1. Kubernetes et les nouvelles façons de partager vos applications

Dans cette section, nous allons explorer :

- Comment transformer nos applications d'une simple idée à quelque chose utilisé par les gens.

- Faire des mises à jour et des changements sur nos applications en utilisant quelque chose appelé contrôle de version.

- Résoudre le casse-tête pour nous assurer que notre application possède tout ce dont elle a besoin pour fonctionner.

- Comprendre le rôle des conteneurs pour maintenir l'environnement de notre application cohérent.

- Ce qui se passe lorsque nous utilisons des conteneurs sans aide pour les gérer.

- Comment Kubernetes intervient pour nous aider à gérer ces conteneurs.

Avant de plonger dans le monde de Kubernetes et de ses fonctionnalités impressionnantes, il est important de comprendre le parcours depuis le tout début de la création de logiciels jusqu'à l'utilisation de Kubernetes. Il ne s'agit pas de l'histoire de la technologie, mais des compétences de base nécessaires pour partager votre logiciel avec le monde.

Ce guide est pour toi si tu as déjà commencé ton aventure dans la programmation, surtout dans la création d'applications ou de logiciels.

Si tu te grattes la tête en lisant des termes comme fonctions (les éléments de base des actions dans ton application), classes (les plans pour créer des parties de ton application), variables (les espaces réservés pour les informations dans ton application), ou substitution de chaîne (échanger des parties de texte dans ton application), tu pourrais vouloir faire une pause. Suivre un cours de base en programmation orientée objet, qui consiste à apprendre comment organiser et structurer ton code, en utilisant Python ou JavaScript, pourrait être une excellente prochaine étape. Voici pourquoi :

- Python et JavaScript sont comme les voisins amicaux des langages de programmation - ils sont faciles à appréhender pour les débutants (Python est un peu comme le plus sympathique).

- Ces deux langages sont comme les élèves populaires à l'école - presque tout le monde les utilise.

- Dans ce guide, nous te montrerons comment mettre en ligne une application Python et une application JavaScript pour que le monde puisse les utiliser.

Imagine Kubernetes comme une application super intelligente qui aide à gérer d'autres applications, surtout lorsqu'elles sont divisées en plus petites parties (conteneurs) pour fonctionner de manière plus efficace. Ce guide t'aidera à comprendre comment passer d'une simple application à l'utilisation de Kubernetes pour la gérer comme un pro.

****1.1 Rendre le Partage d'Applications Facile pour Tout le Monde****

Imagine que tu aies créé un super jeu vidéo ou une application de médias sociaux. Maintenant, tu veux le partager avec le monde. Ce processus de partager ton application pour que d'autres puissent l'utiliser s'appelle le déploiement. Et devine quoi ? Si tu comprends l'anglais, tu es déjà sur la bonne voie pour le faire. Même si certains livres ou guides font du déploiement un véritable casse-tête, c'est quelque chose que tu peux réellement maîtriser. Ce livre est comme un guide pour t'aider, toi et tes amis (ou ton équipe), à apprendre comment partager vos applications avec le monde, étape par étape.

Pense à ce livre comme à un jeu vidéo, où chaque chapitre est un niveau. Tu peux prendre ton temps pour maîtriser un niveau avant de passer au suivant. Nous commencerons par des choses simples et t'introduirons progressivement à des outils et astuces plus avancés et plus cool. Mais voici le truc : tant que tu suis le guide étape par étape, tu découvriras que même les choses avancées sont faciles à comprendre.

La principale leçon à retenir, c'est que le déploiement n'est pas réservé uniquement aux sorciers de la technologie ; c'est pour quiconque a une passion pour la création et le partage de ses créations numériques, tout comme toi. À chaque chapitre, tu développeras tes compétences, en partageant une application simple jusqu'à l'utilisation d'outils de haute technologie, tout en le rendant aussi simple que bonjour.

Entendre parler de ****Kubernetes**** et de ****Docker**** pourrait te faire penser à des choses technologiques super compliquées, mais en réalité, elles ne sont pas si intimidantes.

Toutes deux sont comme les génies du monde informatique : elles font exactement ce que tu souhaites sans que tu aies besoin de leur dire comment le faire. C'est ce qu'on appelle être **déclaratif**, ce qui signifie que tu déclares simplement, ou énonce, ce que tu veux, et tu laisses le "comment" à elles.

Décortiquons quelques exemples de souhaits (déclarations) que tu pourrais dire à ces génies :

- "Je souhaite utiliser cette superbe application Python appelée `insérer_le_nom_de_votre_application`."

- "Je veux qu'elle comprenne et utilise la version 3.10 de Python, car c'est le langage qu'elle parle."

- "S'il te plaît, ouvre-la sur les ports 80 et 443," ce qui revient à dire, "Utilise ces portes d'entrée pour que les gens sur internet puissent venir nous rendre visite."

- "Permet à quiconque sur internet de passer et de jeter un coup d'œil."

- "Et utilisons example.com comme notre adresse pour que les gens sachent où nous trouver."

Lorsque tu formules ces souhaits, tu te concentres sur ce que tu veux que le résultat final soit. Tu n'as pas besoin de t'inquiéter des étapes compliquées pour que cela se produise - Kubernetes et Docker s'occupent de toute la magie en coulisses. C'est comme dire à un constructeur : "Je veux une cabane dans les arbres avec un toboggan et une balançoire," sans avoir à savoir quoi que ce soit sur la charpenterie ou la conception. Tu attends simplement avec impatience la partie amusante : la cabane dans les arbres (ou dans ce cas, ton application qui fonctionne parfaitement pour que les gens l'utilisent).

Imagine Kubernetes comme ton génie de la technologie personnel qui prend tes souhaits (ou déclarations) et les réalise, créant l'environnement parfait pour ton application sans que tu aies besoin de superviser chaque petit détail.

Considère les **fichiers déclaratifs** comme une recette ou une liste de souhaits pour ton application. Lorsque tu cuisines un nouveau plat, tu suis une recette qui te dit ce dont tu as besoin et à quoi le plat final devrait ressembler, mais pas nécessairement chaque étape en détail. De même, lorsque tu exprimes tes souhaits à un génie, tu dis ce que tu veux, pas comment le réaliser. C'est ce que ces fichiers font pour le déploiement des applications.

Élaborer cette "recette" peut sembler un peu compliqué au début. Tu dois être clair sur ce que tu veux, comme choisir les bons ingrédients ou rendre tes souhaits spécifiques. Mais la beauté, c'est que une fois que tu as compris, tu peux l'utiliser encore et encore. Envie de préparer le même plat chez un ami ? Apporte simplement ta recette. Souhaites-tu quelque chose à un génie pendant tes vacances ? Le souhait est le même, peu importe où tu te trouves.

Dans le monde du déploiement d'applications :

- ****Kubernetes**** appelle ces recettes ****Manifestes****. C'est comme ta liste de souhaits détaillée pour la manière dont ton application devrait être configurée et exécutée.

- Pour **Docker**, qui aide ton application à s'exécuter de la même manière sur n'importe quel ordinateur, ces recettes sont connues sous le nom de **Dockerfiles**.

- Les **serveurs Web** comme **Apache** et **NGINX**, qui aident les gens à accéder à ton application via Internet, utilisent également leurs propres types de recettes pour savoir comment servir ton application aux visiteurs.

- Ensuite, il y a des outils comme ****Ansible**** et ****Terraform****, un peu comme des livres de sorts magiques, qui utilisent ces recettes pour configurer et gérer les environnements où les applications s'exécutent, en veillant à ce que tout soit en place avant que ton application ne soit mise en ligne.

Une fois que tu as compris comment écrire ces fichiers déclaratifs, c'est comme débloquer un super pouvoir. Tu peux configurer ton application n'importe où, que ce soit sur ton propre ordinateur ou sur un serveur à l'autre bout du monde, juste en utilisant ces "recettes". Et si tu veux déplacer ton application vers un nouveau serveur ou en créer une copie pour gérer plus de visiteurs, tu réutilises simplement le même fichier. C'est aussi simple que ça !

****1.2 Comment Nous Partageons Nos Applications avec le Monde****

Imagine si les applications étaient comme des super-héros. En 2011, un gars intelligent nommé Marc Andreessen a dit : "Le logiciel dévore le monde." Ce qu'il voulait dire, c'est que les applications deviennent super importantes dans toutes les parties de nos vies, tout comme les super-héros qui sauvent la situation dans toutes sortes de situations. Mais voici le truc : un super-héros (ou une application) ne peut pas faire grand-chose s'il est coincé à la maison. Il doit être là-bas dans le monde. C'est de cela dont il s'agit dans le déploiement - mettre votre application là-bas pour qu'elle puisse commencer à faire sa magie.

Alors, où peuvent aller ces applications pour faire leur travail ? À peu près partout :

- **Appareils personnels :** Pense à ton téléphone, tablette, ordinateur, montre intelligente, télévision, voiture, même ton frigo, thermostat, grille-pain, et plus encore. Si cela possède une sorte de cerveau informatique, il peut exécuter des applications.

- **Le cloud :** C'est comme un endroit magique sur internet où les applications peuvent vivre et s'exécuter sans avoir besoin d'être sur ton appareil personnel. Des entreprises comme Amazon, Google et Microsoft ont d'énormes espaces ici pour que les applications puissent vivre.

- **Serveurs physiques :** Ce sont de vrais ordinateurs qui peuvent être n'importe où - dans ton école, dans une entreprise, ou même à la maison. Ils sont comme des maisons dédiées pour que les applications puissent vivre et s'exécuter. Même de petits ordinateurs abordables comme les Raspberry Pi peuvent servir de maisons pour les applications.

- **Déploiements hybrides :** Cela signifie que ton application pourrait vivre dans plusieurs endroits à la fois, comme avoir des maisons en ville, à la campagne et à la plage en même temps !

Cependant, puisqu'il existe tant d'endroits où les applications peuvent vivre, nous allons principalement nous concentrer sur les applications qui résident dans le cloud et sont utilisées via internet. Pourquoi ? Parce que lorsqu'une application est sur internet, elle peut se connecter et travailler avec d'autres applications du monde entier, grâce à la magie du réseau. C'est comme organiser une fête d'application où tout le monde est invité, peu importe où ils vivent.

En termes simples, déployer ton application signifie la libérer pour une aventure. Que ce soit sur un appareil personnel ou flottant dans un nuage, une fois qu'elle est là-bas, elle peut interagir avec d'autres applications, rendant nos vies plus cool, plus faciles, ou simplement plus amusantes.

Imagine que nous nous embarquions dans un voyage, une feuille de route si tu veux, qui va nous emmener à travers le monde du partage de nos créations (applications) avec tout le monde. Ce voyage consiste à apprendre comment mettre nos applications en ligne, en utilisant un super outil appelé Kubernetes, qui nous aide à gérer et à partager nos applications de manière vraiment intelligente.

****Les Étapes de Notre Voyage :****

1. **Création de Nos Applications :** Nous allons commencer par construire deux applications web simples - l'une utilisant Python et l'autre utilisant JavaScript. Pense à ces applications comme à nos premiers projets ou expériences dans l'apprentissage de comment partager ce que nous créons. Même si elles sont basiques, ce sont nos clés pour comprendre tout sur le partage des applications.

2. **Suivi avec Git :** Après avoir construit nos applications, nous avons besoin d'un moyen de suivre toutes les modifications que nous y apportons. C'est là que Git intervient - c'est comme un journal pour notre code. Chaque fois que nous apportons une modification, Git nous aide à la noter, ainsi nous savons toujours quelles modifications nous avons apportées et pouvons revenir à des versions antérieures si nécessaire.

3. **Stockage de Notre Code sur GitHub :** Nous utiliserons GitHub, un endroit sur internet où nous pouvons stocker notre code en toute sécurité. C'est comme un casier cloud pour nos projets, où nous pouvons y accéder de n'importe où et les partager facilement avec d'autres.

4. **Automatisation avec GitHub Actions :** Pour nous faciliter la vie, nous mettrons en place quelque chose appelé GitHub Actions. C'est un outil qui automatise toutes les tâches ennuyeuses pour nous. Par exemple, chaque fois que nous apportons une modification à notre code et le sauvegardons dans notre casier GitHub, GitHub Actions peut automatiquement effectuer des vérifications pour s'assurer que tout fonctionne bien et même aider à préparer notre application à être partagée avec le monde.

Considère ce voyage comme apprendre à conduire. Au début, nous cherchons simplement à démarrer la voiture et à la mettre en mouvement (construire nos applications). Ensuite, nous apprenons à naviguer sur les routes et à suivre notre itinéraire (utiliser Git et GitHub). Enfin, nous mettons la voiture en mode régulateur de vitesse pour rendre le trajet plus fluide et moins manuel (automatisation avec GitHub Actions). À la fin de ce voyage, nous serons des pros pour partager nos applications avec le monde, grâce à Kubernetes et à ces outils.

****La Magie des Conteneurs (alias Conteneurs Docker)****

Imagine que tu aies créé un jeu vidéo ou une application super cool. Maintenant, tu veux le partager avec tes amis pour qu'ils puissent y jouer sur leurs ordinateurs, tablettes ou téléphones. Mais il y a un hic : tous les appareils ne sont pas identiques. Certains amis peuvent avoir les derniers gadgets, tandis que d'autres pourraient utiliser des modèles plus anciens. C'est là que les **conteneurs Docker** entrent en jeu. Pense à un conteneur Docker comme une boîte magique qui contient tout ce dont ton application a besoin pour

fonctionner - comme le jeu lui-même, les instructions pour y jouer, et même l'environnement dont il a besoin pour fonctionner correctement. Peu importe l'appareil que tes amis ont, tant qu'ils peuvent ouvrir cette boîte magique, ils peuvent jouer à ton jeu.

****Automatiser la Magie avec Git****

Nous allons mettre en place un type spécial de sortilège magique dans Git (notre journal de code) qui emballe automatiquement notre jeu dans ces boîtes magiques chaque fois que nous apportons des modifications ou des améliorations. Ce sortilège s'appelle un **flux de travail d'intégration**. Après que le jeu soit emballé, nous **pousserons** (ce qui est juste une façon élégante de dire "télécharger") nos boîtes magiques vers un **registre de conteneurs**, une bibliothèque spéciale où ces boîtes sont stockées et peuvent être partagées.

****Pourquoi les Conteneurs sont Géniaux****

Utiliser ces boîtes magiques signifie que nous pouvons facilement déplacer notre jeu de notre ordinateur vers Internet ou l'appareil de n'importe qui d'autre sans craindre qu'il ne fonctionne pas correctement. C'est comme avoir un jeu qui peut fonctionner parfaitement dans une salle d'arcade, sur une console de salon, ou même sur un smartphone, sans avoir besoin de modifier quoi que ce soit à propos du jeu lui-même.

****Mais, Il y a un Revirement****

Aussi incroyables que soient ces conteneurs Docker, nous rencontrons un obstacle lorsque nous essayons de partager notre jeu avec beaucoup de personnes à la fois ou lorsque nous voulons mettre à jour le jeu avec de nouveaux niveaux ou fonctionnalités. Bien sûr, nous pouvons utiliser des outils comme **Docker Compose** et **Watchtower** pour jongler avec plus d'un conteneur à la fois, mais cela commence à ressembler à jongler avec trop de boîtes magiques en l'air. Les mettre à niveau et s'assurer que tout le monde puisse jouer sans de longues attentes (gérer le trafic ou la charge supplémentaires) devient un vrai casse-tête.

En plongeant dans le monde des conteneurs Docker, nous débloquons un nouveau niveau dans le partage de nos créations, nous assurant que tout le monde puisse en profiter, peu importe l'appareil qu'ils utilisent. Cependant, nous apprenons également qu'avec un grand pouvoir viennent de nouveaux défis, surtout lorsque nous essayons de partager notre jeu avec beaucoup de gens ou de le maintenir à jour avec du nouveau contenu.

Imagine que ton application ou ton jeu est comme un camion de crème glacée super populaire. Au fur et à mesure que le bouche-à-oreille se propage, de plus en plus de gens se présentent. Pour servir tout le monde, tu pourrais penser : "Ajoutons plus de camions

(calcul) pour gérer la foule (trafic)". C'est un plan solide, mais cela soulève une grande question : Utilisons-nous nos camions et nos réserves de crème glacée (applications basées sur des conteneurs et ressources) de la manière la plus intelligente possible ? Certains camions sont-ils surchargés tandis que d'autres restent inutilisés ? Sommes-nous à court de certains parfums rapidement tandis que d'autres sont à peine touchés ?

C'est là qu'intervient l'**orchestration automatisée des conteneurs**, agissant comme le meilleur gestionnaire de camions de crème glacée au monde, et Kubernetes est l'un des meilleurs noms dans ce domaine.

****Démarrage avec Kubernetes****

Commencer avec Kubernetes, surtout avec un service qui le gère pour vous, c'est un peu comme apprendre les bases de la conduite d'un camion de crème glacée. C'est assez simple. Mais ensuite, imagine que tu veuilles offrir plus que de la crème glacée. Peut-être que tu veux vendre des smoothies, du café, ou même avoir un petit groupe de musique live. Soudain, tu ne gères plus seulement des camions ; tu coordonnes toute une flotte de différents services, dont certains pourraient avoir besoin d'être plus "visibles" (comme le groupe de musique live) ou d'interagir directement avec tes clients, et certains qui travaillent en coulisses (comme l'équipe de réapprovisionnement et de maintenance des camions).

****Ce que Ce Livre Propose****

Ce guide est comme votre système de navigation à travers les rues animées du déploiement et de la gestion efficace de votre application ou jeu. Il est là pour vous aider à comprendre comment gérer la foule croissante d'utilisateurs en douceur, vous assurer que chaque partie de votre service fonctionne parfaitement, et comment élargir vos offres sans transformer votre opération en cauchemar logistique.

En essence, ce voyage à travers Kubernetes et l'orchestration de conteneurs vise à garantir que votre application ou jeu peut gérer la célébrité et la fortune sans broncher, en utilisant chaque ressource à sa disposition de la manière la plus efficace possible.

****1.3 Naviguer dans le Labyrinthe des Dépendances****

Imaginez la création d'une application ou d'un jeu comme la préparation d'une méga pizza pour vous et vos amis. Vous avez vos ingrédients de base : la pâte (votre langage de programmation principal comme Python ou JavaScript), la sauce (le code principal qui fait fonctionner votre application), et le fromage (l'interface utilisateur de base). Mais tout le monde veut des garnitures différentes (fonctionnalités), et c'est là que ça se complique.

Tout comme en cuisine, en codage, vous avez souvent besoin de mélanger différents ingrédients (ou **dépendances**) pour obtenir le résultat souhaité.

Dépendances ? Qu'est-ce que c'est ?

Les dépendances dans les logiciels sont toutes les petites choses supplémentaires dont vous avez besoin pour que votre application ou votre jeu fonctionne :

- **Langages de Programmation :** Les énumérer simplement - Python, JavaScript, Java, et ainsi de suite - revient à choisir entre le pepperoni, les champignons ou les ananas comme garnitures. Chacun apporte une saveur différente (fonctionnalité) à la table.

- **Est-ce que ça Fonctionnera sur Mon Appareil ?** C'est un peu comme demander si votre ami intolérant au gluten peut manger la pâte à pizza. Certaines applications (pizzas) ne fonctionneront pas sur certains appareils (restrictions alimentaires) car elles ne sont pas compatibles.

- **De Quelle Version Parle-t-on ?** Utiliser une sauce tomate périmée peut ruiner la pizza. De même, utiliser une version obsolète d'un langage de programmation peut perturber votre application.

- **Où Trouver Cela ?** Savoir où acheter le meilleur pepperoni revient à savoir où télécharger les bons outils ou langages pour votre application.

- **Stable ou Bêta ?** Choisiriez-vous une mozzarella éprouvée ou un nouveau fromage expérimental ? C'est la même chose avec les versions logicielles.

- **Types d'Applications :** Votre application est-elle une application web, un processus en arrière-plan, ou une application native ? C'est comme demander si vous préparez une pizza à croûte mince, une pizza à pâte épaisse, ou quelque chose d'entièrement différent.

- **Paquets de Tiers :** Ceux-ci sont comme des épices spéciales ou des garnitures supplémentaires que vous ajoutez pour faire ressortir votre application. Mais vous devez vous assurer qu'ils se marient bien avec vos ingrédients principaux.

- **Open Source vs Payant :** C'est comme choisir entre faire votre pizza à partir de zéro ou en commander une. Certains outils sont gratuits (open source), tandis que d'autres doivent être achetés (sous licence).

- **Systèmes d'Exploitation :** Votre pizza sera-t-elle cuite dans un four à bois ou un four de cuisine ordinaire ? De même, votre application fonctionnera-t-elle sous Windows, macOS, Linux, ou autre chose ?

****Pourquoi Est-ce Important ?****

Chaque question sur les dépendances est cruciale car elles déterminent si votre application fonctionnera comme prévu, sera appréciée par vos utilisateurs et sera accessible à tous ceux avec qui vous souhaitez la partager. Obtenir ces dépendances correctement, c'est comme s'assurer que vous avez les bons ingrédients, en bonne quantité, prêts à être utilisés.

Comprendre les dépendances consiste à s'assurer que tout ce dont votre application ou jeu a besoin pour fonctionner correctement est en place, des langages de programmation que vous choisissez aux plateformes sur lesquelles il fonctionnera. Tout comme s'assurer que vous avez toutes les bonnes garnitures pour votre méga fête de pizza !

Lors de la création de logiciels, les développeurs sont comme des chefs qui tentent de préparer un repas gastronomique. Ils ont besoin d'une multitude d'ingrédients (dépendances) pour s'assurer que leur création soit parfaite. Ces ingrédients peuvent varier considérablement, en fonction de ce qu'ils concoctent. Analysons les types d'ingrédients (dépendances) dont ils pourraient avoir besoin :

- **Système d'exploitation :** C'est comme avoir besoin d'un type spécifique de four pour cuire un gâteau. Par exemple, les jeux conçus pour la Xbox One nécessitent cette console pour fonctionner, tout comme certains gâteaux nécessitent un four à convection pour être parfaits.

- **Spécifique à l'application :** Imaginez que vous prépariez une pizza qui nécessite une marque spécifique de sauce pour avoir bon goût. De même, les applications web FastAPI nécessitent Python pour fonctionner car elles sont conçues pour fonctionner avec cette "sauce".

- **Spécifique à la version :** C'est comme avoir besoin de la récolte 2021 d'un vin pour son goût. Par exemple, Django 4.1 nécessite Python 3.8 ou plus récent. Utiliser Python 2.7 serait comme utiliser un vin de 1995 lorsque la recette demande un millésime 2021, ça ne sera tout simplement pas bon.

- **Dépendances tierces :** Pensez à cela comme ayant besoin d'une épice spéciale que vous ne pouvez obtenir que dans un magasin particulier. Par exemple, le paquet Python MoviePy nécessite que FFmpeg soit installé sur votre ordinateur pour fonctionner. C'est comme avoir besoin de safran pour rendre votre plat parfait, mais vous ne pouvez pas le trouver dans n'importe quel supermarché.

- **Matériel Spécifique :** Cela équivaut à avoir besoin d'appareils de cuisine spécifiques pour cuisiner. Par exemple, exécuter des modèles d'apprentissage profond avec PyTorch nécessite souvent un GPU (unité de traitement graphique), tout comme préparer des légumes en spirale peut nécessiter un spiraliseur. Et tout comme vous avez besoin de l'électricité, d'un câble réseau branché et du routeur fonctionnant pour un jeu en ligne, vous avez besoin des bons composants informatiques (CPU et RAM) et d'autres matériels pour exécuter des logiciels complexes.

Ainsi, lorsque les développeurs préparent leur logiciel, ils ne se contentent pas de taper sur leur ordinateur. Ils s'assurent d'avoir tous les bons ingrédients, du bon "four" du système d'exploitation à la version exacte du "vin" dont ils ont besoin, et même des "épices" spéciales provenant d'un magasin tiers. Et bien sûr, ils ont besoin des bons "appareils de cuisine" pour donner vie à tout cela.

Imaginez que vous avez un jeu Xbox One que vous adorez, mais vous aimeriez pouvoir y jouer sur votre iPhone lors d'un long trajet en voiture. Vous pourriez penser : "Eh bien, pourquoi ne pas simplement connecter un lecteur Blu-ray à mon iPhone et insérer le disque de jeu ?" Mais ce n'est pas si simple. C'est comme essayer d'utiliser un disque PlayStation dans un grille-pain ; ils ne sont tout simplement pas conçus pour fonctionner ensemble. Même s'il existe une version du jeu sur l'App Store, elle a été spécialement adaptée (beaucoup modifiée) pour fonctionner sur un iPhone. Et puis, bien sûr, vous devez également tenir compte du modèle d'iPhone que vous possédez, car tous les jeux ne fonctionnent pas sur tous les modèles.

****Pourquoi Est-ce Important ?****

Toute cette situation n'est pas un accident. Elle est mise en place de cette manière délibérée pour vous maintenir dans un certain écosystème, que ce soit celui d'Apple, de Microsoft, ou de quelqu'un d'autre. Cela s'appelle le **verrouillage fournisseur**, et c'est comme si on vous disait que vous ne pouvez utiliser que des ingrédients de marque spécifique dans vos appareils de cuisine. Cette configuration peut avoir une influence énorme sur le succès ou l'échec d'un logiciel (ou d'un jeu), selon la facilité avec laquelle les gens peuvent y accéder et l'utiliser.

La Bonne Chose à Propos de l'Open Source

Cependant, il existe un monde là-bas qui ressemble à un immense jardin communautaire, où tout le monde est libre de planter, de modifier et de récolter à sa guise. C'est le monde du logiciel **open source**, auquel appartient Kubernetes et de nombreux autres outils dont nous parlerons dans ce livre. Open source signifie que le logiciel est librement

disponible pour quiconque souhaite l'utiliser, le modifier ou le partager. Il n'y a pas de verrouillage fournisseur ici ; c'est comme pouvoir utiliser n'importe quel ingrédient dans votre cuisine, indépendamment de la marque ou de l'endroit où vous l'avez acheté.

Les projets open source sont tous basés sur la liberté et la flexibilité. Ils sont conçus pour être partagés et améliorés par n'importe qui, ce qui signifie qu'ils ne vous contraignent pas à les utiliser d'une seule manière sur un seul appareil spécifique. C'est comme avoir une recette que chacun peut ajuster à son goût et à ses besoins alimentaires, plutôt qu'un repas pré-emballé que vous ne pouvez réchauffer que dans un micro-ondes spécifique.

Ainsi, bien que vous ne puissiez pas facilement jouer à votre jeu Xbox sur votre iPhone en raison de toutes ces restrictions et de problèmes de compatibilité, le monde open source offre une alternative rafraîchissante. Il invite tout le monde à contribuer, modifier et utiliser des logiciels de manière qui convient le mieux à leurs besoins, sans les enfermer dans une seule façon de faire les choses.

****1.4 Les Superpouvoirs des Conteneurs****

Imaginez que vous avez créé un jeu vidéo épique, et maintenant vous voulez le partager avec tout le monde, partout. Mais il y a un hic : votre jeu a besoin de certaines choses pour fonctionner correctement, comme des logiciels ou des outils spécifiques, et tous les ordinateurs ou téléphones ne disposent pas de ces éléments. C'est là que les **conteneurs** entrent en jeu, agissant comme un super-héros pour votre jeu.

****Le Rôle des Conteneurs :****

Imaginez un conteneur comme un sac à dos magique pour votre jeu. Ce sac à dos peut contenir votre jeu et tout ce dont il a besoin pour fonctionner, comme des outils spéciaux ou des logiciels. La partie magique ? Une fois que votre jeu est dans ce sac à dos, il peut être facilement déplacé de votre ordinateur vers l'ordinateur portable d'un ami, ou même vers un grand serveur sur Internet, et il fonctionnera parfaitement à chaque fois, peu importe où il va.

****Voici Comment Ça Fonctionne :****

- **FROM python:3.10** : Cette ligne équivaut à dire : "Nous allons utiliser Python 3.10 comme base."

- **COPY . /app** : Cela signifie "Prenez tout ce dont mon jeu a besoin et mettez-le dans un dossier appelé 'app'."

- **WORKDIR /app** : C'est comme dire : "Ouvrez le dossier 'app' ; c'est là que nous allons travailler."

- **RUN python3 -m pip install pip --upgrade** : Ici, nous nous assurons d'avoir l'outil le plus récent pour installer d'autres outils dont nous avons besoin.

- **RUN python3 -m pip install --no-cache-dir -r requirements.txt** : Cela revient à rassembler tous les outils spéciaux et logiciels dont votre jeu a besoin en fonction d'une liste que vous avez faite.

- **CMD ["gunicorn", "--bind", "0.0.0.0:8000", "main:app"]** : Enfin, cette ligne indique à l'ordinateur comment démarrer votre jeu pour que les gens puissent y jouer.

****La Magie en Moins de 10 Lignes :****

Avec seulement quelques lignes dans un Dockerfile (une sorte de liste spéciale pour le conteneur), vous pouvez empaqueter votre jeu et tous ses besoins dans un sac à dos magique portable. Cela rend super facile de partager et d'exécuter votre jeu sur presque n'importe quel appareil. Peu importe si vous avez créé le jeu en Python, en JavaScript, ou dans n'importe quel autre langage de programmation ; le conteneur s'en charge.

Ainsi, les conteneurs sont comme avoir un sac à dos magique qui garantit que votre jeu ou application peut voyager n'importe où et fonctionner parfaitement sur n'importe quel appareil, rendant le partage et le déploiement de vos créations un jeu d'enfant.

De l'idée à la diffusion de votre application partout avec les conteneurs

Donc, vous avez cette idée d'application géniale, et vous avez griffonné comment vous voulez qu'elle fonctionne dans un Dockerfile (pensez-y comme une recette pour votre application). Ce qui vient ensuite est assez simple.

****Transformer la recette en une vraie application :****

1. **Construction de l'application :** Vous utilisez un outil appelé un **constructeur de conteneurs** (imaginez un four super avancé pour les applications) pour transformer votre recette (le Dockerfile) en quelque chose de réel - une **image de conteneur**. Cette image est comme une version lyophilisée de votre application ; elle contient tout ce dont votre application a besoin pour prendre vie, attendant simplement le bon environnement.

2. **Partage ou exécution de votre application :**

- Si vous voulez partager votre application avec le monde, vous téléchargez l'image de conteneur dans un **registre de conteneurs**, un peu comme la poster dans une bibliothèque cloud où d'autres peuvent la télécharger.

- Pour l'exécuter, vous utilisez un **moteur de conteneurs**, qui est comme un dispositif magique capable de lire votre application lyophilisée et de la faire vivre sur n'importe quel ordinateur ou serveur.

Docker : Le commencement de la magie des conteneurs

Quand nous parlons de création, de partage ou d'exécution de ces images de conteneurs, beaucoup d'entre nous disent simplement "Docker", car ils ont été les pionniers de cette magie. Mais Docker n'est pas seulement la magie - c'est aussi le nom de l'entreprise qui l'a inventée ! Cependant, la recette secrète derrière Docker est maintenant partagée avec le monde sous forme **open-source**. Cela signifie que n'importe qui peut l'utiliser, le modifier et l'améliorer, faisant de Docker juste l'une des nombreuses façons de travailler avec les conteneurs maintenant.

Quel type de magie les conteneurs peuvent-ils faire ?

Pensez à toutes les applications et services que vous utilisez - il y a de fortes chances qu'ils fonctionnent dans des conteneurs. Voici quelques exemples :

- **Langages de programmation :** Applications écrites dans des langages comme Python, Ruby, Node.js (JavaScript), Go, Rust et PHP.

- **Serveurs Web :** La technologie qui diffuse les sites Web, comme NGINX et HTTPD.

- **Bases de données :** Les cerveaux où les applications stockent et récupèrent des données, comme MySQL, PostgreSQL, MongoDB.

- **Files d'attente de messages :** Systèmes qui aident les différentes parties d'une application à communiquer entre elles de manière fluide, comme RabbitMQ et Kafka.

Les conteneurs sont comme des enveloppes magiques, vous permettant d'emballer votre application et tous ses secrets afin qu'elle puisse fonctionner n'importe où - de votre téléphone à une immense ferme de serveurs. Docker a commencé cette magie, mais

maintenant le grimoire est ouvert à tous pour l'utiliser et l'améliorer, alimentant presque tout dans le monde numérique.

Pourquoi les conteneurs et le logiciel open source sont comme le beurre de cacahuète et la gelée

Imaginez que vous préparez un déjeuner pour une grande aventure - comme coder votre propre application ou lancer un site Web. Dans ce monde, les **conteneurs** sont comme votre boîte à lunch, vous permettant de transporter facilement votre repas (ou votre logiciel) partout où vous voulez aller.

Le logiciel open source = le sandwich parfait

Maintenant, si les ingrédients de votre sandwich (le logiciel que vous utilisez) sont **open source**, cela signifie que tout le monde peut partager, utiliser et ajuster la recette. C'est comme si vous aviez un approvisionnement infini de beurre de cacahuète et de gelée à votre disposition pour faire vos sandwichs. Le logiciel open source est génial pour les conteneurs car c'est comme si ces ingrédients étaient faits pour s'adapter parfaitement dans votre boîte à lunch, vous permettant de les emballer et de les déballer où que vous alliez dans votre aventure de codage.

Intégrer Kubernetes dans le Mélange

Une fois que vous avez pris le coup de main pour utiliser votre boîte à lunch (les conteneurs), vous êtes prêt à passer au niveau supérieur en organisant un pique-nique complet pour vous et vos amis, et c'est là que **Kubernetes** intervient. Kubernetes est comme un panier de pique-nique magique qui non seulement contient toutes vos boîtes à lunch, mais qui sait aussi exactement comment installer votre aire de pique-nique, gérer qui obtient quoi à manger, et même nettoyer une fois que vous avez terminé. Donc, si vous savez utiliser une boîte à lunch (conteneurs), vous pouvez certainement gérer un panier de pique-nique magique (Kubernetes) pour rendre votre aventure encore plus épique.

En essence, le voyage à travers la programmation avec des conteneurs et des logiciels open source est comme se préparer pour le plus grand pique-nique jamais organisé. Une fois que vous êtes à l'aise avec le fait de préparer votre boîte à lunch, passer à la gestion de tout le pique-nique avec Kubernetes semblera être une étape suivante tout à fait naturelle.

1.5 Pourquoi il est difficile de créer et d'utiliser des conteneurs sans Kubernetes

Imaginez la création d'un conteneur pour votre application comme la construction de votre propre PC de jeu sur mesure. Commencer la construction - choisir vos pièces (ou rédiger le Dockerfile pour votre conteneur) - peut être simple si vous savez ce que vous faites. Mais tout comme avec un PC de jeu, la partie délicate est de s'assurer que toutes les pièces fonctionnent parfaitement ensemble.

Trouver les bonnes pièces (Dépendances) :

Avant même de commencer à assembler les choses, vous devez comprendre toutes les pièces (dépendances) dont vous avez besoin. Il ne s'agit pas seulement du matériel (comme les éléments au niveau du système d'exploitation), mais aussi du logiciel (les éléments au niveau de l'application) qui permet à votre jeu (ou application) de fonctionner correctement. Parfois, comprendre ces éléments peut être plus complexe que vous ne le pensez.

Rédiger les instructions (Dockerfile) :

Une fois que vous avez toutes vos pièces en main, vous rédigez les instructions (le Dockerfile) sur la manière de tout assembler. Cela peut être très simple, comme simplement installer un jeu à partir d'un disque, ou très compliqué, impliquant des paramètres personnalisés, des pilotes spéciaux, et plus encore. Tout comme pour la construction d'un PC, la complexité peut varier énormément.

Lorsque les choses ne se déroulent pas comme prévu :

C'est là que les choses deviennent un peu stressantes. Vous pourriez suivre toutes les instructions parfaitement, mais lorsque vous essayez de démarrer votre PC de jeu (ou de construire et exécuter votre conteneur), les choses peuvent encore mal tourner. Peut-être que le jeu plante au démarrage, ou que le système surchauffe à cause d'un ventilateur défectueux (un bogue dans votre conteneur). Parfois, vous ne savez même pas qu'il y a un problème avant d'être complètement configuré et prêt à jouer (ou à déployer votre application). Et même s'il démarre bien, il pourrait y avoir des problèmes qui surviennent plus tard, comme des plantages aléatoires ou des problèmes de performance.

So, building and running containers without something like Kubernetes is a bit like embarking on a PC building project without an expert to guide you. It might seem fun and

straightforward at first, but without a good understanding and some help to manage the complexities, you could end up with a flashy setup that doesn't work as expected.

Naviguer dans le chaos des conteneurs sans Kubernetes

Imaginez lancer une nouvelle version de votre jeu vidéo multijoueur préféré en ligne. Dans un monde idéal, cette mise à jour se déroule sans accroc, et tout le monde peut jouer à la nouvelle version instantanément. Mais la réalité a souvent d'autres plans, et tout comme dans le jeu, le déploiement des applications peut devenir assez chaotique.

Pourquoi les mises à jour ne se déroulent pas toujours sans accroc :

Imaginons que vous vouliez mettre à jour le jeu sur votre serveur (comme mettre à jour le code dans un conteneur). Idéalement, vous feriez cela sans que personne ne le remarque, en maintenant le jeu en ligne. Mais souvent, vous pourriez devoir mettre en pause, en expulsant temporairement les joueurs (indisponibilité), pour mettre cette mise à jour en place. C'est comme devoir arrêter le serveur pour une pause de maintenance - nécessaire, mais ennuyeux pour les joueurs qui veulent juste continuer à jouer.

Obstacles courants avec les conteneurs (sans Kubernetes) :

1. **Mise à niveau des versions des conteneurs :** Imaginez avoir besoin de mettre à niveau votre serveur de jeu pour gérer plus de joueurs ou corriger des bugs. Faire cela en douceur sans quelque chose comme Kubernetes, c'est comme essayer de changer le moteur d'un avion en plein vol.

2. **Gestion de plusieurs conteneurs :** Faire tourner plus de deux ou trois conteneurs en même temps, c'est comme essayer de jouer à plusieurs jeux vidéo sur différents appareils en même temps. Sans aide, c'est écrasant.

3. **Mise à l'échelle vers le haut ou vers le bas :** Si votre jeu devient soudainement très populaire, vous avez besoin de plus de serveurs (conteneurs) pour que tout fonctionne correctement. C'est comme avoir besoin de plus de bus pour une augmentation surprise du nombre de passagers sur une ligne de bus. Le faire manuellement ? Bonne chance.

4. **Rebondir après des échecs :** Parfois, les serveurs tombent en panne. Sans un système comme Kubernetes, la récupération est comme essayer de réparer un pneu crevé avec la voiture en mouvement. C'est possible, mais c'est beaucoup plus difficile.

5. **Communication entre les conteneurs :** Imaginez si dans un jeu multijoueur, les joueurs ne pouvaient pas communiquer entre eux. Les conteneurs sont confrontés à un défi similaire sans moyen de gérer leurs interactions.

Kubernetes à la rescousse :

Tous ces défis - effectuer des mises à jour sans interruption, gérer de nombreux conteneurs, ajuster rapidement l'échelle vers le haut ou vers le bas, récupérer après des incidents, et garantir que les conteneurs peuvent "communiquer" entre eux - sont des domaines où Kubernetes excelle. C'est comme avoir un administrateur de jeu ultra-intelligent qui peut gérer tous les serveurs, les demandes des joueurs et les mises à jour sans le moindre effort.

Tout au long de ce livre, nous explorons comment Kubernetes transforme le chaos potentiel en une navigation fluide, veillant à ce que vos applications fonctionnent comme des machines bien huilées, prêtes à s'adapter à un tournoi mondial ou à se réduire pour une soirée tranquille de quêtes en solo.

1.6 Naviguer dans les Défis de Kubernetes

Kubernetes peut sembler être un outil de super-héros pour le déploiement et la gestion des applications, mais même les super-héros ont leur kryptonite. Regardons de plus près certains des aspects délicats de l'utilisation de Kubernetes, un peu comme apprendre à conduire une voiture super avancée.

1. Installation et Mise à Jour :

Mettre en place Kubernetes et le maintenir à jour, c'est un peu comme essayer d'installer un jeu vidéo haut de gamme sur un vieil ordinateur. Cela peut être un processus complexe et frustrant, nécessitant des connaissances spécifiques et de la patience.

2. Surcharge de Configuration :

Kubernetes peut être personnalisé de tant de manières, ce qui est génial mais aussi écrasant. C'est un peu comme se voir remettre le panneau de contrôle d'un vaisseau spatial. Il y a tellement de boutons et de leviers que comprendre ce que chacun fait est déjà un défi en soi.

3. Applications avec état vs Applications sans état :

Dans le monde des applications, certaines doivent se souvenir des actions passées (avec état), tandis que d'autres non (sans état). Gérer ces différents types d'applications avec Kubernetes peut être délicat. Imaginez essayer d'équilibrer un régime alimentaire qui doit changer quotidiennement (avec état) par rapport à une routine d'entraînement fixe (sans état).

4. Sécurité et Permissions :

Veiller à ce que seules les bonnes personnes aient accès pour effectuer des tâches spécifiques dans Kubernetes est essentiel mais compliqué. C'est semblable à mettre en place un système de sécurité pour une maison, en décidant qui obtient les clés et à quelles portes.

5. Communication entre Conteneurs :

Faire en sorte que les conteneurs communiquent efficacement entre eux est crucial mais pas toujours simple. Imaginez organiser un projet de groupe où tout le monde doit communiquer sans se parler par-dessus.

6. Stockage :

Les conteneurs sont excellents pour exécuter des applications, mais quand il s'agit de stocker des données, les choses se compliquent. C'est un peu comme décider si vous devez garder vos fichiers sur une clé USB, dans le cloud ou sur votre disque dur.

7. Récupération et Débogage :

Quand les choses tournent mal, comprendre le problème et le corriger peut être difficile. C'est comme essayer de trouver une aiguille dans une botte de foin, surtout si vous ne savez pas où chercher.

Le Résultat ?

En raison de ces défis, certains développeurs pourraient commencer à chercher des moyens plus simples de déployer leurs applications. C'est un peu comme décider s'il faut apprendre à conduire cette voiture super avancée avec toutes ses complexités ou simplement rester fidèle à un vélo qui vous emmène là où vous devez aller sans tracas.

Alors que Kubernetes offre des outils puissants pour gérer les applications à grande échelle, le maîtriser comporte son propre ensemble de défis. C'est un peu comme apprendre à jouer d'un nouvel instrument ; cela nécessite beaucoup de pratique et d'apprentissage avant de pouvoir produire une belle musique.

Plongée dans Kubernetes : Un Guide pour Débutants

Kubernetes : L'Entraîneur d'Équipe pour les Conteneurs de Votre Application

Imaginez que vous jouiez à un jeu vidéo comme "Minecraft" ou "Fortnite" au sein d'une équipe avec des amis. Dans ce scénario, Kubernetes est comme l'entraîneur qui décide qui joue quel rôle, où et quand. Il organise votre équipe (les conteneurs) à travers différents serveurs de jeu (machines virtuelles) de manière à ce que cela soit le plus logique pour gagner la partie (faire fonctionner votre application en douceur).

Choisir Kubernetes Géré : Le Bouton Facile

Configurer et mettre à jour Kubernetes par vous-même, c'est un peu comme essayer de construire un PC de jeu à partir de zéro lorsque vous ne l'avez jamais fait auparavant. C'est faisable, mais pourquoi se compliquer la tâche alors que vous pouvez obtenir un modèle pré-construit ? C'est pourquoi nous utiliserons Kubernetes géré dans ce guide - c'est comme obtenir un PC de jeu haut de gamme prêt à l'emploi dès la sortie de la boîte.

Le Jeu de Configuration : Configuration de Votre Application

Même avec Kubernetes configuré, faire fonctionner votre application est un peu comme entrer dans un nouveau niveau de jeu avec ses propres règles. Vous devez vous occuper d'un tas de paramètres (configuration) juste pour faire fonctionner une seule application web. Et lorsque vous voulez mettre à jour votre application, c'est un peu comme apprendre de nouveaux mouvements (implémenter des outils CI/CD et RBAC) pour naviguer et apporter des modifications sans obtenir de "game over".

Avec État vs Sans État : Tu Te Souviens de Moi ?

Les conteneurs n'ont généralement pas de "mémoire" (sans état), comme un personnage de jeu qui se réinitialise à chaque fois que vous jouez. Mais certaines applications doivent se souvenir de choses (avec état), comme un jeu où votre progression est sauvegardée. Mélanger les deux peut être délicat, comme jouer à un jeu qui n'a pas été conçu pour sauvegarder votre progression sur une console qui le peut.

Conteneurs Qui Parlent : Est-ce que tu M'entends ?

Faire en sorte que vos conteneurs communiquent entre eux est un autre casse-tête. C'est un peu comme être dans un jeu multijoueur où vous devez comprendre comment parler efficacement à vos coéquipiers sans que tout le monde crie par-dessus les autres.

Oops... Comment Réparons-Nous Cela ?

Enfin, lorsque les choses tournent mal (et elles le feront), comprendre ce qui s'est passé et comment le réparer revient à essayer de résoudre un mystère dans un jeu sans aucun indice. Cela peut être frustrant, mais c'est tout un voyage.

Tout au long de ce guide, nous aborderons ces défis de front, comme passer des niveaux dans un jeu vidéo. Chaque étape nous rapprochera de la maîtrise de Kubernetes, prêts à déployer n'importe quelle application en toute confiance, tout comme vous améliorez vos compétences de jeu en montant de niveau.

1.7 Le Plan de Jeu pour Apprendre dans ce Livre

Considérez ce livre comme votre manuel personnel d'entraînement pour maîtriser l'art du déploiement d'applications, un peu comme apprendre à jouer à un nouveau sport ou jeu vidéo. Nous allons commencer par les bases et monter en compétence au fur et à mesure,

en utilisant deux projets principaux - une application Python et une application JavaScript - comme terrain d'entraînement.

Niveau 1 : Déploiement manuel

C'est comme apprendre à faire du vélo avec des roues d'entraînement. Nous commencerons par configurer manuellement nos applications à l'ancienne. C'est pratique et simple : vous pouvez voir et contrôler chaque étape du processus, comprendre les détails de la mise en place de votre application depuis votre ordinateur jusqu'à sa diffusion dans le monde.

Niveau 2 : Déploiement de conteneurs

Maintenant, imaginez passer à un skateboard. C'est plus rapide et plus cool, mais cela demande un peu plus de compétence. C'est là que les conteneurs entrent en jeu. Nous emballerons nos applications dans des conteneurs, ce qui les rend très portables et faciles à exécuter n'importe où. C'est une façon élégante de déployer des applications sans se soucier de tous les détails de l'environnement dans lequel elles seront exécutées.

Niveau 3 : Déploiement Kubernetes

Enfin, c'est comme passer à un réacteur dorsal. Kubernetes fera passer nos compétences en déploiement d'applications au niveau supérieur, en gérant et en automatisant le processus sur de nombreuses machines. C'est puissant et nous permet de gérer des applications plus complexes et évolutives sans nous épuiser. En nous appuyant sur ce que nous avons appris avec les déploiements manuel et de conteneurs, utiliser Kubernetes se sentira comme débloquer un superpouvoir.

En empilant ces compétences, vous apprendrez non seulement le comment, mais aussi le pourquoi de chaque méthode, vous offrant une boîte à outils prête pour tout défi de déploiement que vous pourriez rencontrer à l'avenir. Considérez cela comme une évolution d'un joueur novice à un professionnel aguerri, équipé des connaissances nécessaires pour déployer des applications de la manière la plus efficace et la plus efficace possible.

Comprendre Kubernetes : Comme Apprendre à Conduire Avant de Passer à l'Automatique

Se lancer directement dans l'utilisation de Kubernetes pour le déploiement d'applications sans saisir les bases, c'est comme essayer de conduire une voiture automatique de haute technologie sans jamais avoir appris les fondamentaux de la conduite. Bien sûr, la voiture peut faire beaucoup pour vous, mais que se passe-t-il lorsque quelque chose ne va pas ?

C'est pourquoi commencer par des déploiements manuels, c'est comme conduire une voiture avec une boîte de vitesses manuelle : vous avez une véritable sensation de ce qui se passe sous le capot. Cela vous apprend le pourquoi et le comment de l'automatisation du déploiement d'applications, donc lorsque les choses tournent mal (et elles le feront), vous n'êtes pas laissé perplexe.

Pourquoi Python et JavaScript ?

Nous nous concentrons sur Python et JavaScript car ce sont un peu les Coca-Cola et Pepsi des langages de programmation - largement connus et utilisés par de nombreux développeurs (y compris l'auteur). Mais hé, si vous préférez un autre langage, ne vous inquiétez pas. Sur notre page GitHub, nous avons également des démonstrations dans d'autres langages, montrant qu'une fois que votre application est prête pour un conteneur, le monde du déploiement vous appartient, que ce soit de manière manuelle, conteneurisée ou avec Kubernetes.

Approfondissement de Kubernetes

La seconde moitié de ce livre ressemble à des leçons de conduite avancées pour Kubernetes. Vous avez appris à conduire ; maintenant, il est temps de naviguer sur les autoroutes et les routes secondaires du déploiement d'applications. Nous explorerons comment maintenir votre déploiement Kubernetes en marche sans heurts, en intégrant de nouvelles fonctionnalités et applications sans causer d'embouteillages. Tout est question de s'assurer que votre déploiement peut croître et changer aussi facilement que la publication d'une mise à jour sur les réseaux sociaux.

À la fin de ce voyage, vous comprendrez non seulement comment déployer des applications avec Kubernetes, mais aussi comment adapter et faire évoluer votre stratégie de déploiement au fil du temps, en veillant à ce que vos applications restent à jour et prêtes à répondre aux demandes de leurs utilisateurs, tout comme on garde une voiture bien réglée et prête pour la route.