DevOps

Initialisation à la culture DevOps

Référence : EF-TEST-TEST

Auteur(s) :
Yann BENHAMRON

Destinataire(s) :
Easyformer

Date de modification : 14/09/23

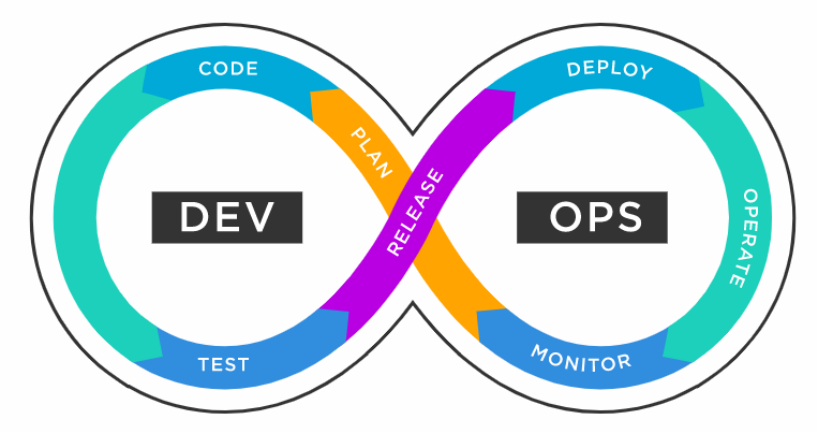
Version : 1

Sommaire

page

1 Introduction à la culture DevOps

1.1 Définition



DevOps est une culture et une méthode de développement logiciel qui se concentre sur la collaboration, l'automatisation et l'amélioration continue entre les équipes de développement et d'opérations. L'objectif principal de DevOps est de réduire le temps de mise sur le marché d'un produit tout en améliorant sa qualité.

L'histoire de DevOps remonte au début des années 2000, lorsque les entreprises ont commencé à adopter des méthodologies de développement Agile et Lean pour accélérer le processus de développement.

Les méthodologies de développement Agile sont une façon plus flexible et collaborative de créer des logiciels, qui est apparue pour répondre aux limitations des méthodes traditionnelles. Au lieu de planifier tout le projet à l'avance, les équipes travaillent ensemble de manière itérative et incrémentale pour développer et tester des fonctionnalités tout au long du processus. Les pratiques courantes incluent la planification de sprint, le développement en tandem, le test continu, l'intégration continue et la livraison continue. Les méthodes Agile aident les équipes à s'adapter aux changements, à travailler plus rapidement et à créer des logiciels de meilleure qualité.

Cependant, les équipes d'opérations n'ont pas réussi à suivre le rythme et ont souvent rencontré des problèmes de déploiement, de gestion de l'infrastructure et de maintenance.

En réponse à ces défis, des pionniers de DevOps tels qu'Andrew Shafer, Patrick Devois, Jez Humble et Dave Farley ont commencé à promouvoir la collaboration entre les équipes de développement et d'opérations en adoptant des pratiques tels que :

- **L'infrastructure en tant que code (IaC)** : c'est une pratique qui permet de gérer l'infrastructure informatique de manière programmable, en utilisant des fichiers de code source. Cette pratique est courante en DevOps, elle permet de réduire les erreurs manuelles, d'accélérer les déploiements et d'améliorer la qualité globale du système. Les outils couramment utilisés pour l'IaC sont Terraform, Ansible, Chef et Puppet.
- **La livraison continue(CD)** : c'est une pratique qui permet aux équipes de développement de publier rapidement des mises à jour logicielles. Elle permet une intégration et des tests automatisés, ce qui permet de détecter les erreurs rapidement. La livraison continue accélère le processus de publication et améliore la qualité globale du produit final.
- **L'intégration continue (CI)** : L'intégration continue est une pratique de développement logiciel qui consiste à intégrer régulièrement les modifications du code à une base de code commune. Cette pratique permet de détecter rapidement les erreurs et les conflits de code. L'objectif est d'améliorer la qualité du logiciel en détectant les erreurs plus tôt dans le processus de développement et en garantissant que le code est régulièrement testé et intégré.

1.2 Les Objectifs du DevOps

Aujourd'hui, DevOps est largement adopté par les entreprises de toutes tailles et est considéré comme une étape importante pour atteindre l'agilité et la compétitivité sur le marché.

En 2020, la taille du marché mondial de DevOps était estimée à 4 311,95 millions USD. Cette estimation a été revue à la hausse pour 2021, atteignant 5 114,57 millions USD. Les prévisions pour 2022 sont plus incertaines, avec une fourchette allant de 2,9 à 8 milliards de dollars. Cependant, avec un taux de croissance annuel composé de 18,95 %, le marché devrait atteindre 12 215,54 millions USD d'ici 2026.

Voici quelques statistiques supplémentaires sur l'adoption de DevOps :

- En 2020, 49 % des entreprises ont signalé une réduction du temps de mise sur le marché des logiciels et des services.
- En 2021, 83 % des décideurs informatiques ont mis en place des pratiques DevOps pour obtenir une plus grande valeur commerciale.

Le DevOps est une approche de développement logiciel qui met l'accent sur la collaboration entre les équipes de développement, de tests et d'exploitation. Les objectifs du DevOps sont multiples et variés, mais ils ont tous pour but d'améliorer la qualité, la rapidité et l'efficacité du développement et de l'exploitation des logiciels.

- **Collaboration accrue entre les équipes** : L'un des principaux objectifs du DevOps est d'améliorer la collaboration entre les équipes de développement, de tests et d'exploitation. Les équipes travaillent ensemble pour identifier les problèmes, résoudre les erreurs et planifier les déploiements. Cela permet de s'assurer que tout le monde est sur la même longueur d'onde et travaille vers un objectif commun.
- **Réduction des délais de mise en production** : Le DevOps vise à réduire les délais de mise en production des logiciels. Les équipes travaillent ensemble pour automatiser les tâches manuelles, réduire les erreurs et accélérer le processus de développement. Cela permet aux équipes de livrer des logiciels plus rapidement, avec une qualité supérieure.
- **Amélioration de la qualité du code et de la stabilité des applications** : Le DevOps vise à améliorer la qualité du code et la stabilité des applications. Les équipes de développement travaillent avec les équipes d'exploitation pour automatiser les tests, surveiller les performances et diagnostiquer les problèmes plus rapidement. Cela permet de détecter et de résoudre les problèmes avant qu'ils ne deviennent des problèmes majeurs.
- **Augmentation de la fréquence et de la fiabilité des déploiements de logiciels** : Le DevOps vise à augmenter la fréquence et la fiabilité des déploiements de logiciels. Les équipes travaillent ensemble pour automatiser le processus de déploiement, réduire les erreurs et augmenter la fréquence des déploiements. Cela permet aux équipes de livrer des logiciels plus souvent, avec moins de risques.
- **Réduction des coûts de développement et d'exploitation** : Le DevOps vise à réduire les coûts de développement et d'exploitation en automatisant les tâches manuelles, en améliorant l'efficacité et en réduisant les erreurs. Cela permet aux équipes de développer et de déployer des logiciels plus rapidement et à moindre coût.
- **Amélioration de la satisfaction des utilisateurs** : Le DevOps vise à améliorer la satisfaction des utilisateurs grâce à une expérience utilisateur améliorée et une meilleure disponibilité des applications. Les équipes travaillent ensemble pour s'assurer que les logiciels sont stables, fiables et répondent aux besoins des utilisateurs.
- **Facilitation de l'innovation continue** : Le DevOps vise à faciliter l'innovation continue en permettant des cycles de développement plus courts et en permettant une rétroaction rapide des utilisateurs. Les équipes peuvent rapidement expérimenter de nouvelles idées et fonctionnalités, et améliorer les logiciels en temps réel.

1.3 Les avantages de l'adoption de DevOps pour une entreprise

L'adoption de DevOps est devenue de plus en plus courante pour les entreprises cherchant à améliorer leur agilité, leur rapidité et leur qualité de développement logiciel. DevOps encourage une collaboration étroite entre les équipes de développement et d'opérations, ce qui permet de réduire les délais de développement, d'accélérer la livraison des produits et des fonctionnalités et d'améliorer la qualité et la fiabilité du logiciel. En outre, DevOps permet aux entreprises de réduire les coûts et d'améliorer leur efficacité globale. Dans cette perspective, examinons de plus près les avantages clés de l'adoption de DevOps pour une entreprise.

- **Livraison rapide des applications** : L'adoption de DevOps permet de livrer rapidement les applications grâce à une collaboration étroite entre les développeurs et les opérateurs système. Cela permet d'accélérer le déploiement des applications et de répondre plus rapidement aux besoins des clients.
- **Meilleure qualité des applications** : DevOps permet également d'améliorer la qualité des applications en intégrant des tests automatisés tout au long du processus de développement. Cela permet d'identifier rapidement les erreurs et les bogues, ce qui facilite leur correction.
- **Réduction des coûts** : DevOps permet de réduire les coûts de développement et d'exploitation en permettant une meilleure utilisation des ressources. Cela peut être réalisé en utilisant des conteneurs, des machines virtuelles, et en automatisant les tâches manuelles.
- **Meilleure collaboration** : DevOps favorise une meilleure collaboration entre les différentes équipes. Les développeurs et les opérateurs système travaillent ensemble, ce qui permet d'éviter les silos de communication et de partager plus efficacement les connaissances.

1.4 Qui sont les Devs et les Ops ?

Les Devs et les Ops sont deux équipes au sein d'une entreprise qui travaillent ensemble dans le cadre de la méthodologie DevOps. Les Devs (développeurs) sont responsables de la création et de la maintenance du code source des applications, tandis que les Ops (administrateurs système) sont responsables de la mise en production et de la maintenance des infrastructures de production :

- Les missions des Devs sont de développer de nouvelles fonctionnalités et de maintenir le code existant. Ils sont également chargés de tester le code et de s'assurer qu'il fonctionne correctement avant qu'il ne soit livré à l'équipe Ops pour le déploiement.
- Les missions des Ops sont de mettre en production les applications, de s'assurer que les infrastructures de production sont opérationnelles et de surveiller les performances pour détecter les éventuels problèmes. Ils sont également responsables de la maintenance des systèmes et de la résolution des problèmes techniques.

Les problématiques auxquelles sont confrontées les deux équipes sont souvent différentes. Les Devs peuvent être confrontés à des problèmes liés à la qualité du code, à la stabilité des applications et à la collaboration avec l'équipe Ops. Les Ops peuvent être confrontés à des problèmes liés à la gestion de l'infrastructure, à la mise en production des applications et à la coordination avec l'équipe Dev.

Prenons un exemple de problématique auxquelles sont confrontées les deux équipes Dev et Ops :

Lorsqu'une nouvelle fonctionnalité doit être ajoutée à une application, les Devs doivent s'assurer que le code fonctionne correctement en environnement de développement et de test. Les Ops doivent ensuite déployer la nouvelle version de l'application en environnement de production, tout en s'assurant que le déploiement n'impacte pas la disponibilité de l'application pour les utilisateurs. Les deux équipes doivent donc collaborer étroitement pour s'assurer que la nouvelle fonctionnalité est déployée avec succès et sans perturber la disponibilité de l'application.

Dans ce scénario, les Devs peuvent rencontrer des problématiques liées à la qualité du code, au respect des délais de développement et à la coordination avec l'équipe Ops pour s'assurer que la nouvelle fonctionnalité est déployée correctement. Les Ops, quant à eux, peuvent rencontrer des problématiques liées à la gestion de l'infrastructure de production, au déploiement de la nouvelle version de l'application et à la coordination avec l'équipe Dev pour s'assurer que le nouveau code est stable et fonctionne correctement en environnement de production.

La méthodologie DevOps vise à résoudre ces problématiques en encourageant la collaboration et la communication entre les équipes Dev et Ops, en automatisant les tâches répétitives et en améliorant la qualité des applications et de l'infrastructure.

1.5 Les pratiques et outils de devops

Pour résoudre les problèmes de collaboration entre les équipes Dev et Ops, il est possible d'utiliser des pratiques et des outils adaptés, tels que ceux proposés par la culture DevOps. Cette dernière encourage une collaboration étroite, une automatisation des processus, une mesure et une analyse des performances, et un partage de connaissances entre les équipes.

La culture DevOps s'articulent autour de quatre catégories

1. **Collaboration** : La collaboration est un élément clé de la culture DevOps, car elle favorise une meilleure communication et une meilleure compréhension mutuelle entre les équipes de développement et d'opérations informatiques. Les membres des deux équipes travaillent ensemble de manière cohérente pour établir des objectifs communs et les atteindre de manière efficace et efficiente. Une communication ouverte et transparente permet de résoudre les problèmes plus rapidement, d'éviter les malentendus et d'améliorer la qualité du logiciel.
2. **Automatisation** : L'automatisation est un autre élément important de la culture DevOps, car elle permet de réduire les erreurs, les coûts et le temps de développement. Les processus manuels sont remplacés par des outils automatisés, tels que des scripts, des tests automatisés et des outils de déploiement, ce qui permet d'augmenter la vitesse de déploiement et de minimiser les risques liés à des erreurs humaines.
3. **Mesure** : La mesure est un élément crucial de la culture DevOps, car elle permet de suivre les performances des équipes et des processus, d'identifier les points de blocage et d'améliorer l'efficacité du développement. La mesure peut être utilisée pour évaluer la qualité du logiciel, l'efficacité des processus de développement et de déploiement, la satisfaction des utilisateurs et d'autres indicateurs clés.
4. **Partage** : Le partage est un élément essentiel de la culture DevOps, car il favorise la création d'une communauté de praticiens et d'experts en développement logiciel, qui partagent leurs connaissances, leurs idées et leurs expériences. Le partage peut prendre la forme de forums de discussion, de sessions de formation, de conférences et de webinaires, et il permet aux membres de la communauté de rester à jour sur les dernières tendances et les meilleures pratiques de l'industrie.

La culture DevOps repose sur la collaboration, l'automatisation, la mesure et le partage. Ces éléments clés sont étroitement liés et se renforcent mutuellement pour permettre une meilleure communication, une automatisation accrue, une évaluation des performances et une amélioration continue.

1.6 L’adoption de DevOps dans une entreprise : les étapes à suivre.

L’adoption de DevOps dans une entreprise peut être un processus complexe qui nécessite une planification et une mise en œuvre soigneuses. Voici quelques étapes à suivre pour adopter DevOps avec succès :

1. Identifier les objectifs et les besoins de l'entreprise en matière de développement logiciel et d'opérations informatiques.
2. Communiquer clairement les avantages de DevOps aux parties prenantes de l'entreprise, y compris la direction, les employés et les clients.
3. Évaluer les compétences actuelles des employés et identifier les formations nécessaires pour qu'ils puissent maîtriser les outils et les processus de DevOps.
4. Mettre en place des projets pilotes pour tester et évaluer l'efficacité de DevOps avant de le déployer à grande échelle.
5. Mettre en place des pratiques de sécurité pour minimiser les risques de sécurité liés à DevOps.
6. Évaluer et améliorer constamment les processus de développement et de déploiement pour optimiser l'efficacité de DevOps.
7. Encourager la collaboration et la communication entre les équipes Dev et Ops pour favoriser une culture de travail unifiée.

Après avoir suivi ces étapes, il faut mesurer les résultats de l'adoption de DevOps en utilisant des indicateurs clés de performance (KPIs) tels que

- Le temps de déploiement
- La fréquence des déploiements
- Le taux d'erreur,
- Le temps de récupération après incident, etc.

Il est important de noter qu’il faut continuer à améliorer et à ajuster les pratiques DevOps pour répondre aux besoins changeants de l'entreprise et de ses clients. Enfin, il est recommandé de documenter les processus et les pratiques DevOps pour faciliter la formation et l'intégration des nouveaux employés.

Prenons un exemple concret :

Une entreprise de développement de logiciels a rencontré des difficultés dans la collaboration entre les équipes de développement et d'opérations informatiques. Les processus de développement et de déploiement manuels ont causé des retards et des erreurs, ce qui a entraîné des coûts élevés et des pertes de productivité.

Pour résoudre ces problèmes, l'entreprise a décidé d'adopter la culture DevOps. Voici les étapes qu'elle a suivies :

- **Formation des employés** : L'entreprise a commencé par former tous ses employés sur les concepts clés de DevOps, tels que la collaboration, l'automatisation, la mesure et le partage. Cette formation a aidé les employés à comprendre les avantages de DevOps et à se familiariser avec les outils et les processus impliqués.
- **Identification des outils nécessaires** : L'entreprise a identifié les outils nécessaires pour automatiser les processus de développement et de déploiement. Elle a choisi des outils open source pour réduire les coûts.
- **Mise en place de processus automatisés** : L'entreprise a automatisé les processus de développement et de déploiement à l'aide des outils identifiés. Cela a réduit les erreurs et les coûts tout en augmentant la vitesse de déploiement.
- **Collaboration étroite** : Les équipes de développement et d'opérations informatiques ont travaillé ensemble de manière étroite pour atteindre des objectifs communs. Une communication ouverte et transparente a permis de résoudre les problèmes plus rapidement, d'éviter les malentendus et d'améliorer la qualité du logiciel.
- **Mesure et analyse des performances** : L'entreprise a mis en place des mécanismes de mesure et d'analyse des performances pour évaluer la qualité du logiciel, l'efficacité des processus de développement et de déploiement, la satisfaction des utilisateurs et d'autres indicateurs clés.
- **Partage des connaissances** : L'entreprise a encouragé le partage des connaissances entre les équipes de développement et d'opérations informatiques en organisant des sessions de formation, des forums de discussion et d'autres événements.

Après avoir suivi ces étapes, l'entreprise de développement de logiciels a réussi à adopter la culture DevOps et à surmonter les difficultés rencontrées dans la collaboration entre les équipes de développement et d'opérations informatiques.

La formation des employés, l'identification des outils nécessaires, la mise en place de processus automatisés, la collaboration étroite, la mesure et analyse des performances, ainsi que le partage des connaissances ont permis d'améliorer la qualité du logiciel, d'augmenter la vitesse de déploiement, de réduire les erreurs et les coûts, et d'améliorer la productivité globale de l'entreprise.

1.7 Résoudre les problèmes liés à l'adoption de DevOps

L'adoption de DevOps peut présenter certains défis pour une entreprise, notamment un investissement initial élevé, une complexité accrue, un changement culturel important et un risque accru de sécurité. Voici quelques mesures que les entreprises peuvent prendre pour résoudre ces problèmes et adopter avec succès la culture DevOps.

1.7.1 Investissement initial élevé :

L'investissement initial élevé peut être un obstacle pour les entreprises qui envisagent d'adopter DevOps. Cela peut inclure des coûts de formation, l'achat de nouveaux outils de développement et d'automatisation, ainsi que le temps nécessaire pour la mise en place de nouveaux processus.

Pour surmonter cet obstacle, les entreprises peuvent :

- Prioriser les investissements : Les entreprises peuvent commencer par investir dans les outils et les processus qui offrent les avantages les plus significatifs à court terme, puis étendre l'investissement à mesure que la culture DevOps se développe.
- Utiliser des outils open-source : Les outils open-source sont souvent gratuits ou peu coûteux, ce qui peut réduire les coûts initiaux de l'adoption de DevOps.
- Faire appel à des consultants : Les consultants peuvent aider les entreprises à identifier les besoins spécifiques de l'entreprise et à recommander les outils et les processus les plus adaptés.

1.7.2 Complexité

La complexité de DevOps peut rendre la mise en place difficile, en particulier pour les petites entreprises. Les nombreuses technologies, outils et processus impliqués peuvent être difficiles à comprendre et à gérer.

Pour surmonter cet obstacle, les entreprises peuvent :

- Se concentrer sur les objectifs : Les entreprises peuvent se concentrer sur les objectifs commerciaux clés et utiliser les outils et les processus qui permettent de les atteindre.
- Éviter la surcharge technologique : Les entreprises peuvent éviter de mettre en place trop d'outils et de technologies à la fois. Elles peuvent se concentrer sur les outils qui répondent à leurs besoins immédiats.
- Former les employés : La formation des employés sur les outils et les processus de DevOps peut aider à réduire la complexité en améliorant la compréhension et l'utilisation des outils.

1.7.3 Changement culturel

L'adoption de DevOps nécessite souvent un changement culturel important, en particulier en ce qui concerne la collaboration étroite entre les équipes de développement et d'exploitation.

Pour surmonter cet obstacle, les entreprises peuvent :

- Établir une vision claire : Les entreprises peuvent établir une vision claire et communiquer les avantages de DevOps aux employés. Cela peut aider à générer un engagement et un soutien pour le changement culturel.
- Favoriser la collaboration : Les entreprises peuvent favoriser la collaboration en organisant des réunions régulières, des ateliers et des formations pour les employés de différentes équipes.
- Récompenser la collaboration : Les entreprises peuvent récompenser les comportements et les résultats qui encouragent la collaboration, tels que la résolution rapide des problèmes.

1.7.4 Risque accru de sécurité

L'adoption de DevOps peut entraîner un risque accru de sécurité si les équipes ne prennent pas les précautions nécessaires pour sécuriser les processus de développement et de déploiement. Cela peut entraîner des failles de sécurité ou des violations de données.

Pour minimiser le risque de sécurité associé à l'adoption de DevOps, les entreprises peuvent:

- Effectuer des tests de sécurité : Les entreprises peuvent effectuer des tests de sécurité réguliers pour identifier les failles de sécurité potentielles et les corriger avant qu'elles ne soient exploitées.
- Adopter des pratiques de sécurité : Les entreprises peuvent adopter des pratiques de sécurité telles que l'authentification forte, le chiffrement des données et la gestion des accès pour minimiser les risques de sécurité.
- Former les employés : Les entreprises peuvent former leurs employés sur les meilleures pratiques de sécurité pour éviter les erreurs de sécurité courantes, telles que le partage de mots de passe et l'accès non autorisé.
- Suivre les mises à jour de sécurité : Les entreprises doivent suivre les mises à jour de sécurité des outils utilisés dans le processus DevOps et les appliquer rapidement pour minimiser les risques de sécurité.

1.8 Les méthodes agiles en DevOps

Les méthodes agiles sont souvent associées à DevOps car elles favorisent la collaboration, la flexibilité et la rapidité de développement. Les deux méthodes agiles les plus couramment utilisées en DevOps sont Scrum et Kanban.

1.8.1 Scrum

Scrum est une méthode agile de gestion de projet qui se concentre sur des cycles de développement courts, appelés « sprints », et une collaboration étroite entre les membres de l'équipe.

Dans Scrum, l'équipe de développement travaille en étroite collaboration avec le propriétaire du produit pour définir les objectifs et les fonctionnalités du produit.

Le processus Scrum se déroule en plusieurs étapes, notamment :

- **Planification de sprint** : L'équipe de développement et le propriétaire du produit se réunissent pour définir les objectifs du sprint et les fonctionnalités à développer.
- **Sprint** : L'équipe de développement travaille sur les fonctionnalités définies pendant la planification du sprint, avec des réunions quotidiennes pour assurer une communication fluide.
- **Revue de sprint** : L'équipe de développement et le propriétaire du produit se réunissent pour examiner les fonctionnalités développées pendant le sprint et discuter des résultats.
- **Rétrospective de sprint** : L'équipe de développement examine le sprint écoulé pour identifier les domaines d'amélioration.

Scrum peut être utilisé en combinaison avec les pratiques DevOps pour améliorer la rapidité de développement et de déploiement, ainsi que la collaboration entre les équipes.

Prenons un exemple de Scrum appliqué dans un contexte DevOps.

Contexte : Une équipe de développement souhaite utiliser Scrum pour gérer le développement d'une nouvelle fonctionnalité d'une application Web. L'objectif est de livrer cette fonctionnalité de manière régulière et fiable en utilisant des pratiques DevOps :

Planification de sprint : L'équipe de développement se réunit avec le propriétaire du produit pour définir les objectifs du sprint et les fonctionnalités à développer. Les objectifs sont de créer une nouvelle page d'inscription pour l'application et d'intégrer cette nouvelle fonctionnalité avec le système de gestion des utilisateurs existant. Voici les étapes :

1. **Sprint** : L'équipe de développement travaille sur les fonctionnalités définies pendant la planification du sprint. Ils utilisent des pratiques DevOps telles que l'intégration continue et le déploiement continu pour assurer que chaque changement est rapidement intégré et testé. L'équipe travaille en étroite collaboration avec l'administrateur système pour garantir que le déploiement de chaque mise à jour est automatisé et vérifié.
2. **Revue de sprint** : À la fin du sprint, l'équipe de développement et le propriétaire du produit se réunissent pour examiner les fonctionnalités développées pendant le sprint et discuter des résultats. L'administrateur système participe également à la revue pour partager les commentaires sur l'intégration continue et le déploiement continu.
3. **Rétrospective de sprint** : L'équipe de développement examine le sprint écoulé pour identifier les domaines d'amélioration. Ils discutent de l'efficacité de l'utilisation de Scrum et des pratiques DevOps, et identifient des améliorations pour le sprint suivant.

En utilisant Scrum avec des pratiques DevOps, l'équipe de développement peut livrer rapidement et efficacement des fonctionnalités fiables et de haute qualité pour l'application Web.

Il est important de noter que l'utilisation de Scrum avec des pratiques DevOps nécessite une collaboration étroite entre l'équipe de développement et l'administrateur système. Cette collaboration est essentielle pour garantir l'intégration continue, le déploiement continu et l'automatisation des tests, ce qui permet à l'équipe de développement de livrer rapidement et régulièrement des fonctionnalités à haute qualité. Il est également important d'effectuer des rétrospectives régulières pour identifier les domaines d'amélioration et s'assurer que les pratiques DevOps sont efficaces et efficacement intégrées dans le processus de développement de l'équipe.

1.8.2 Kanban

Kanban est une autre méthode agile qui peut être utilisée en DevOps. Elle se concentre sur l'amélioration continue et la réduction des délais de livraison. Le processus Kanban utilise des tableaux pour visualiser le flux de travail et suivre la progression des tâches.

Le processus Kanban se déroule en plusieurs étapes, notamment :

- **Définir les étapes du flux de travail** : L'équipe de développement décompose le flux de travail en étapes distinctes, telles que « à faire », « en cours » et « terminé ».
- **Limiter le travail en cours** : L'équipe de développement définit des limites pour chaque étape du flux de travail pour éviter les goulets d'étranglement.
- **Suivre les métriques** : L'équipe de développement suit les métriques, telles que le temps de cycle et le temps de traitement, pour identifier les domaines d'amélioration.
- **Amélioration continue** : L'équipe de développement examine régulièrement le processus Kanban pour identifier les domaines d'amélioration et apporter des modifications au flux de travail.

Kanban peut être utilisé en combinaison avec les pratiques DevOps pour améliorer la visibilité et la rapidité de livraison, ainsi que pour réduire les déchets et les retours en arrière.

Prenons un exemple de Scrum appliqué dans un contexte DevOps.

Contexte : Une équipe de développement utilise Kanban pour gérer le développement d'une application mobile. L'objectif est de garantir la rapidité et la fiabilité du processus de développement et de livraison grâce à des pratiques DevOps. Voici les étapes :

- **Définition du flux de travail** : L'équipe de développement utilise Kanban pour définir le flux de travail, les étapes de développement et les personnes responsables de chaque étape.
- **Création du tableau Kanban** : L'équipe crée un tableau Kanban pour visualiser le flux de travail et suivre l'état de chaque tâche.
- **Utilisation de la gestion de version** : L'équipe utilise la gestion de version pour suivre les changements de code et garantir l'intégrité du code de l'application.
- **Utilisation de l'intégration continue** : L'équipe utilise des pratiques DevOps telles que l'intégration continue pour garantir que chaque modification de code est rapidement intégrée, testée et déployée.
- **Utilisation du déploiement continu** : L'équipe utilise le déploiement continu pour automatiser le processus de déploiement, ce qui permet de livrer rapidement les nouvelles fonctionnalités et de corriger rapidement les bugs.
- **Amélioration continue** : L'équipe utilise Kanban pour suivre les métriques de performance telles que le temps moyen de cycle et le taux d'achèvement des tâches. Ils utilisent ces métriques pour identifier les goulots d'étranglement et les inefficacités du processus de développement, et pour apporter des améliorations continues.

En utilisant Kanban avec des pratiques DevOps, l'équipe de développement peut garantir la rapidité et la fiabilité de leur processus de développement et de livraison pour l'application mobile.

Il est important de noter que pour tirer le meilleur parti de Kanban et des pratiques DevOps, il est essentiel que l'équipe soit bien formée et comprenne les principes fondamentaux de chaque méthode. De plus, l'adaptation du processus de développement pour intégrer des pratiques DevOps peut nécessiter une certaine résistance au changement de la part de l'équipe, il est donc important de communiquer clairement les avantages potentiels et de faire preuve de patience et de flexibilité tout au long du processus.

En conclusion, les méthodes agiles telles que Scrum et Kanban peuvent être utilisées pour améliorer les pratiques DevOps en favorisant la collaboration, la rapidité de développement et l'amélioration continue. Les entreprises peuvent choisir la méthode agile qui convient le mieux à leur environnement de travail et la combiner avec les pratiques DevOps pour obtenir des résultats optimaux.

1.9 Les pratique utilisé dans un contexte DevOps

1.9.1 L'intégration continue (CI)

Les applications modernes sont décomposées en petits services indépendants qu'on appelle aussi micro services, chaque développeur travaille ainsi simultanément sur différentes parties d'une application.

Lorsqu'un développeur apporte des modifications à une application elles peuvent entrer en conflit avec les autres modifications apportées simultanément par d'autres développeurs, un outil scm (source code management) va permettre de centraliser au fur et à mesure les différentes modifications de code et ainsi gérer toutes les évolutions, git et gitlab sont des exemples d'outils scn très utilisés.

Le serveur d'intégration va récupérer ce code, ce serveur d'intégration est composé de plusieurs services :

Un service orchestration comme Jenkins va d'abord déclencher une série d'actions, lorsqu'il détecte une évolution du code source en fonction du langage de programmation, Jenkins va appeler un service chargé de réaliser la compilation, compiler ça veut dire créer un build c'est-à-dire une version exécutable de l'application ensuite le serveur d'intégration va réaliser une série de tests rapides pour vérifier que la modification n'introduit aucune erreur dans le code source.

On va ici retrouver les tests codés par le développeur, ce sont les tests unitaires qui sont là pour valider le bon fonctionnement d'une fonction, puis on a les tests d'intégration système qui sont des tests fonctionnels pour vérifier le bon fonctionnement de l'application dans son ensemble en Java, j unit est très populaire pour ces 2 tâches.

On a ensuite des contrôles qualité, un premier service va vérifier que la qualité du code n'a pas été dégradée par les modifications du code et pour ça on a par exemple sonar cube.

Si durant cette phase de test un problème est identifié la chaîne CI s'arrête et le serveur d'intégration informe le développeur, en revanche si la série de tests automatisés est passée avec succès le nouveau build est alors validé et archivé dans un repository maintenant que le nouveau build est centralisé il est prêt à être déployé.

1.9.2 Livraison continue (CD)

Le déploiement continu consiste à effectuer de nouveaux tests automatisés qui vont être réalisés à chaque livraison de l'application sur les différents environnements jusqu'à la production.

Pour cette nouvelle série de tests on a d'abord des tests automatisés de recettes via un outil comme Selenium, qui permet de dérouler des scénarios complets de tests d'une application.

Ensuite on a des tests qui permettent de tester l'écosystème applicatif complet, ce type de test est pris en charge par exemple par l'outil UFT one, c'est aussi le moment de réaliser des tests de charge et de performance pour vérifier que les nouvelles fonctionnalités ne dégradent pas les temps d'accès et d'exécution de l'application et qu'elles continuent bien de supporter un nombre défini d'utilisateurs simultanés.

Pour chacun de ces tests on va à la fois contrôler les nouvelles fonctions mais aussi la non régression dans ce cas on identifie pour chaque type de test vu précédemment des scénarios importants qui vont être revérifiés à chaque fois concernant le déploiement de l'application sur un environnement.

Il y a plusieurs possibilités historiquement on a un déploiement sur un serveur que ce soit physique ou virtuel via un simple script ou un outil de déploiement automatisé ou alors sur les applications plus modernes on va avoir recours au déploiement via conteneur avec une image docker pour les tests, il est même possible d'utiliser l'infrastructure as code pour livrer un environnement à partir d'un code.

Le principe c'est de repérer une erreur le plus tôt possible dans les étapes pour la corriger rapidement et le fait d'avoir des itérations plus petites et plus régulières les rend plus faciles à maîtriser, les erreurs ont ainsi moins d'impact puisqu'elles sont plus faciles à détecter et à résoudre.

1.9.3 L'intégration continue CI/CD

LE CI/CD est un ensemble de pratiques qui permettent d'accélérer le rythme de déploiement des applications à travers 2 grandes étapes clés :

- **Le CI** ou intégration continue cherche à automatiser les opérations autour du développement
- **Le CD** ou déploiement continu cherche à automatiser les opérations de déploiement

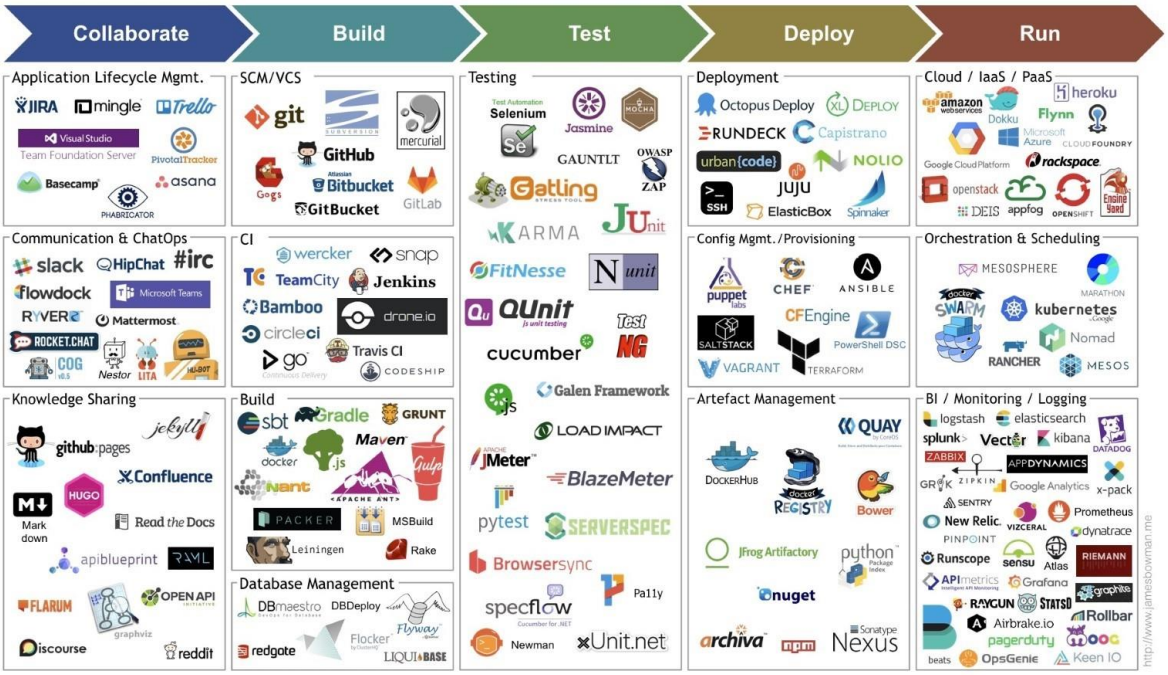
La rythme de mise en production d'une application est de 2 à 3 versions majeures par an avec à chaque fois un ensemble de nouvelles fonctionnalités on arrive donc à plusieurs mois entre 2 nouveautés majeures.
Ce délai s'explique notamment par un processus séquentiel entre les équipes de développeurs et les équipes d'Ops.
Sur les principales étapes de déploiement d'une nouvelle fonctionnalité, chacune est gérée par une équipe et leur enchaînement prend du temps.
En effet, à chaque étape il est nécessaire de synchroniser les différents acteurs, coordonner leur planning, planifier les actions à mettre en place etc...

1.9.4 Le pipeline CI/CD

L'implémentation d'un pipeline CI/CD vise à supprimer les activités humaines et manuelles, la volonté c'est d'automatiser autant que possible les différentes étapes pour pousser un développement jusqu'à la mise en production

1.10 Les outils

Vous pourrez retrouver les plus part des outils utilisés en DevOps dans l'image ci-dessous :



2 Ansible

2.1 Introduction à Ansible

2.1.1 Définition

Ansible est un outil open-source d'automatisation de la gestion de configuration, du déploiement d'applications, et de l'orchestration de tâches sur des systèmes informatiques.

Il permet de gérer de manière efficace et reproductible la configuration et les mises à jour de nombreux serveurs, en automatisant les tâches répétitives, en permettant une gestion centralisée, et en assurant la cohérence des configurations sur l'ensemble des systèmes gérés.

Ansible utilise une syntaxe simple en langage YAML pour décrire les configurations et les tâches à effectuer, et permet d'effectuer des opérations en mode push ou pull sur des hôtes distants via SSH. Ansible est largement utilisé en DevOps pour gérer les infrastructures, déployer les applications et automatiser les tâches récurrentes.

YAML est un format de données informatiques facile à lire et à écrire pour les humains. Il permet de stocker des informations structurées, comme des configurations de logiciels, des données de base de données, etc. Les données sont organisées en clés et en valeurs, avec une syntaxe simple et facile à comprendre. C'est un choix populaire pour la configuration de logiciels et les échanges de données entre différents systèmes en raison de sa simplicité et de sa flexibilité.

2.1.2 Histoire

Ansible a été créé en 2012 par **Michael DeHaan**. Il a d'abord travaillé sur le projet comme un projet personnel et a finalement décidé de le publier en open source.

Ansible tire son nom d'un concept de science-fiction. Dans l'univers de science-fiction de l'écrivain américain Ursula K. Le Guin, les "ansible" sont des dispositifs de communication instantanée qui permettent des échanges de messages à des vitesses superluminiques, ce qui signifie qu'ils peuvent communiquer à des distances astronomiques en un temps

presque nul. Le créateur d'Ansible, Michael DeHaan, a choisi le nom Ansible pour représenter l'objectif de l'outil : permettre aux équipes de développement de communiquer et de collaborer rapidement et efficacement.

La première version publique d'Ansible a été lancée en février 2012, et depuis lors, le projet a connu une croissance rapide en popularité. En 2015, Ansible a été acquis par Red Hat, une entreprise de logiciels open source, et a depuis été intégré dans l'offre de produits Red Hat.

Aujourd'hui, Ansible est l'un des outils les plus populaires pour l'automatisation de la gestion de configuration et le déploiement d'applications, en particulier dans le contexte de l'infrastructure informatique et du cloud computing. La communauté open source d'Ansible continue de développer le projet avec des mises à jour régulières et de nouvelles fonctionnalités.

2.1.3 Objectifs

Les objectifs d'Ansible sont de simplifier et d'automatiser les tâches de configuration, de déploiement et de gestion de l'infrastructure des systèmes informatiques. Plus précisément, Ansible vise à :

- Simplifier les tâches de configuration et de déploiement en utilisant une syntaxe simple et compréhensible par les humains
- Automatiser les tâches répétitives pour économiser du temps et minimiser les erreurs humaines
- Permettre une gestion de l'infrastructure à grande échelle en fournissant une plateforme de gestion centralisée
- Offrir une grande flexibilité pour s'adapter à différents types de systèmes, de configurations et de structures d'infrastructure
- Favoriser la collaboration et la communication entre les membres de l'équipe de développement et les équipes opérationnelles (DevOps) en fournissant des outils de gestion partagés
- Assurer la sécurité en permettant le cryptage des données de configuration et l'authentification des utilisateurs.

En résumé, Ansible vise à simplifier et à automatiser les tâches de gestion de l'infrastructure, tout en offrant une grande flexibilité et des fonctionnalités de sécurité, pour

faciliter la collaboration et la communication entre les équipes de développement et les équipes opérationnelles.

2.2 Fonctionnement

Ansible est un outil open-source qui permet de gérer la configuration et le déploiement d'infrastructures informatiques de manière automatisée. Il fonctionne sur une architecture de type client-serveur, où un serveur central (appelé "contrôleur Ansible") envoie des commandes à des machines distantes (appelées "nœuds").

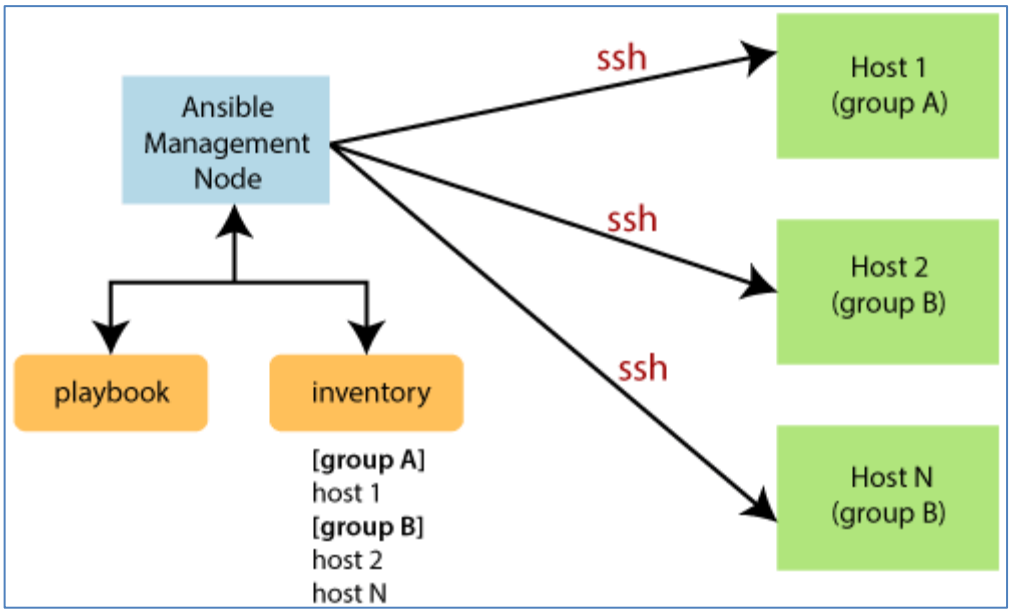
L'une des caractéristiques importantes d'Ansible est son système d'"inventory", qui permet de lister et de regrouper les nœuds à gérer. L'"inventory" peut être créé manuellement ou à l'aide de sources de données externes, telles que des listes de serveurs dans un fichier texte ou un service de découverte automatique. Cette fonctionnalité permet d'organiser et de catégoriser les nœuds en fonction de leur fonction, de leur emplacement géographique, etc., facilitant ainsi la gestion de grandes infrastructures.

Pour utiliser Ansible, il faut tout d'abord installer le contrôleur Ansible sur une machine, ainsi que des nœuds qui seront gérés par Ansible. Une fois les machines installées, Ansible utilise SSH pour se connecter aux nœuds et exécuter les tâches de configuration ou de déploiement.

Ansible utilise des "playbooks" pour décrire les tâches à effectuer. Un playbook est un fichier YAML qui contient une liste de tâches, chacune avec une description de l'état souhaité du système (par exemple, installer un logiciel, copier un fichier, etc.). Les playbooks sont exécutés en séquence, avec une tâche exécutée sur chaque nœud selon les spécifications du playbook.

Les playbooks Ansible peuvent également inclure des "rôles", qui sont des ensembles de tâches préconfigurées pour des fonctions courantes, telles que l'installation d'un serveur web ou la configuration d'une base de données.

L'un des avantages clés d'Ansible est sa simplicité d'utilisation et sa flexibilité. En utilisant des fichiers YAML pour décrire les tâches, Ansible permet une configuration facile à lire et à comprendre, ainsi qu'une automatisation rapide et facile de tâches complexes. De plus, Ansible peut être utilisé pour gérer des infrastructures de différentes tailles, allant de quelques machines à des milliers de nœuds.



2.3 Installation d'Ansible

Ansible est un outil de gestion de configuration open-source qui permet l'automatisation de tâches de déploiement, de configuration et de gestion de systèmes informatiques. Il peut être installé sur différents types de systèmes d'exploitation, y compris Linux, MacOs et Windows. Dans ce cours, nous allons expliquer comment installer et configurer Ansible sur différentes plateformes.

2.3.1 Installation et configuration d'Ansible sur Linux

Sur la plupart des distributions Linux, Ansible est disponible dans les dépôts officiels. Vous pouvez l'installer en utilisant votre gestionnaire de paquets. Par exemple, pour installer Ansible sur Ubuntu ou Debian, ouvrez un terminal et exécutez la commande suivante :

```
sudo apt-get update
sudo apt-get install ansible
```

2.3.2 Installation et configuration d'Ansible sur MacOs

Sur MacOs, vous pouvez installer Ansible en utilisant le gestionnaire de paquets Homebrew. Si vous n'avez pas encore installé Homebrew, vous pouvez le faire en utilisant la commande suivante dans un terminal :

```
/bin/bash -c "$(curl –fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Après avoir installé Homebrew, vous pouvez installer Ansible en utilisant la commande suivante :

```
brew install ansible
```

Homebrew est un gestionnaire de paquets pour les systèmes d'exploitation macOS et Linux. Il permet d'installer et de gérer facilement des logiciels et des outils open-source sur votre système. Homebrew est basé sur la ligne de commande et utilise des fichiers appelés "formules" pour installer des logiciels.

2.3.3 Installation et configuration d'Ansible sur Windows

Sur Windows, Ansible peut être installé en utilisant l'un des gestionnaires de paquets suivants : Chocolatey, Scoop ou Ansible Package Manager (APM).

Pour installer Ansible avec Chocolatey, ouvrez une invite de commande en tant qu'administrateur et exécutez la commande suivante :

```
choco install ansible
```

Pour installer Ansible avec Scoop, ouvrez une invite de commande et exécutez la commande suivante :

```
scoop install ansible
```

Pour nos ateliers, nous utiliserons une distribution Linux (Ubuntu 20.04). Ansible est mieux sur Linux car il est conçu pour fonctionner avec des systèmes Linux, offre une compatibilité complète avec les distributions Linux et utilise des outils Linux tels que SSH pour se connecter aux nœuds distants. De plus, la plupart des fonctionnalités avancées d'Ansible sont mieux prises en charge sur les systèmes Linux.

2.4 Configuration de ansible

2.4.1 Configuration du fichier d'inventaire

Une fois installer vous devrez configurer le fichier d'inventaire. Pour rappel, ce fichier permet de lister et de regrouper les nœuds à gérer. L'inventory peut être créé manuellement ou à l'aide de sources de données externes, telles que des listes de serveurs dans un fichier texte ou un service de découverte automatique.

Cette fonctionnalité permet d'organiser et de catégoriser les nœuds en fonction de leur fonction, de leur emplacement géographique, etc., facilitant ainsi la gestion de grandes infrastructures.

Pour modifier le contenu de votre inventaire Ansible sur un Ubuntu par défaut, on ouvre le répertoire /etc/ansible/hosts, sur votre nœud de contrôle Ansible :

```
nano /etc/ansible/hosts
```

Dans ce fichier nous pouvons nommer nos groupes de serveur et les associés à nos serveur de notre parc informatique.

L'exemple suivant définit 3 groupes :

- Un groupe nommé [servers] avec deux serveurs différents, chacun identifié par un alias personnalisé (ubu2, ubu3) et associé à leurs adresse IP
- Un groupe nommé [server2] avec un seul serveur Ubuntu (ubu2)
- Un groupe nommé [server3] avec un seul serveur Ubuntu (ubu3)

La commande ansible_host permet de préciser l'adresse IP de nos serveurs

Le **[all: vars]** sous-groupe définit le **ansible_python_interpreter** paramètre d'hôte qui sera valide pour tous les hôtes inclus dans cet inventaire.

Ce paramètre garantit que le serveur distant utilise **/usr/bin/python3** exécutable Python 3 au lieu de **/usr/bin/python** (Python 2.7), qui n'est pas présent sur les versions récentes d'Ubuntu.

```
[servers]
ubu2 ansible_host=192.168.240.13
ubu3 ansible_host=192.168.240.14

[server2]
ubu2 ansible_host=192.168.240.13

[server3]
ubu3 ansible_host=192.168.240.14

[all:vars]
ansible_python_interpreter=/usr/bin/python3
```

2.4.2 Activation de ssh sur les nœuds

Une que ansible est installé, vous devrez activer le ssh sur le contrôleur Ansible et sur les nœuds. Pour cela on se rend sur le fichier sshd_config et on active le port 22 et le PermitRootLogin :

```
nano /etc/ssh/sshd_config
```


```
Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin yes
```

"PermitRootLogin" est une directive de configuration dans le fichier de configuration SSH sur les systèmes Linux et Unix. Lorsque cette directive est définie sur "yes", elle autorise les connexions directes à l'utilisateur "root" via SSH, ce qui peut présenter un risque de sécurité potentiel pour le système. Il est généralement recommandé de définir cette directive sur "no" pour empêcher les connexions directes à l'utilisateur "root".

PermitRootLogin est une directive de configuration dans le fichier de configuration SSH sur les systèmes Linux et Unix. Lorsque cette directive est définie sur "yes", elle autorise les connexions directes à l'utilisateur "root" via SSH

2.4.3 Rattachement aux serveurs gérés et premières commandes

Une fois que le serveur Ansible est configurer et que le ssh est activé sur le contrôleur Ansible et sur les nœuds, nous devons établir notre premier contact entre les nœuds et le contrôleur Ansible.

Comme indiqué plus haut, les échanges seront effectués en utilisant le protocole SSH. A ce titre, deux méthodes d'authentifications sont utilisables :

- **La méthode login/password :** Méthode standard et là plus utilisée lors d'une connexion SSH manuelle, on saisit le login avec lequel on souhaite se connecter, puis le mot de passe
- **La méthode clé :** un échange de clé est à établir. Le serveur souhaitant se connecter doit avoir sa clé publique comme autorisée du côté du serveur sur lequel il souhaite se connecter.

Avec Ansible, nous préférons travailler avec l'échange de clé, il serait sinon nécessaire de saisir un mot de passe à chaque fois qu'une exécution de commande est effectuée sur un serveur distant.

Cela est difficilement pensable pour plusieurs dizaines de serveurs. Pour utiliser la méthode clé, vous devez tout d'abord générer une paire de clés SSH sur le serveur Ansible à l'aide de la commande "ssh-keygen" :

```
ssh-Kegen
```

Ensuite, la clé publique générée doit être copiée sur les nœuds gérés à l'aide de la commande "ssh-copy-id" qui permet d'envoyer à distance clé publique généré sur les nœuds :

```
ssh-copy-id root@adresse_IP_du_noeud-1
ssh-copy-id root@adresse_IP_du_noeud_2
```

Après chacune de ces deux commandes, le mot de passe de l'utilisateur root sur sera à saisir. Après cela, l'authentification SSH est automatiquement prise en compte

2.5 Concepts de base d'Ansible

Les concepts de base d'Ansible sont les playbooks, les tâches et les rôles. Comprendre ces concepts est essentiel pour utiliser Ansible efficacement.

2.5.1 Playbooks

Un playbook est un fichier YAML qui contient une liste de tâches à effectuer sur les nœuds cibles. Les playbooks sont conçus pour être facilement lisibles et éditables par les utilisateurs, car ils utilisent une syntaxe simple et facile à comprendre.

Un playbook peut contenir une ou plusieurs tâches, chacune contenant une description de l'état souhaité du système cible (par exemple, installer un paquet, copier un fichier, etc.). Les playbooks peuvent également inclure des variables et des conditions, ce qui permet de personnaliser les tâches en fonction de la situation.

Afin d'exécuter un playbook, il suffit d'utiliser la commande ansible-playbook suivit du nom de la playbook :

```
ansible-playbook playbook.yml
```

2.5.2 Tâches

Les tâches sont les éléments de base des playbooks. Chaque tâche décrit une action à effectuer sur les nœuds cibles. Les tâches sont généralement des commandes, des scripts ou des modules Ansible préconstruits qui sont exécutés sur les nœuds cibles.

Les tâches sont exécutées dans l'ordre dans lequel elles apparaissent dans le playbook, sauf si un ordre différent est spécifié. Chaque tâche est exécutée sur tous les nœuds cibles spécifiés dans le playbook.

2.5.3 Rôles

Un rôle est un ensemble de tâches, de variables et de fichiers organisés de manière logique et réutilisable. Les rôles permettent de structurer les playbooks de manière modulaire et de les rendre plus faciles à gérer et à maintenir.

Les rôles peuvent être utilisés pour décrire des tâches courantes ou pour encapsuler des tâches complexes dans une unité de gestion unique.

Par exemple, un rôle peut être créé pour installer et configurer un serveur web, ou pour gérer la configuration de la sécurité du système.

Les rôles peuvent être appelés depuis des playbooks ou depuis d'autres rôles. Les variables peuvent également être définies à l'intérieur d'un rôle, ce qui permet de personnaliser le comportement du rôle en fonction de la situation.

2.6 Atelier pratique: Création d'un playbook simple avec Ansible

Dans cet atelier pratique, l'objectif est d'installer le serveur web Apache 2 sur deux nœuds distants (192.168.240.40 et 192.168.240.50) en utilisant Ansible. Pour ce faire, vous devrez suivre les étapes suivantes :

1. Assurez-vous que les nœuds distants sont accessibles depuis le contrôleur Ansible en activant le SSH sur les nœuds et sur le contrôleur Ansible.
2. Configuration le fichier d'inventaire afin de lister et de regrouper les nœuds à gérer, pour notre cas ça sera 2 Ubuntu serveur version 20.04 en ayant comme address IP 192.168.240.40 et 192.168.240.50
3. Etablir le contact entre les nœuds et le contrôleur Ansible grâce à la méthode clé
4. Créez un fichier de configuration Ansible appelé "apache2.yml" contenant les tâches à exécuter pour installer Apache 2 sur les nœuds distants. Le fichier YAML doit inclure les éléments suivants :
    - La définition de l'hôte ou du groupe d'hôtes cibles pour lesquels les tâches doivent être exécutées.
    - La définition des tâches à effectuer pour installer Apache 2.
    - L'utilisation d'un module Ansible approprié (par exemple, "apt" pour les distributions Linux basées sur Debian ou Ubuntu) pour installer les paquets Apache 2 sur les nœuds distants.
5. Exécutez le fichier de configuration Ansible à l'aide de la commande "ansible-playbook" pour installer Apache 2 sur les nœuds distants. Vérifiez que l'installation a été effectuée correctement en accédant aux serveurs Apache à partir d'un navigateur web.

Cet atelier pratique vous permettra de vous familiariser avec les commandes de base d'Ansible pour exécuter des tâches sur des nœuds distants et installer des applications telles qu'Apache 2.

2.6.1 Prérequis

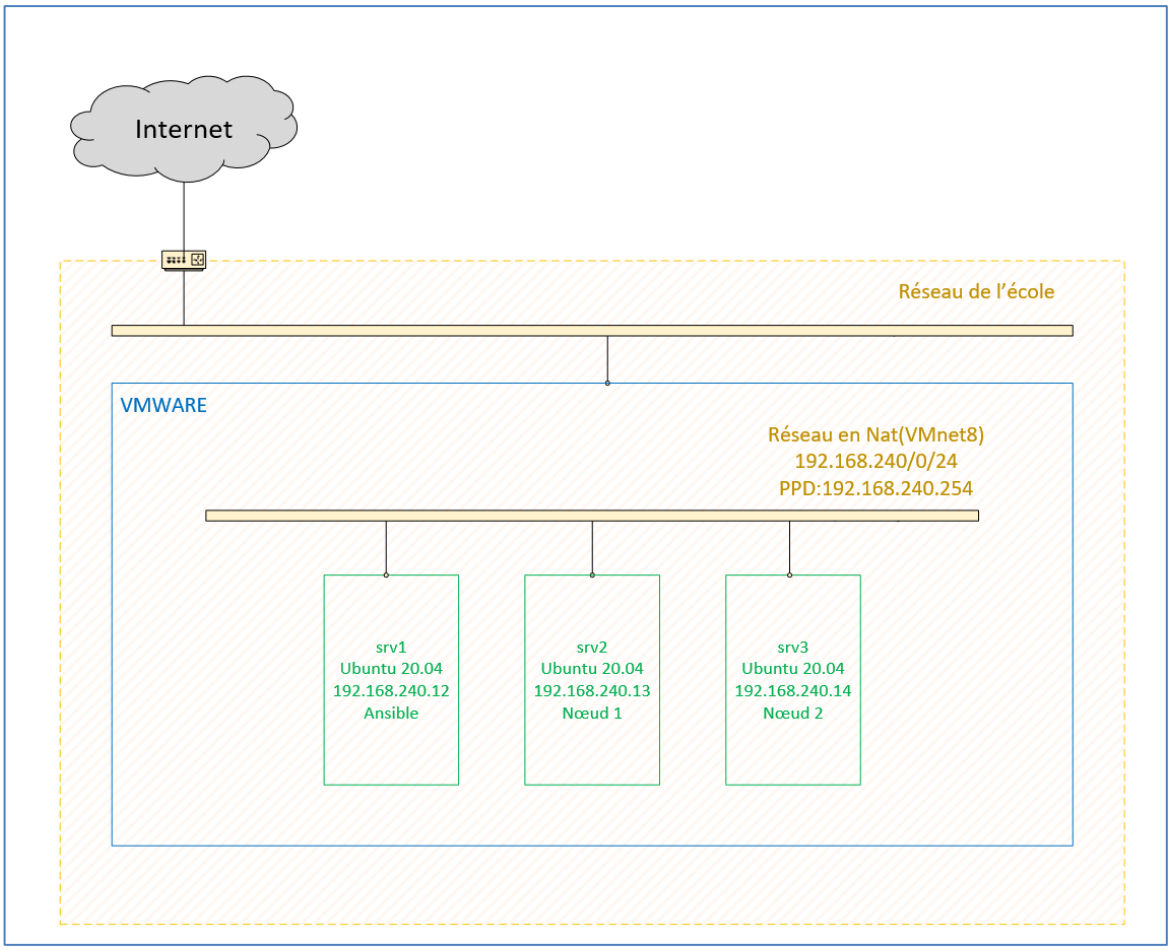
Pour ce TP, nous aurons besoin :

- 1 Ubuntu 20.04 avec Ansible installé : 192.168.240.20
- 2 Ubuntu 20.04 qui seront nos nœuds : 192.168.240.30 et 192.168.240.40

Ces 3 machines devront être dans le même sous-réseau, en Nat vmnet8 sur VMware WorkStation pro 16 ou 17 :

- Adresse réseau :192.168.240.0/24
- Passerelle : 192.168.240.254

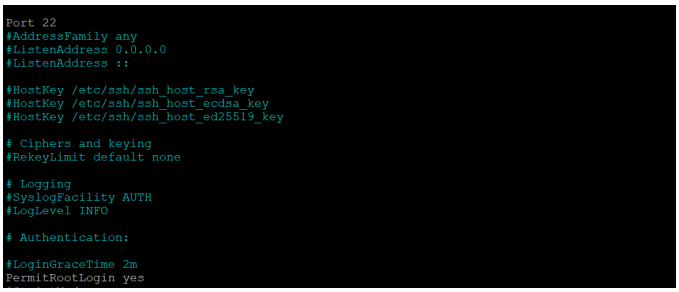
Voici le schéma réseau :



2.6.2 Configuration de SSH

Pour commencer, nous allons activé SSH sur nos 3 machines Ubuntu en activant le port 22 et le PermitRootLogin :

```
nano /etc/ssh/sshd_config
```

2.6.3 Installation d'Ansible

Afin d'utiliser Ansible comme moyen de gestion de votre infrastructure de serveur, nous devons installer le logiciel Ansible sur la machine qui servira de nœud de contrôle Ansible. Depuis notre nœud de contrôle (srv1), on exécute la commande suivante pour inclure le PPA (archive personnelle de packages) du projet officiel dans la liste des sources de notre système :

```
sudo apt-add-repository ppa:ansible/ansible
```

Appuyez sur ENTER lorsque vous êtes invité à accepter l'ajout de PPA. Ensuite, on actualise l'index des packages de notre système afin qu'il connaisse les packages disponibles dans le PPA nouvellement inclus :

```
sudo apt update
```

Suite à cette mise à jour, nous devons installer le logiciel Ansible avec :

```
sudo apt install ansible
```

Notre nœud de contrôle Ansible dispose désormais de tous les logiciels nécessaires pour administrer nos hôtes. Ensuite, nous verrons comment ajouter nos hôtes au fichier d'inventaire du nœud de contrôle afin qu'il puisse les contrôler.

2.6.4 Configuration du fichier d'inventaire

Une fois installer vous devrez configurer le fichier d'inventaire. Pour rappel, ce fichier permet de lister et de regrouper les nœuds à gérer

Pour modifier le contenu de votre inventaire Ansible par défaut, on ouvre le répertoire **/etc/ansible/hosts**, sur votre nœud de contrôle Ansible :


```
nano /etc/ansible/hosts
```

L'exemple suivant définit le groupe nommé **[servers]** avec deux serveurs différents, chacun identifié par un alias personnalisé (ubu2, ubu3) et associé à leurs adresse IP

Le **[all: vars]** sous-groupe définit le **ansible_python_interpreter** paramètre d'hôte qui sera valide pour tous les hôtes inclus dans cet inventaire.

Ce paramètre garantit que le serveur distant utilise **/usr/bin/python3** exécutable Python 3 au lieu de **/usr/bin/python** (Python 2.7), qui n'est pas présent sur les versions récentes d'Ubuntu.

```
[servers]
ubu2 ansible_host=192.168.240.13
ubu3 ansible_host=192.168.240.14
```

```
[all:vars]
ansible_python_interpreter=/usr/bin/python3
```

Lorsque vous avez terminé, enregistrez et fermez le fichier en appuyant sur CTRL+X puis Y et ENTER pour confirmer vos modifications. Chaque fois que vous souhaitez vérifier votre inventaire, vous pouvez exécuter :

```
ansible-inventory --list -y
```

L'alias dans le fichier d'inventaire ne doit pas nécessairement être le nom des hôtes des machines. L'alias est un moyen de référencer les hôtes dans les playbooks et les tâches, et peut être un nom personnalisé qui facilite la lecture et la compréhension du playbook.

Cependant, il est important de noter que l'alias doit être unique pour chaque hôte dans le fichier d'inventaire, et doit correspondre au nom de l'hôte tel que défini dans la configuration SSH.

2.6.5 Rattachement aux serveurs

Une fois que le serveur Ansible est configurer et que le ssh est activé sur le contrôleur Ansible et sur les nœuds, nous devons établir notre premier contact entre les nœuds et le contrôleur Ansible via la méthode clé. Pour utiliser la méthode clé, vous devez tout d'abord générer une paire de clés SSH sur le serveur Ansible à l'aide de la commande "ssh-keygen" :


```
ssh-Kegen
```

Ensuite, la clé publique générée doit être copiée sur les nœuds gérés à l'aide de la commande "ssh-copy-id" qui permet d'envoyer à distance clé publique généré sur les nœuds :

```
ssh-copy-id root@192.168.240.13
ssh-copy-id root@192.168.240.14
```

Après chacune de ces deux commandes, le mot de passe de l'utilisateur root sur sera à saisir. Après cela, l'authentification SSH est automatiquement prise en compte.

2.6.6 Test de connexion

Après avoir configuré le fichier d'inventaire pour inclure vos serveurs, il est temps de vérifier si Ansible est capable de se connecter à ces serveurs et d'exécuter des commandes via SSH.

Pour ce guide, nous utiliserons le compte racine Ubuntu car c'est généralement le seul compte disponible par défaut sur les serveurs nouvellement créés. Si vos hôtes Ansible ont déjà créé un utilisateur sudo régulier, nous vous encourageons à utiliser ce compte à la place.

Vous pouvez utiliser **–u** argument pour spécifier l'utilisateur du système distant. S'il n'est pas fourni, Ansible essaiera de se connecter en tant qu'utilisateur système actuel sur le nœud de contrôle. À partir de votre ordinateur local ou du nœud de contrôle Ansible, exécutez :

```
ansible all -m ping -u root
```

Cette commande utilisera le **ping** module intégré d'Ansible pour exécuter un test de connectivité sur tous les nœuds de votre inventaire par défaut, en vous connectant en tant que root. Le ping module testera :

- Si les hôtes sont accessibles ;
- Si vous disposez d'informations d'identification SSH valides ;
- Si les hôtes sont capables d'exécuter des modules Ansible à l'aide de Python.

2.6.7 Création du playbook

Maintenant nous allons créer un playbook Ansible pour installer Apache2 sur les deux nœuds :

```
nano /etc/ansible/galera/tasks/playbook.yml
```


```
---
- name: Installer Apache2
  hosts: servers
  become: true
  tasks:
    - name: Installer Apache2
      apt:
        name: apache2
        state: present
    - name: Démarrer le service Apache 2
      service:
        name: apache2
        state: started
        enabled: true
```

Explications :

- **La première ligne ---** : c'est une convention YAML pour indiquer le début d'un nouveau document.
- **Le nom du playbook** est défini dans la ligne suivante (name: Installer Apache 2).
- **Les adresses IP des nœuds** cibles sont définies dans la section hosts (ici, le groupe servers).
- **La ligne become**: true permet de s'assurer que les tâches seront exécutées en mode "superutilisateur" (root) pour pouvoir installer des paquets et démarrer des services.
- **Les tâches** sont définies dans la section tasks. La première tâche installe Apache 2 en utilisant le module apt. La deuxième tâche démarre le service Apache 2 et le configure pour qu'il démarre automatiquement au démarrage de la machine.

2.6.8 Exécuter le playbook

Pour exécuter ce playbook utiliser la commande suivante :

```
ansible-playbook playbook.yml
```

Une fois installer, retourné sur vos nœuds pour vérifier si apache 2 est bien installer

3 Vagrant

3.1 Introduction à Vagrant

3.1.1 Définition

Vagrant est un outil open-source qui permet de créer et de gérer des environnements de développement locaux. Il utilise des technologies de virtualisation telles que VirtualBox pour créer des machines virtuelles qui peuvent être facilement configurées et partagées entre les membres de l'équipe.

Avec Vagrant, les développeurs peuvent créer des environnements de développement cohérents et reproductibles, ce qui peut aider à résoudre les problèmes liés aux différences entre les systèmes d'exploitation et les configurations matérielles des développeurs.

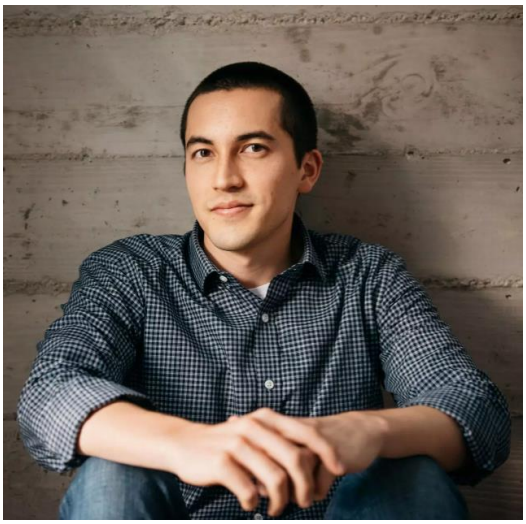
Vagrant utilise des "fichiers de configuration" appelés **Vagrantfiles** pour décrire la configuration de la machine virtuelle. Ces fichiers peuvent inclure des informations sur la mémoire et le processeur alloués à la machine virtuelle, les ports de communication, les chemins de partage de fichiers et d'autres paramètres de configuration.

Vagrant prend également en charge l'installation automatisée de logiciels sur la machine virtuelle à l'aide de "**provisioners**", qui sont des scripts d'installation automatisés qui peuvent être écrits en utilisant des outils tels que Ansible ou Shell.

Vagrant est souvent utilisé en conjonction avec des outils de gestion de configuration tels que Ansible pour créer des environnements de développement complets et reproductibles pour les projets logiciels.

3.1.2 Histoire

Vagrant est un outil open-source qui a été créé en 2010 par Mitchell Hashimoto et John Bender. Hashimoto a créé Vagrant après avoir travaillé dans le développement de logiciels et avoir remarqué la difficulté de configurer les environnements de développement de manière cohérente sur plusieurs machines.




Il a commencé à travailler sur Vagrant pour résoudre ce problème en fournissant un moyen facile de créer et de gérer des environnements de développement portables.

Vagrant a connu une croissance rapide dans la communauté de développement, en grande partie grâce à sa simplicité d'utilisation et sa flexibilité. En 2012, Hashimoto a fondé la société HashiCorp pour développer Vagrant ainsi que d'autres outils de gestion d'infrastructure, tels que Packer, Consul et Terraform.

Aujourd'hui, Vagrant est utilisé par des milliers de développeurs et d'entreprises pour gérer leurs environnements de développement et faciliter le processus de développement logiciel.

3.1.3 Objectifs

L'objectif de Vagrant est de faciliter la création et la gestion d'environnements de développement.

En utilisant des outils de virtualisation tels que VirtualBox ou VMware, Vagrant permet aux développeurs de créer des machines virtuelles préconfigurées pour des projets spécifiques, évitant ainsi les problèmes de configuration et de compatibilité des environnements de développement entre les membres de l'équipe.

Vagrant permet également de simplifier la mise en place d'outils de provisionnement tels que Ansible, Chef ou Puppet pour automatiser l'installation et la configuration de logiciels sur les machines virtuelles.

3.2 Fonctionnement

Vagrant est un outil open-source qui permet aux développeurs de créer et de gérer facilement des environnements de développement isolés. Ces environnements sont créés en utilisant des technologies de virtualisation et sont entièrement configurables pour répondre aux besoins du projet de développement. Voici comment Vagrant fonctionne :

1. **Installation de Vagrant** : Tout d'abord, vous devez installer Vagrant sur votre machine. Vous pouvez télécharger le package d'installation correspondant à votre

système d'exploitation à partir du site officiel de Vagrant et suivre les instructions pour l'installer.

2. **Création d'un Vagrantfile** : Le Vagrantfile est un fichier de configuration qui décrit l'environnement de développement que vous souhaitez créer. Pour créer un Vagrantfile, vous pouvez utiliser la commande suivante, Cela créera un fichier Vagrantfile vide dans le répertoire actuel :

```
vagrant init
```

3. **Configuration du Vagrantfile** : Vous pouvez maintenant configurer votre environnement de développement en éditant le fichier Vagrantfile. Le fichier Vagrantfile utilise une syntaxe simple pour spécifier des détails tels que le système d'exploitation, la quantité de mémoire et de processeurs, les ports à ouvrir, les logiciels à installer, etc.

   Par exemple, pour configurer un environnement de développement avec Ubuntu 20.04 et 2 Go de mémoire, le Vagrantfile ressemblerait à ceci :

```
 Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/focal64"
  config.vm.provider "virtualbox" do |vb|
    vb.memory = "2048"
  end
end
```

4. **Démarrage de la machine virtuelle** : Une fois que vous avez configuré votre Vagrantfile, vous pouvez lancer votre environnement de développement en utilisant la commande suivante :

```
vagrant up
```

   Cela va démarrer une machine virtuelle basée sur les spécifications de votre Vagrantfile et installer les logiciels spécifiés.

5. **Accès à la machine virtuelle** : Une fois que la machine virtuelle est lancée, vous pouvez vous y connecter en utilisant la commande suivante, cela vous permettra de vous connecter à la machine virtuelle et d'exécuter des commandes dans l'environnement de développement :

```
vagrant ssh
```

6. **Gestion de la machine virtuelle** : Vous pouvez utiliser l'interface en ligne de commande de Vagrant pour gérer la machine virtuelle. Par exemple, vous pouvez l'arrêter en utilisant la commande suivante :

```
vagrant halt
```

Vous pouvez également suspendre ou redémarrer la machine virtuelle en utilisant les commandes correspondantes :

```
vagrant reload
```

7. **Provisionnement** : Vagrant prend en charge plusieurs outils de provisionnement, tels que Ansible, Chef et Puppet, qui vous permettent d'automatiser la configuration de la machine virtuelle. Vous pouvez spécifier des scripts de provisionnement dans votre fichier Vagrantfile qui installeront automatiquement les logiciels et les dépendances dont votre projet a besoin lors du démarrage de la machine virtuelle.

En résumé, vous éditez un fichier de configuration « Vagrantfile », vous le donnez à manger à Vagrant, Vagrant récupère l'image système (sur internet ou en local) que vous lui avez indiqué, il envoie la configuration et l'image système à l'hyperviseur que vous lui avez indiqué, ce dernier créé la machine virtuelle décrite dans le Vagrantfile, vous pouvez maintenant jouir de votre VM de la manière que vous souhaitez (via Vagrant, via l'interface de l'hyperviseur, via un terminal tiers, SSH, etc.)

3.3 Téléchargement de Vagrant

L'installation de Vagrant est une étape importante pour pouvoir utiliser cet outil de gestion d'environnements de développement.

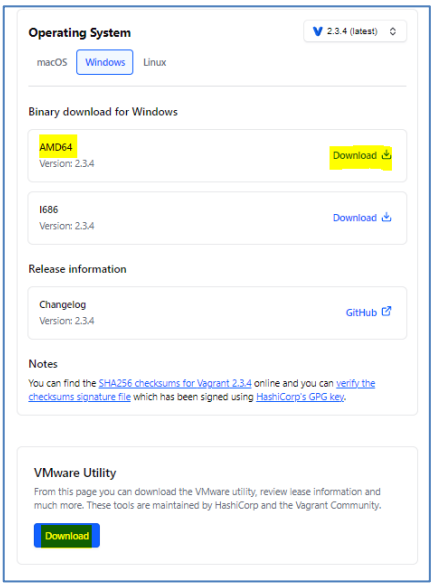
Avant d'installer Vagrant, assurez-vous que votre système dispose des prérequis nécessaires. Vagrant nécessite une version récente de VirtualBox ou de VMware pour pouvoir créer et

gérer des machines virtuelles. Pour notre cas, nous allons utiliser VMware WorkStation Pro 16 ou 17, dont voici le lien de téléchargement de l'installateur de la version Pro :

https://www.vmware.com/go/getworkstation-win

Après l'installation de VMware, rendez-vous sur le site officiel de Vagrant pour télécharger le package d'installation correspondant à votre système d'exploitation. Il existe des packages pour Windows, Mac OS X, Linux et d'autres systèmes d'exploitation, pour notre cas, nous allons télécharger la package windows et vue que nous allons utiliser VMware WorkStation Pro 16 ou 17, nous allons également télécharger le plugin **Vagrant VMware Utility**. Si votre ordinateur a un processeur 32 bits, vous devez utiliser la version **i686** de Vagrant. Si votre ordinateur a un processeur 64 bits, vous devez utiliser la version **amd64** de Vagrant. :

https://www.vagrantup.com/downloads

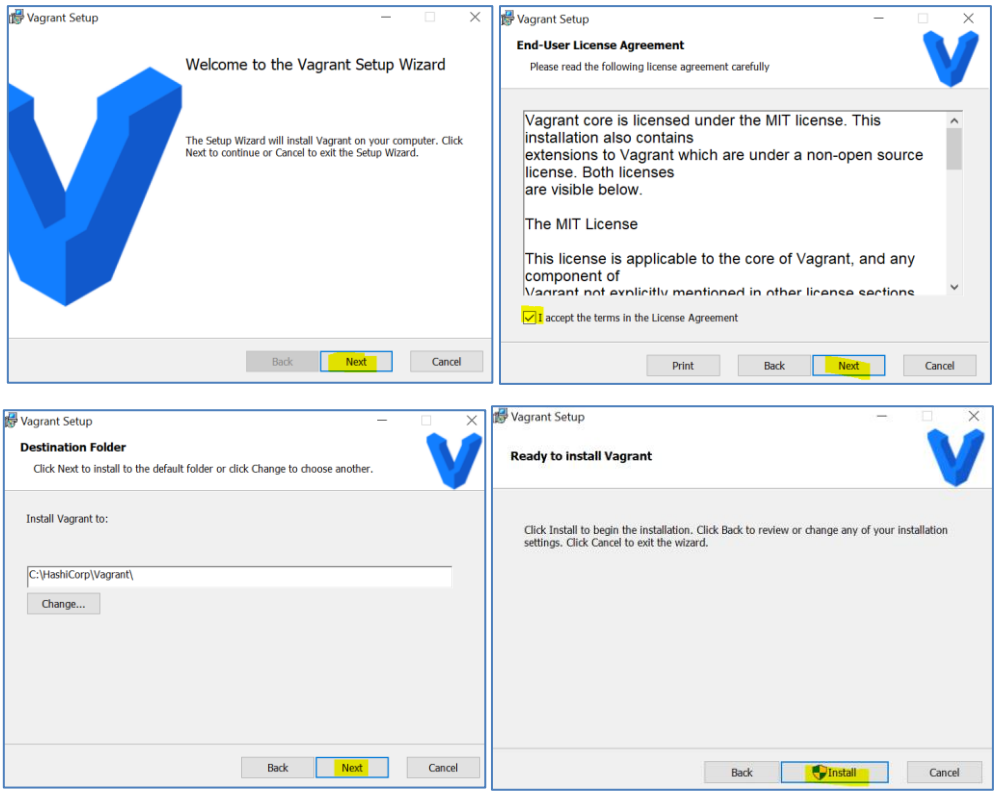


3.4 Installation de Vagrant

Une fois que vous avez téléchargé le package d'installation, vous pouvez exécuter l'installation Vagrant en double-cliquant dessus pour lancer l'assistant d'installation.

Vous devez d'abord installer Vagrant avant d'installer le plugin Vagrant VMware Utility pour prendre en charge VMware en tant que fournisseur de virtualisation. Le plugin Vagrant VMware Utility est un plugin qui s'intègre à Vagrant pour permettre à Vagrant d'utiliser VMware Workstation ou Fusion en tant que fournisseur de virtualisation. Le plugin ajoute des fonctionnalités supplémentaires à Vagrant pour prendre en charge les machines virtuelles créées avec VMware.

Suivez les instructions de l'assistant pour installer Vagrant sur votre système Windows :



Après avoir installé Vagrant, vous pouvez vérifier si l'installation s'est bien déroulée en ouvrant une invite de commande et en tapant la commande suivante :

```
vagrant --version
```


3.5 Installation du plugin Vagrant VMware Utility

Une fois vagrant install, vous devrez installer le plugin Vagrant VMware Utility.

Pour installer le plugin, vous pouvez ouvrir un terminal ou une invite de commande et exécuter la commande :

```
vagrant plugin install vagrant-vmware-desktop
```

Cela téléchargera et installera le plugin, et ajoutera le vmware_desktop à Vagrant.

3.6 Concepts de base Vagrant

Les concepts de base de Vagrant sont les fichiers Vagrantfiles, les Boîtes Vagrant et les commandes Vagrant. Comprendre ces concepts est essentiel pour utiliser Ansible efficacement.

3.6.1 Les fichiers Vagrantfiles

Un fichier Vagrantfile est un fichier de configuration qui décrit l'environnement de développement que vous souhaitez créer à l'aide de Vagrant. Le fichier est écrit en Ruby et utilise une syntaxe simple pour spécifier des détails tels que le système d'exploitation, la quantité de mémoire et de processeurs, les ports à ouvrir, les logiciels à installer, etc.

Voici les éléments clés d'un fichier Vagrantfile :

1. **Version de Vagrant** : La première ligne d'un fichier Vagrantfile spécifie la version de Vagrant utilisée. La syntaxe standard est **Vagrant.configure("2") do |config|** pour la version 2 de Vagrant.
2. **Configuration de la machine virtuelle** : Cette section du fichier Vagrantfile spécifie les détails de la machine virtuelle à créer. Vous pouvez spécifier la boîte Vagrant à utiliser, les interfaces réseau, la quantité de mémoire et de processeurs, les scripts de provisionnement, etc.

Par exemple, pour spécifier l'utilisation de la boîte Vagrant Ubuntu 20.04, vous pouvez utiliser la ligne suivante :

```
config.vm.box = "ubuntu/focal64"
```

Pour spécifier une interface réseau chez VMware, vous pouvez utiliser la ligne suivante :

```
config.vm.network "private_network", ip: "192.168.240.10", :adapter => 2
```

Cette ligne est utilisée pour configurer une interface réseau privée avec une adresse IP statique. Cette ligne indique à Vagrant de créer une nouvelle interface réseau dans la machine virtuelle, avec une adresse IP de 192.168.240.10.

3. **Fournisseur de virtualisation** : Pour utiliser VMware comme hyperviseur, vous devez spécifier le fournisseur de virtualisation dans le fichier Vagrantfile. Vous pouvez utiliser la ligne suivante pour spécifier VMware :

```
config.vm.provider "vmware_workstation" do |v|
```

La section v spécifie les options de configuration pour le fournisseur de virtualisation VMware. Vous pouvez spécifier des options telles que la quantité de mémoire et de processeurs, l'emplacement du fichier de disque virtuel, la compatibilité de la machine virtuelle, etc.

3.6.2 Les commandes Vagrant

Les commandes Vagrant sont utilisées pour interagir avec les machines virtuelles et les fichiers de configuration Vagrantfile. Voici les principales commandes Vagrant que vous pouvez utiliser :

- **Vagrant up** : Cette commande démarre la machine virtuelle en fonction de la configuration définie dans le fichier Vagrantfile.
- **Vagrant ssh** : Cette commande vous permet de vous connecter à la machine virtuelle via une connexion SSH.
- **Vagrant halt** : Cette commande arrête la machine virtuelle en cours d'exécution.
- **Vagrant destroy** : Cette commande supprime définitivement la machine virtuelle et tous ses fichiers associés.
- **Vagrant reload** : Cette commande redémarre la machine virtuelle en appliquant les dernières modifications apportées à la configuration Vagrantfile.
- **Vagrant status** : Cette commande affiche l'état actuel de la machine virtuelle.
- **Vagrant provision** : Cette commande exécute à nouveau la provision de la machine virtuelle en appliquant les modifications de configuration qui ont été apportées.
- **Vagrant package** : Cette commande permet de créer une boîte Vagrant personnalisée à partir de la machine virtuelle en cours d'exécution.
- **Vagrant box add** : Cette commande permet d'ajouter une nouvelle boîte Vagrant à la liste des boîtes disponibles.
- **Vagrant box remove** : Cette commande permet de supprimer une boîte Vagrant de la liste des boîtes disponibles.

Il est important de noter que la plupart des commandes Vagrant doivent être exécutées dans le répertoire où se trouve le fichier Vagrantfile correspondant. Les commandes peuvent être utilisées avec des options et des arguments supplémentaires pour personnaliser leur comportement. Vous pouvez obtenir une aide détaillée sur l'utilisation de chaque commande en exécutant vagrant help [command], où [command] est le nom de la commande pour laquelle vous avez besoin d'aide.

Voici d'autres commandes qui peuvent être utiles :

Commandes	
box	gère les boitiers : pose, dépose, etc.
connect	se connecter à un environnement Vagrant partagé à distance
destroy	arrête et efface toute trace de la machine
global-status	état des sorties environnements Vagrant pour cet utilisateur
halt	arrête la machine Vagrant
help	affiche l'aide d'une sous-commande
init	initialise un nouvel environnement Vagrant en créant un Vagrantfile
login	connectez-vous à l'Atlas de HashiCorp
package	emballe un environnement Vagrant en cours d'exécution dans une boîte
plugin	gère les plugins : installer, désinstaller, mettre à jour, etc.
provision	approvisionne la machine Vagrant
push	déploie le code dans cet environnement vers une destination configurée
rdp	se connecte à la machine via RDP
reload	redémarre la machine vagrant, charge la nouvelle configuration de Vagrantfile
resume	reprendre une machine Vagrant suspendue
share	partagez votre environnement Vagrant avec n'importe qui dans le monde
ssh	se connecte à la machine via SSH
ssh-config	sorties OpenSSH configuration valide pour se connecter à la machine
status	état des sorties de la machine Vagrant
suspend	suspend la machine
up	commence et approvisionne l'environnement Vagrant
version	imprime la version actuelle et la dernière version de Vagrant

3.6.3 Les Boîtes Vagrant

Les boîtes Vagrant sont des images de machines virtuelles préconfigurées qui contiennent un système d'exploitation et un ensemble de logiciels spécifiques. Les boîtes Vagrant sont utilisées comme base pour créer des machines virtuelles personnalisées avec Vagrant.

Les boîtes Vagrant sont fournies par la communauté Vagrant ou par des fournisseurs tiers, et sont disponibles pour différents systèmes d'exploitation.

Les boîtes Vagrant sont des fichiers compressés qui contiennent une image de la machine virtuelle, ainsi qu'un fichier de configuration qui définit la configuration initiale de la machine virtuelle.

Vous pouvez rechercher des boîtes Vagrant disponibles à partir de la commande vagrant box list, ou vous pouvez ajouter une nouvelle boîte Vagrant à partir d'un fichier .box en utilisant la commande vagrant box add.

Vous pouvez également créer votre propre boîte Vagrant à partir d'une machine virtuelle existante en utilisant la commande vagrant package.

Lorsque vous créez une machine virtuelle Vagrant à partir d'une boîte, Vagrant copie la boîte et crée une nouvelle machine virtuelle basée sur cette copie. La boîte originale reste disponible pour créer d'autres machines virtuelles.

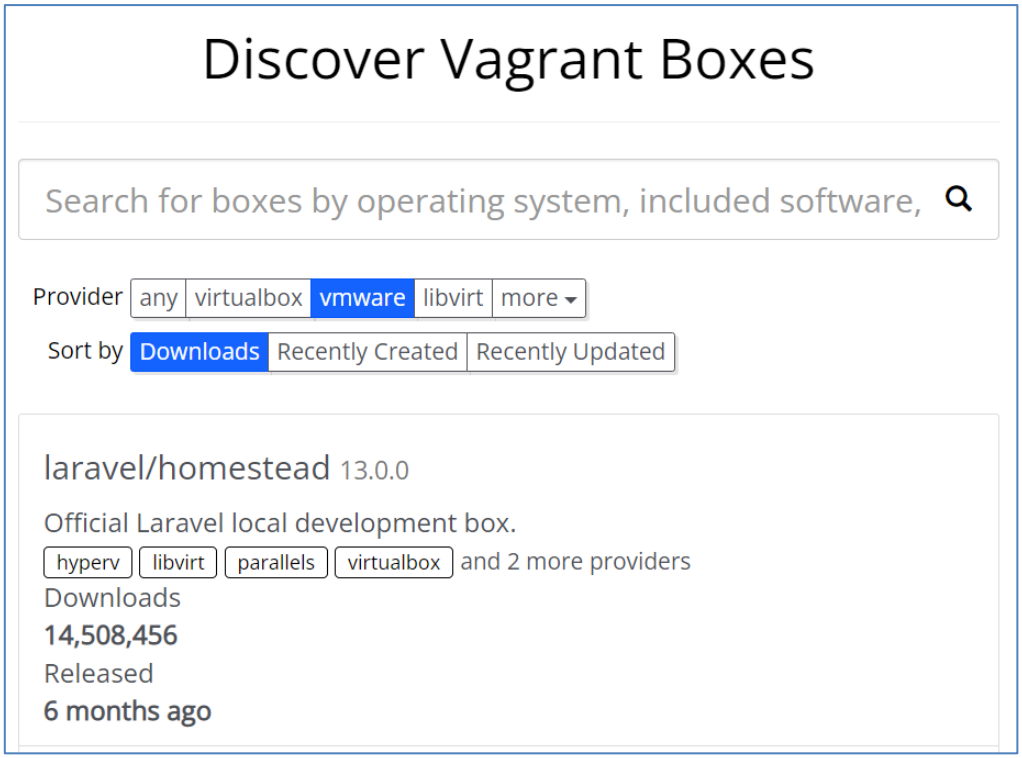
Les boîtes Vagrant sont un moyen pratique de partager et de distribuer des configurations de machine virtuelle, car elles contiennent toutes les informations nécessaires pour reproduire la même configuration de machine virtuelle sur une autre machine ou auprès d'autres membres de votre équipe.

Mais toutes les boxes ne sont pas compatibles avec tous les hyperviseurs, à plus forte raison lorsqu'il s'agit d'une image docker. Vous pouvez télécharger des boîtes Vagrant depuis plusieurs sources, notamment :

- Le site officiel de Vagrant propose une liste de boîtes Vagrant prêtes à l'emploi pour différents systèmes d'exploitation et configurations logicielles : https://app.vagrantup.com/boxes/search.
- Les fournisseurs de cloud computing, tels que AWS, Google Cloud Platform et Microsoft Azure, proposent également des boîtes Vagrant personnalisées pour leurs environnements de cloud.
- Les fournisseurs tiers proposent également des boîtes Vagrant pour différents systèmes d'exploitation et configurations logicielles. Vous pouvez trouver des boîtes Vagrant sur des sites tels que HashiCorp Atlas, Atlas HashiCorp, Vagrant Cloud ou encore sur GitHub.

Pour trouver des boxes compatibles VMware suivez ce lien :

https://app.vagrantup.com/boxes/search?provider=vmware



UTILE

Il est important de noter que toutes les boîtes Vagrant ne sont pas créées égales. Avant de télécharger une boîte, assurez-vous de vérifier les commentaires et les notes de la communauté pour vous assurer qu'elle répond à vos besoins et est de bonne qualité.

En résumé, les boîtes Vagrant sont un élément clé de l'écosystème Vagrant et permettent aux développeurs de créer rapidement et facilement des machines virtuelles personnalisées. En utilisant des boîtes Vagrant préconfigurées ou en créant vos propres boîtes Vagrant, vous pouvez accélérer considérablement la mise en place d'environnements de développement isolés et reproductibles.

3.7 Atelier pratique: Automatiser la création d'un Windows Server

L'objectif de cet atelier est de comprendre comment automatiser la création d'un environnement de développement Windows Server 2022 en utilisant Vagrant. En utilisant Vagrant, vous apprendrez comment définir les paramètres de la machine virtuelle, comment télécharger la box Vagrant pour Windows Server, comment automatiser l'installation de Windows Server, et comment configurer les interfaces réseau.

En automatisant la création de votre environnement de développement Windows Server, vous pourrez gagner du temps, améliorer votre productivité et réduire les erreurs liées à la configuration manuelle de votre environnement de développement.

Cela va vous permettre d'acquérir des compétences en automatisation de l'infrastructure et de mieux comprendre comment Vagrant peut aider à améliorer la gestion des environnements de développement.

3.7.1 Prérequis

Pour automatiser la création d'une machine virtuelle Windows Server en utilisant Vagrant avec VMware comme fournisseur, vous devez tout d'abord installer VMware Workstation pro 17, Vagrant et son plugin.

Ensuite, vous devez créer un fichier Vagrant appelé Vagrantfile qui contient la configuration de votre machine virtuelle. Ce fichier doit spécifier la boîte de la machine virtuelle, le nom d'hôte, l'adresse IP, le fournisseur de virtualisation (VMware Workstation) et d'autres paramètres comme la quantité de mémoire et de processeurs alloués à la machine virtuelle.

Une fois le fichier Vagrant créé, vous pouvez exécuter la commande **vagrant up** pour démarrer la machine virtuelle. Après avoir exécuté cette commande, vous pouvez vous connecter à la machine virtuelle à l'aide de l'interface graphique de VMware Workstation.

3.7.2 Installation de Vagrant

Pour télécharger Vagrant, aller sur le site officiel :

https://www.vmware.com/go/getworkstation-win

3.7.3 Installation du plugin

Une fois vagrant install, vous devrez installer le plugin Vagrant VMware Utility.

Pour installer le plugin, vous pouvez ouvrir un terminal ou une invite de commande et exécuter la commande :

```
vagrant plugin install vagrant-vmware-desktop
```

Cela téléchargera et installera le plugin, et ajoutera le vmware_desktop à Vagrant.

3.7.4 Configuration de VMware

Avant de procéder à la configuration de Ansible, nous devons restaurer par défaut la configuration réseau VMware.

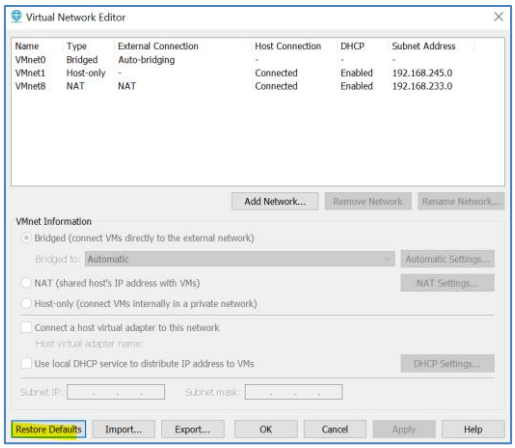
Si vous modifiez votre réseau virtuel NAT (vmnet8), le déploiement pourrait ne pas appliquer certaines configurations.

Pour information voici ma configuration NAT par défaut, si vous n'avez pas la même c'est normal et si vous n'y avez jamais touché, n'y touchez pas ce sera mieux (même si la Gateway est en .2 et que ça vous dérange…), et si vous l'avez modifié, vous devez le restaurez par défaut.

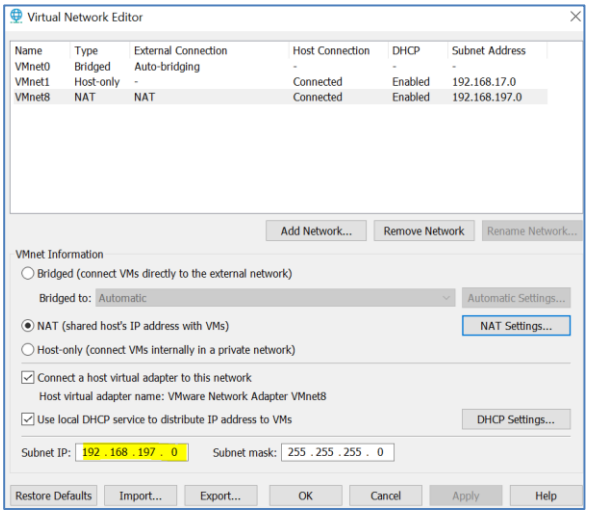
Pour restaurer les paramètres de réseau par défaut de VMware Workstation Pro, vous pouvez suivre les étapes suivantes :

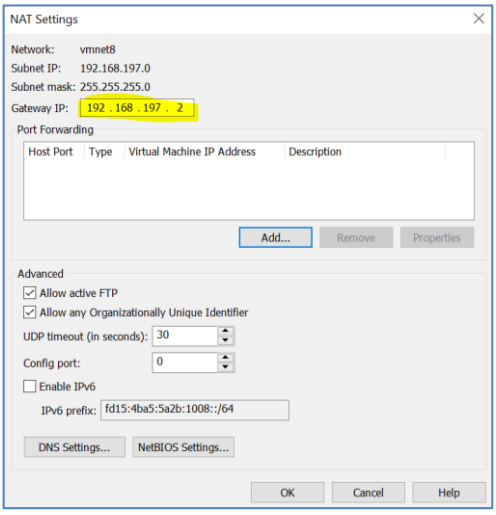
1. Ouvrez le gestionnaire de réseaux virtuels de VMware Workstation Pro en cliquant sur "Edit" > "Virtual Network Editor" dans la barre de menus de l'application.
2. Cliquez sur le bouton "Restore Defaults" dans le coin inférieur droit de la fenêtre.
3. Confirmez que vous souhaitez restaurer les paramètres de réseau par défaut en cliquant sur "Yes" dans la boîte de dialogue de confirmation.

Les paramètres de réseau par défaut seront restaurés, y compris les adaptateurs de réseau virtuel et les sous-réseaux virtuels.



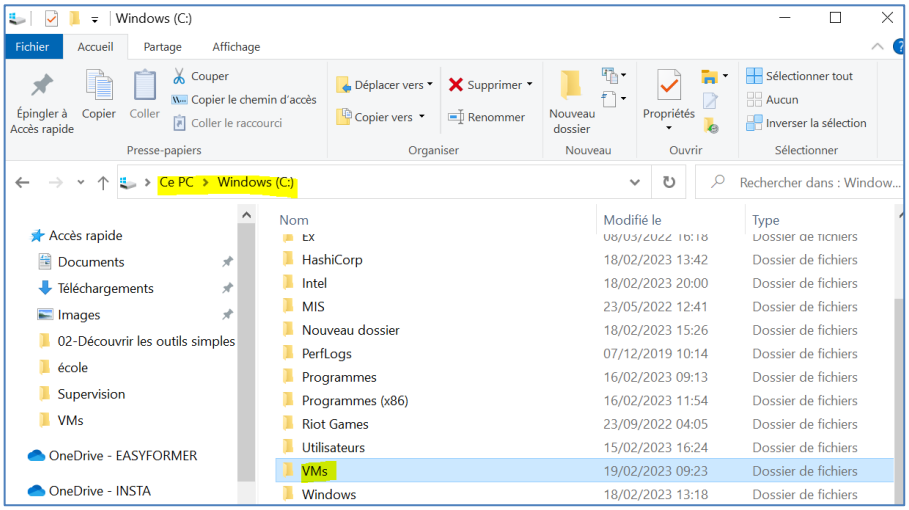
Une fois que les paramètres de réseau par défaut sont restaurés, veuillez noter le réseau et la passerelle du Nat :





3.7.5 Préparer le fichier de configuration

Tout d'abord, nous allons créer un répertoire dédié à notre fichier de configuration Vagrantfile. Moi j'ai choisi de le créer dans : « C:\VMs\ ». Toute la configuration de Vagrant va se faire ici :



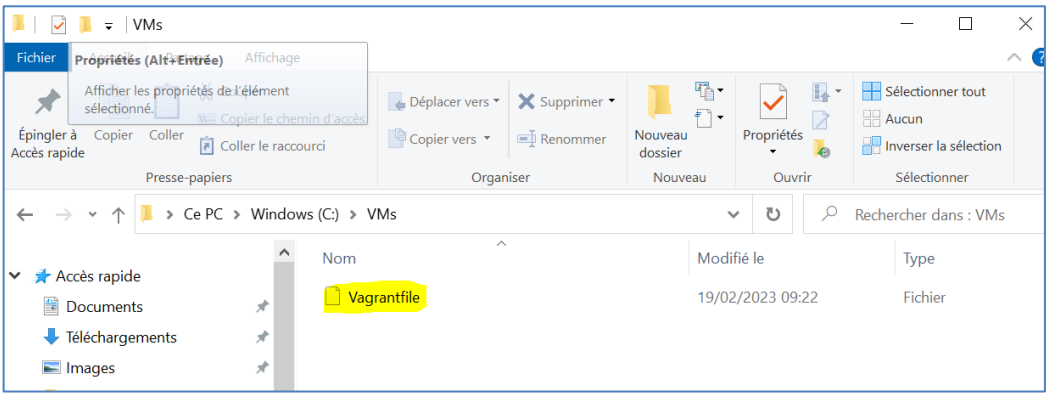
Ensuite on ouvre l'invité de commande cmd et on se déplace dans le répertoire VMs via la commande :

```
cd /d C:\VMs
```

Ensuite on utilise la commande **vagrant init** pour créer un fichier Vagrantfile avec des configurations par défaut pour une box Vagrant spécifique. La commande crée un fichier Vagrantfile dans le répertoire :

```
vagrant init
```

On édite le fichier Vagrantfile :



Ensuite copier-coller les commande ci-dessous, noté qu'il y'a déjà des configurations par défaut intégré dans le fichier, veuillez-les supprimez et copier-coller la configuration ci-dessous :

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
Vagrant.configure("2") do |config|

 config.vm.hostname = "windows-server-vmware"
 config.vm.box = "dstoliker/winserver2022-max"
 config.vm.guest = "windows"
 config.vm.base_mac = "AA:11:CD:12:08:08"
 config.vm.network "private_network", ip: "10.10.10.150", :hostonly => "vmnet2"
 config.vm.provision "shell", inline: <<-SHELL
    netsh.exe int ip set address "Ethernet1" static 192.168.10.150 255.255.255.0
    netsh.exe int ip set address "Ethernet0" static 192.168.197.150 255.255.255.0
192.168.197.2
    netsh.exe interface ip set dns "Ethernet0" static 8.8.8.8 primary
    netsh.exe int ip show addresses
 SHELL
 config.vm.provider "vmware_workstation" do |v|
   v.vmx["displayName"] = "Windows Server 2022"
   v.vmx["memsize"] = "2048"
   v.vmx["numvcpus"] = "2"
   v.vmx["ethernet0.pcislotnumber"] = "33"
   v.gui = true
 end


end
```

Voici une brève explication des configurations présentes dans ce fichier :

- La ligne "**Vagrant.configure("2") do |config|"** définit la version de la configuration Vagrant à utiliser.
- "**config.vm.hostname**" définit le nom d'hôte de la machine virtuelle.
- "**config.vm.box**" spécifie la box Vagrant à utiliser pour créer la machine virtuelle. Dans ce cas, la box "dstoliker/winserver2022-max" est utilisée. Vagrant télécharge automatiquement cette box à partir de l'URL de la box spécifiée lors de l'exécution de la commande "vagrant up".
- "**config.vm.guest**" spécifie le système d'exploitation invité, qui est Windows dans ce cas.
- "**config.vm.base_mac**" définit l'adresse MAC de l'interface réseau de la machine virtuelle.
- "**config.vm.network**" configure l'interface réseau de la machine virtuelle en utilisant une adresse IP statique pour l'interface "Ethernet1" et une adresse IP statique, un masque de sous-réseau et une passerelle pour l'interface "Ethernet0".
- "**config.vm.provision**" exécute une commande shell pour configurer les paramètres réseau de la machine virtuelle :
    - "**netsh.exe int ip set address "Ethernet1" static 192.168.10.150 255.255.255.0"** définit une adresse IP statique pour l'interface Ethernet1 de la machine virtuelle. L'adresse IP est 192.168.10.150, le masque de sous-réseau est 255.255.255.0 et il n'y a pas de passerelle par défaut spécifiée.
    - "**netsh.exe int ip set address "Ethernet0" static 192.168.197.150 255.255.255.0 192.168.197.2**" définit une adresse IP statique pour l'interface Ethernet0 de la machine virtuelle. L'adresse IP est 192.168.197.150, le masque de sous-réseau est 255.255.255.0 et la passerelle par défaut est 192.168.197.2.
    - "**netsh.exe interface ip set dns "Ethernet0" static 8.8.8.8 prima**ry" configure la résolution de noms DNS pour l'interface Ethernet0 en utilisant l'adresse IP 8.8.8.8 du serveur DNS primaire.
    - "**netsh.exe int ip show addresses**" affiche les adresses IP et les paramètres de configuration de toutes les interfaces réseau de la machine virtuelle.
- "**config.vm.provider**" définit les paramètres spécifiques au fournisseur. Dans ce cas, "vmware_workstation" est le fournisseur et des paramètres tels que la taille de la mémoire, le nombre de processeurs et le nom d'affichage sont définis.

3.7.6 Déployer la machine virtuelle

Vous êtes maintenant prêt à démarrer le déploiement automatisé de votre VM.

Lancez PowerShell avec les droits d'administrateur

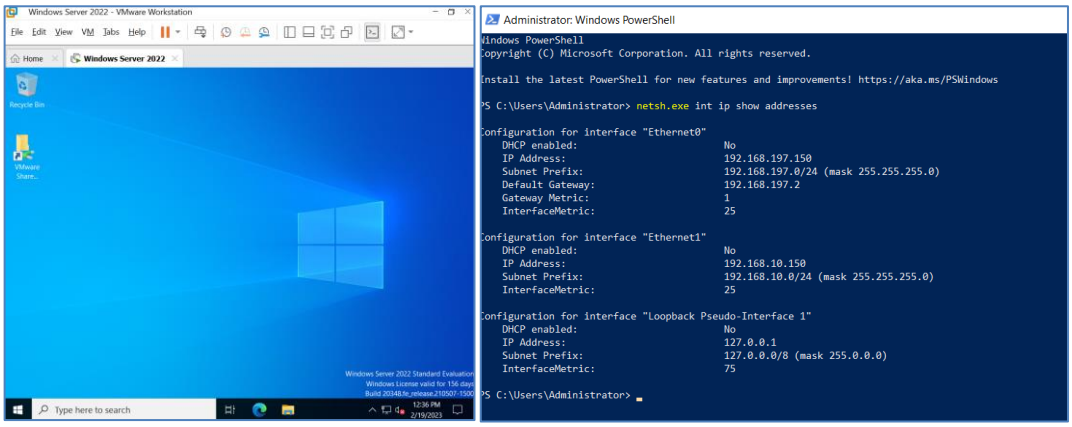
Rendez-vous dans le dossier où se trouve le Vagrantfile. Utilisez pour cela la commande « cd » (pour « Change Directory ») suivie d'un espace et du chemin d'accès complet vers votre fichier Vagrantfile ; dans mon exemple c'est :

```
cd C:\VMs\
```

Lancez maintenant la commande PowerShell suivante :

```
vagrant up
```

Attendez ensuite quelques minutes car il faut que Vagrant télécharge la box Windows Server et qu'il déploie la VM donc soyez patients. Après ces quelques minutes vous arriverez automatiquement sur cet écran avec votre VM toute prête :



Le paramètre pour fixer l'IP de l'interface réseau dans le vmnet1 (Host-only) et le Nat ont bien fonctionné.

