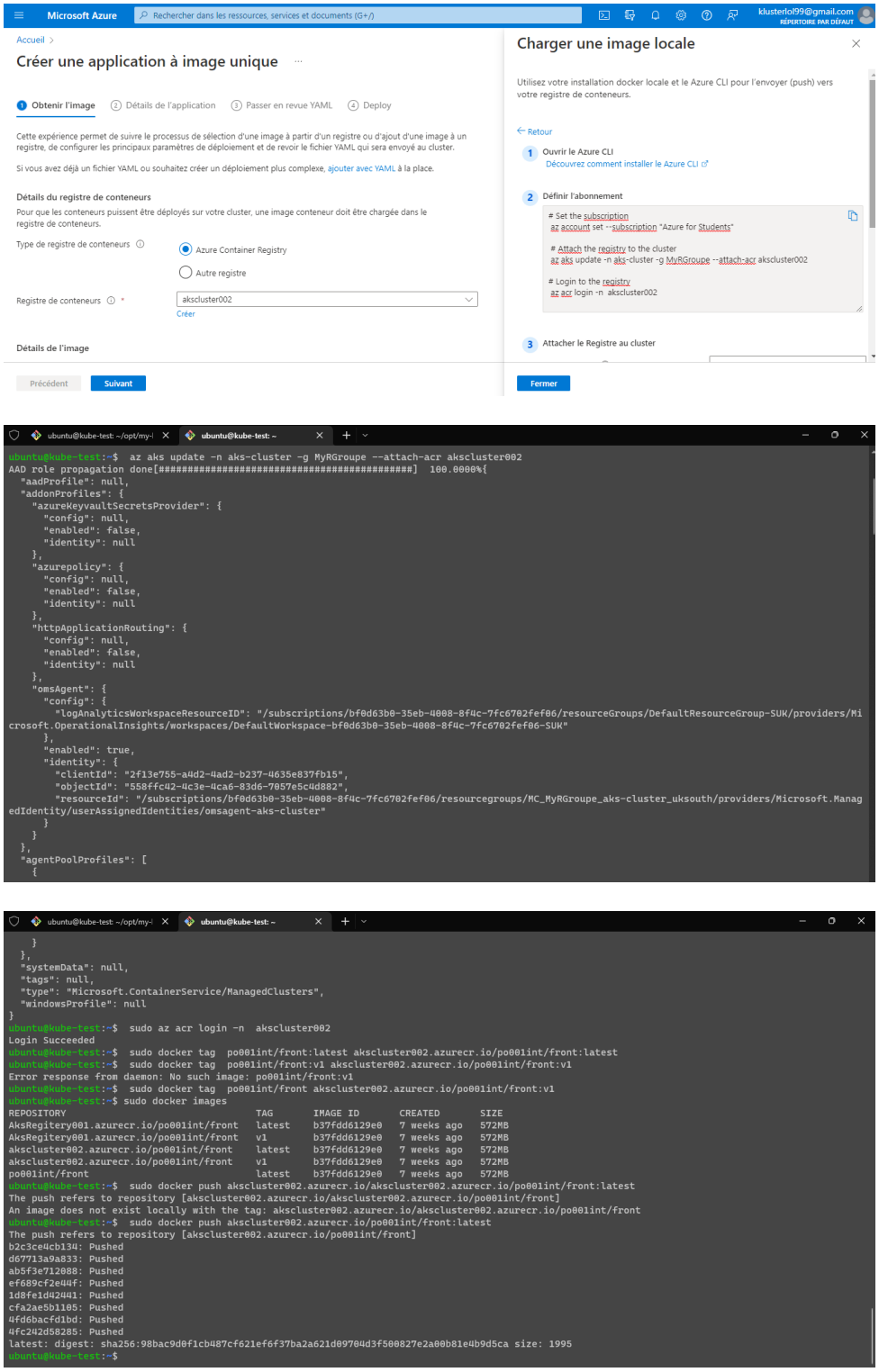


Figure 5.25: Méthode deux

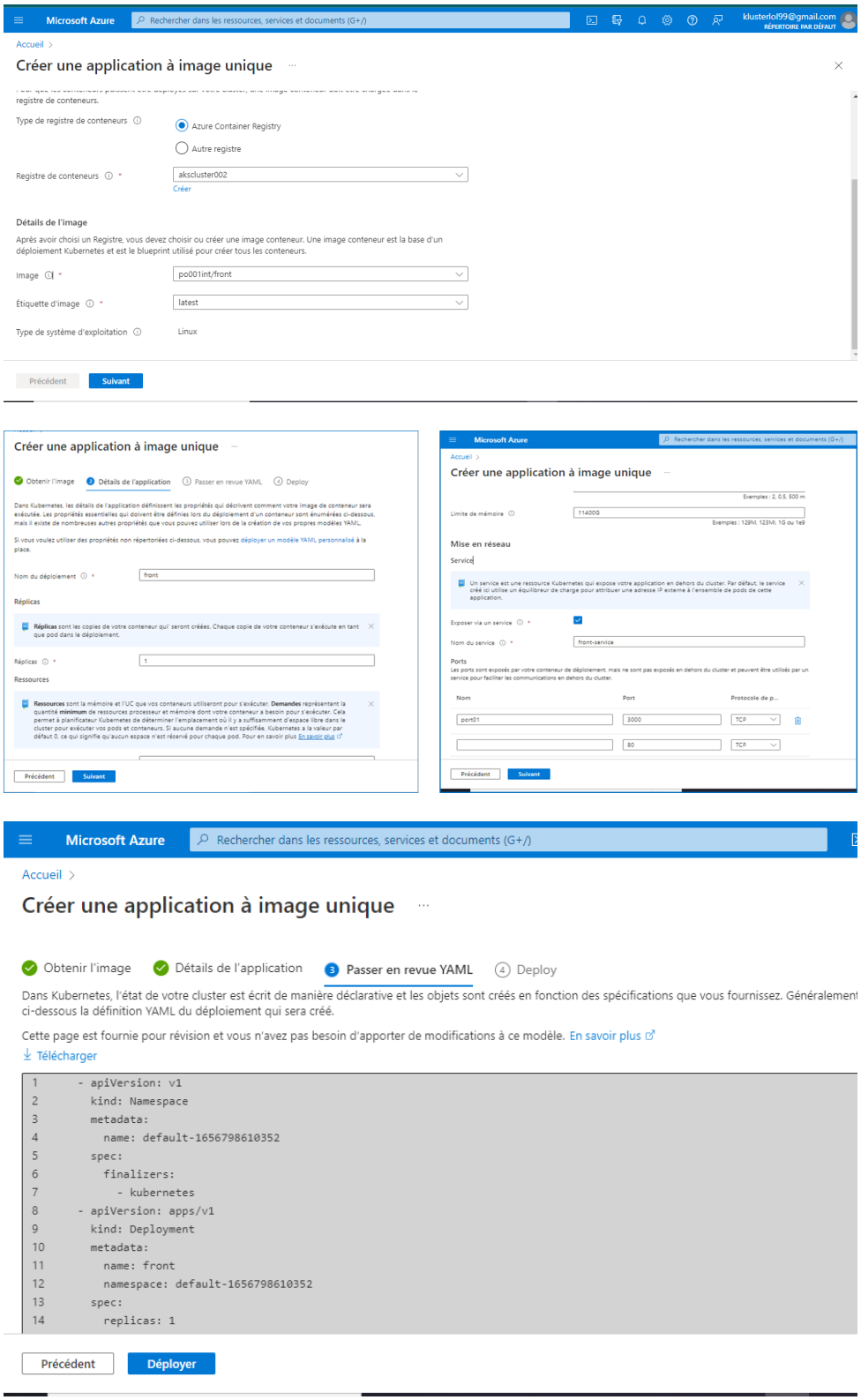


Figure 5.26: Méthode deux étapes

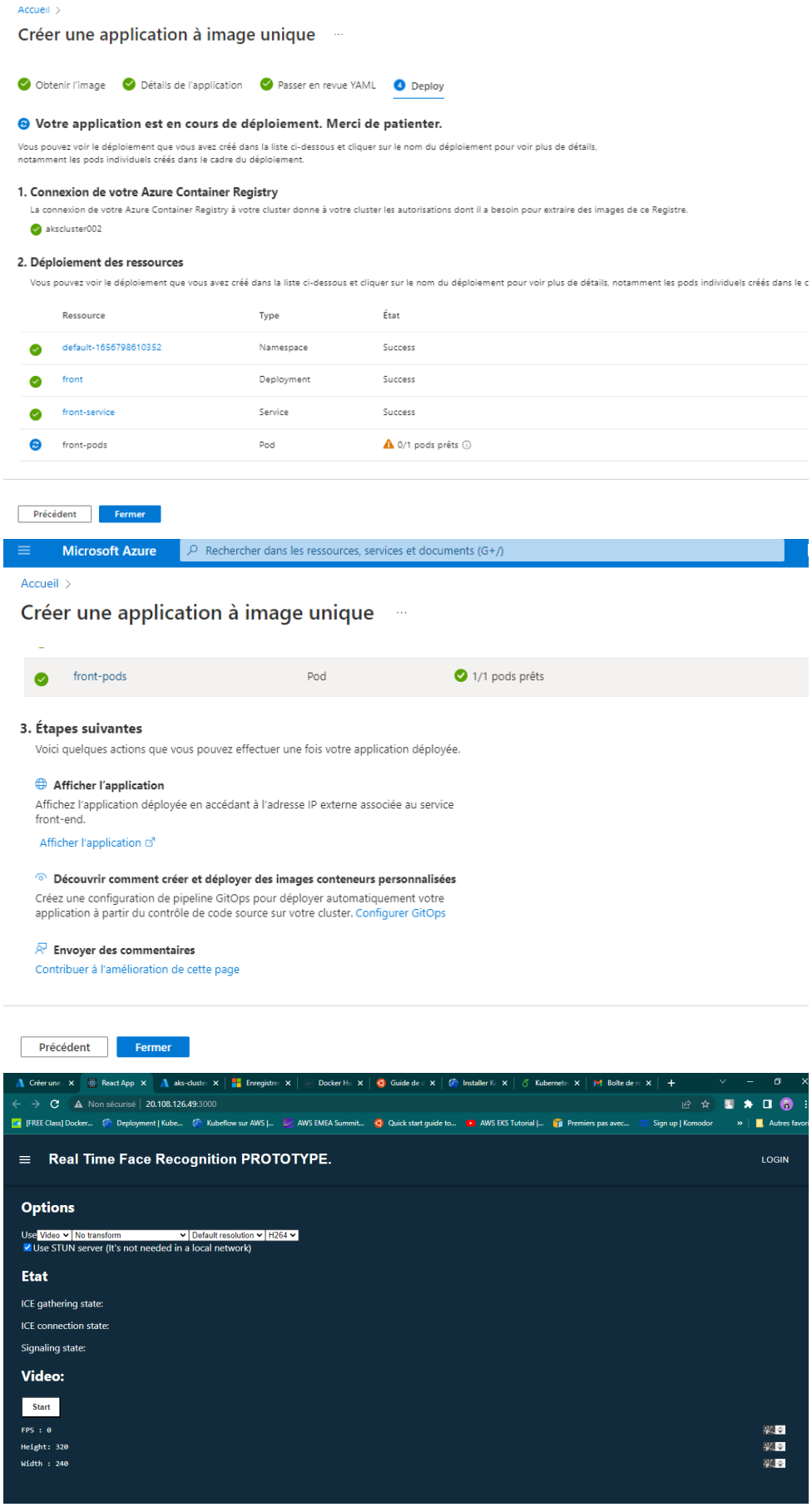


Figure 5.27: Déployer le front

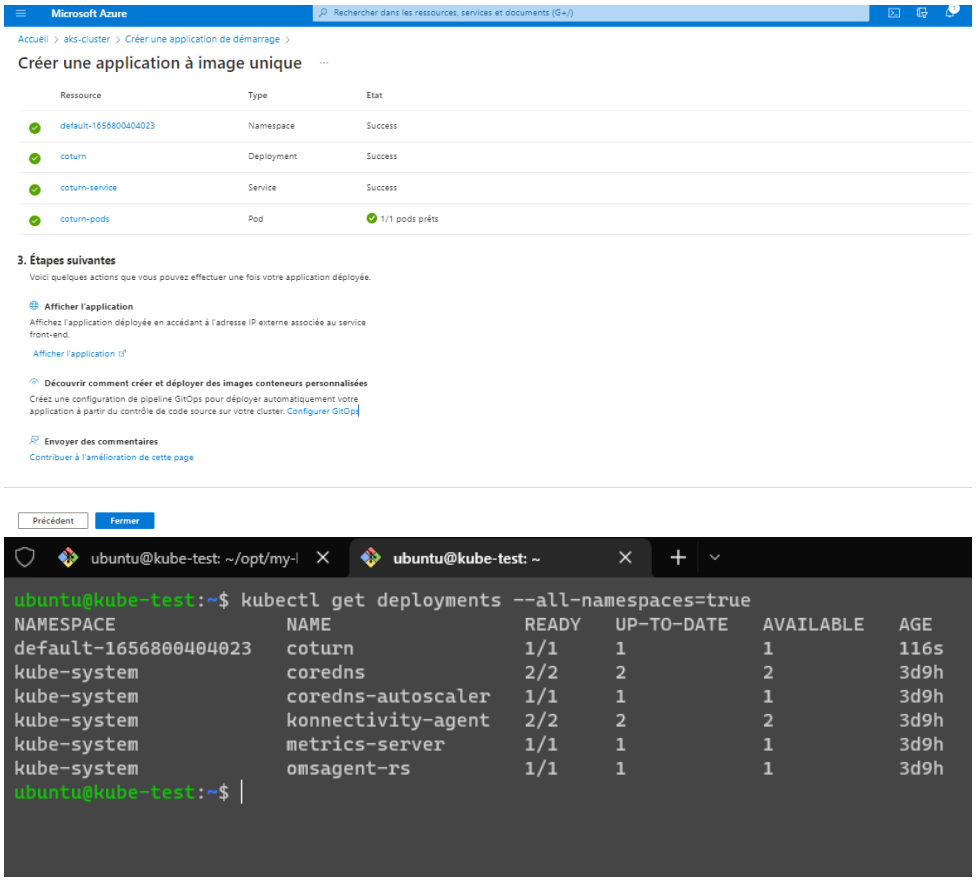


Figure 5.28: Déployer le coturn

3. **Deploiement de kubeflow**
   Nous aurons besoin de kfctl pour exécuter les commandes qui contrôlent le déploiement. Après la connexion à Azure, nous allons installer kfctl puis suivre les instructions suivantes. [33]

```
# The following command is optional. It adds the kfctl binary to your path.
# If you don't add kfctl to your path, you must use the full path
# each time you run kfctl.
# Use only alphanumeric characters or - in the directory name.
export PATH=$PATH:"<path-to-kfctl>"

# Set KF_NAME to the name of your Kubeflow deployment. You also use this
# value as directory name when creating your configuration directory.
# For example, your deployment name can be 'my-kubeflow' or 'kf-test'.
export KF_NAME=<your choice of name for the Kubeflow deployment>

# Set the path to the base directory where you want to store one or more
# Kubeflow deployments. For example, /opt/.
# Then set the Kubeflow application directory for this deployment.
export BASE_DIR=<path to a base directory>
export KF_DIR=${BASE_DIR}/${KF_NAME}

# Set the configuration file to use when deploying Kubeflow.
# The following configuration installs Istio by default. Comment out
# the Istio components in the config file to skip Istio installation.
# See https://github.com/kubeflow/kubeflow/pull/3663
export CONFIG_URI="https://raw.githubusercontent.com/kubeflow/manifests/v1.2-branch/kfdef/kfctl_k8s_ist

mkdir -p ${KF_DIR}
cd ${KF_DIR}
kfctl apply -V -f ${CONFIG_URI}
```

Figure 5.29: Etapes du déploiement de kubeflow sur azure

Figure 5.30: Déploiement de kubeflow sur azure

Conclusion Générale

Cette recherche a été menée en mettant l'accent sur les possibilités de mettre en production des modèles d'apprentissage automatique, Nous avons tout d'abord commencé par la mise en œuvre de docker, une technologie de conteneurisation conçue pour exécuter un service ou une application unique de manière redondantes, ainsi que Kubernetes conçu à l'origine pour orchestrer des applications de microservices à grande échelle.

Nous avons ensuite testé le déploiement local avec un seul nœud à l'aide de l'outil Minikube, ce qui nous a permis de tester les images officielles extraites du registre central de Docker appelé Docker Hub ainsi que les microservices de notre système. Cependant, nous avons rencontré de nombreux problèmes que nous avons pu traiter par la suite comme : Dockershim, Imagepullbackoff, Installation...

Ensuite, il y a le déploiement cloud que nous avons effectué sur les plates-formes cloud dominantes : AWS et Azure, Nous avons ainsi découvert les conditions de fonctionnement et de production de différents services dans le cloud.Le résultat parvenue de ce travail est la mise en place de kubeflow, une solution conçue pour développer des applications de machine learning et déployer sur Kubernetes.

Toutes les étapes franchies au cours de ce travail nous a permis de découvrir et de développer des concepts liés aux technologies les plus courantes, d'essayer certaines possibilités, de déboguer certains problèmes, Et pour finalement arriver à une solution qui n'est peut-être que le début d'un autre problème qui peut continuer à améliorer le développement de notre travail.

Bibliography

[1] https://venturebeat.com/2019/07/19/why-do-87-of-data-science-projects-never-make-it-into-production/.

[2] https://www.seldon.io/how-to-deploy-your-machine-learning-models.

[3] https://fr.wikipedia.org/wiki/Cycle$_e n_V$.

[4] https://www.edureka.co/blog/devops-lifecycle/.

[5] https://milannovovic.medium.com/robot-framework-ci-cd-with-azure-devops-cf708a64b389.

[6] http://www.https://codefresh.io/learn/software-deployment/section-software-deployment-strategiesl.

[7] https://www.fiddler.ai/mlops.

[8] https://databricks.com/glossary/mlops.

[9] **2**.

[10] https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained.

[11] https://developer.nvidia.com/blog/choosing-a-server-for-deep-learning-training/.

[12] https://www.lebigdata.fr/gpu-ia-big-data.

[13] https://scikit-learn.org/stable/modules/clustering.html.

[14] https://fr.ryte.com/wiki/Release$_m anagement$.

[15] https://microservices.io/.

[16] https://www.lebigdata.fr/reconnaissance-faciale-tout-savoir.

[17] https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/benefits.

[18] https://www.stackscale.com/blog/cloud-service-models/.

[19] https://datascientest.com/docker-guide-complet.

[20] https://www.sdxcentral.com/cloud/containers/definitions/containers-vs-vms/.

[21] https://datascientest.com/docker-guide-complet.

[22] https://kubernetes.io/fr/docs/concepts/overview/what-is-kubernetes/.

[23] https://www.journaldunet.fr/web-tech/guide-de-l-entreprise-digitale/1440612-kubernetes-definition-architecture-conseil/.

[24] https://wiki.sfeir.com/kubernetes/architecture/composants/services/.

[25] https://www.seldon.io/deploying-machine-learning-models-on-kubernetes: :text=Instead%20of%20different%20parts%20of,on%20local%20or%20cloud%20servers.

[26] https://platform9.com/blog/why-and-how-to-run-machine-learning-workloads-on-kubernetes/.

[27] https://ubuntu.com/ai/what-is-kubeflow.

[28] https://blent.ai/kubernetes-et-machine-learning/.

[29] https://github.com/RumbleFiend/recog-coturn.

[30] https://github.com/RumbleFiend/recog-front.

[31] https://github.com/RumbleFiend/recog$_p rototype$.

[32] https://docs.aws.amazon.com/fr$_f r/eks/latest/userguide/what - is - eks.html$.

[33] https://www.kubeflow.org/docs/distributions/azure/deploy/install-kubeflow/.