Infrastructure et déploiement

Plan et objectifs

- Comprendre les particularités des différents types d’environnements entourant le développement et les opérations d’une application web
- Comprendre les différents environnements logiciels et matériels pour exécuter une application web
- Type d’infrastructure d’infonuagique - IaaS, PaaS, FaaS
- Pipeline automatisé CI/CD

Environnements

Le processus de développement d'une application web requiert différents environnements qui ont chacun des caractéristiques et des objectifs particuliers.

Les environnements typiques sont:

- développement
- test
- mise en scène
- production

Environnement de développement

Chaque développeur possède son propre environnement de développement qui se retrouve souvent sur son ordinateur.

Le développeur a le contrôle de cet environnement.

L'environnement de développement est

- individuel - non partagé
- unique - car souvent modifié

Il contient les outils pour développer, exécuter et tester l'application web.

Pour accélérer l'intégration des nouveaux développeurs au projet (ou pour réparer rapidement un environnement) , la création de l'environnement de développement doit être documentée et, si possible, automatisée.

- *Ça marche sur ma machine!*

Environnement de tests

Pour les tests, on désire un environnement stable et contrôlé.

Cet environnement doit pouvoir être re-créé facilement à partir de zéro afin d’ éviter que le résultat des tests passés affecte le résultat des nouveaux tests.

Cet environnement contient l’application et les outils pour la tester.

Il peut être partagé.

L’objectif est de valider une version spécifique de l’application en isolation.

Environnement utilisé pour

- intégration continue et tests automatisés
- test manuels (souvent partagé dans ce cas)
- test système
- test de performance

Environnement de mise en scène (staging)

Nous devons être capable de valider certains changements dans un environnement qui perdure dans le temps et dont les caractéristiques et les données ressemblent à celles de la production.

L'environnement de mise en scène (staging) permet donc de valider:

- le processus de mise à jour (update)
- les changements de configuration
- l'évolution de l'application (migration des données, rétrocompatibilité, etc)
- avec des données semblable à la production

Il est rarement utile d'avoir plus d'un environnement de mise en scène et, normalement, cet environnement évolue en parallèle avec l'environnement de production.

Souvent utilisé pour des démos.

Environnement de production

L’environnement de production est responsable de l’instance de l’application web qui est utilisée par les utilisateurs.

Il doit être stable et les accès aux composants du système sont limités et sécurisés.

Cet environnement doit toujours être disponible par les utilisateurs et il est surveillé constamment.

Lorsqu’un problème est détecté, une action est prise immédiatement.

L’état est régulièrement sauvegardé (backup), car on ne peut se permettre de repartir à zéro.

Selon le type d’application, il peut y avoir un environnement de production par client ou un seul pour tous les utilisateurs.

Infrastructure de déploiement

L’infrastructure sur lequel l’application web est exécutée varie selon le type d’environnement.

Différentes options sont possibles:

- localement sur la machine du développeur
- sur une machine virtuelle
- dans un conteneur
- sur une plateforme d’infonuagique (cloud)

Virtualisation

Pour pouvoir maximiser l’utilisation des ressources matérielles, il est possible d’utiliser une solution de virtualisation.

L’objectif est de créer plusieurs machines (virtuelles) différentes et indépendantes qui s'exécutent sur les mêmes ressources matérielles.

Il existe différentes options de virtualisation.

Machine virtuelle

Pour pouvoir créer plusieurs machines virtuelles distinctes et isolées les unes des autres, il est possible d'introduire un logiciel appelé **hyperviseur** qui offre une abstraction du matériel.

Il existe 2 type d'hyperviseurs:

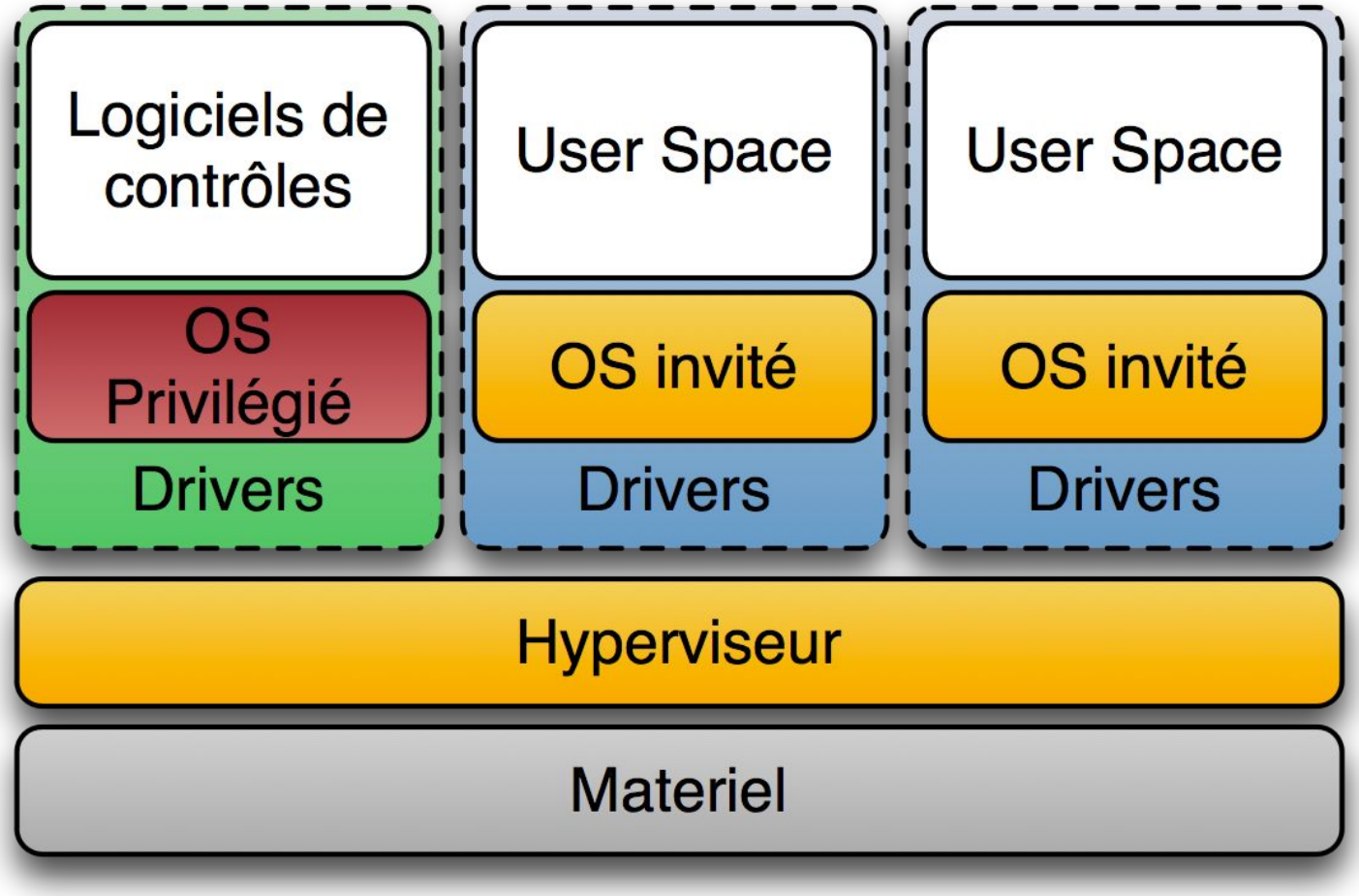
- Type 1 - *Bare Metal* - S'exécute directement sur le matériel (e.g. Oracle VM Server, Xen)
- Type 2 - *Hosted* - S'exécute dans un OS (e.g Oracle VirtualBox, VMware Workstation)

Chaque machine virtuelle s'exécute *sur* l'hyperviseur et possède ses propres ressources virtuelles (CPU, RAM, etc). Elle ne peut pas accéder aux autres machines virtuelles présentes sur la même machine physique.

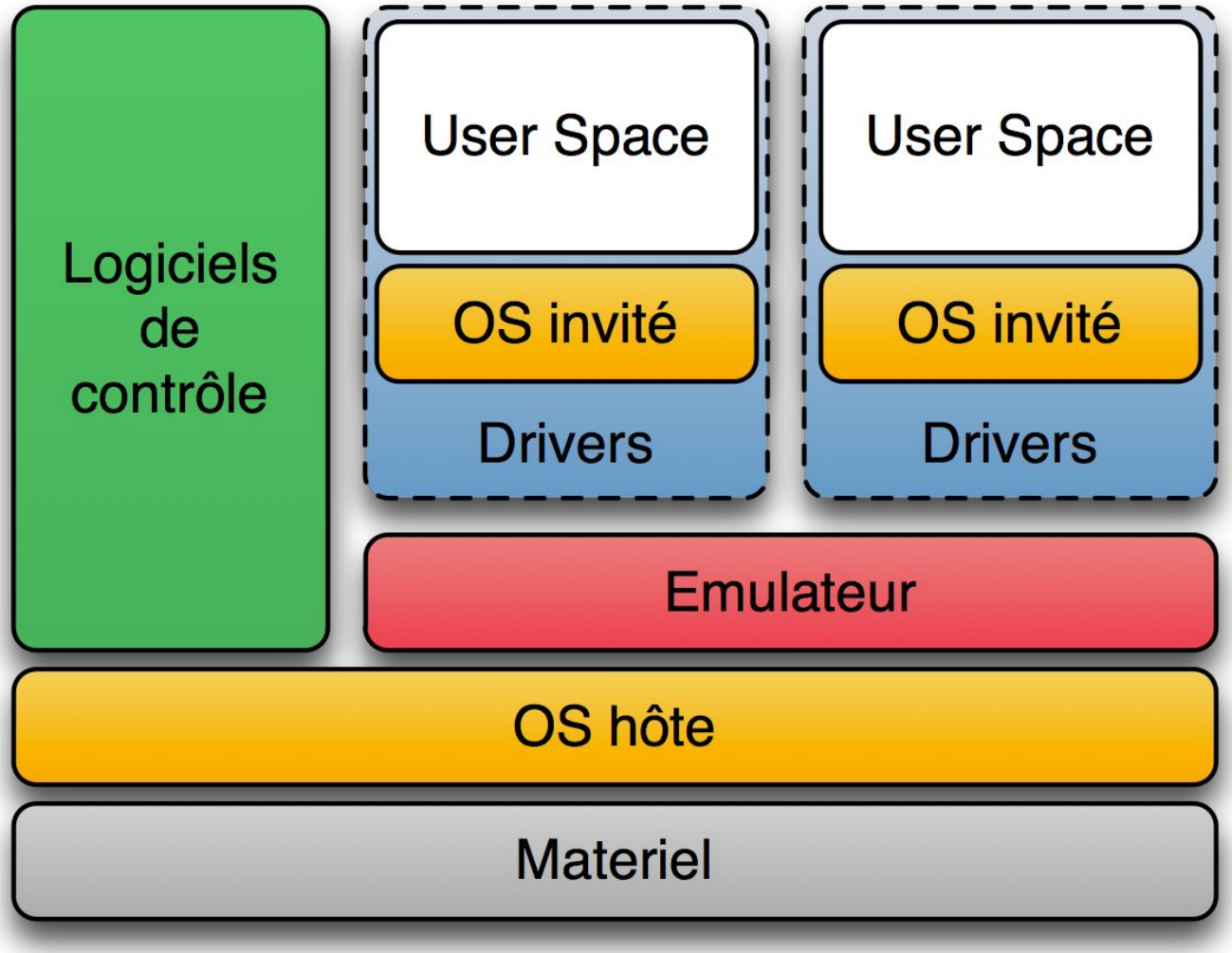
Chaque machine virtuelle peut exécuter un système d'exploitation différent.

Les machines virtuelles peuvent aussi être copiées ou transférées d'un hôte à l'autre.

Machine virtuelle - Type 1



Machine virtuelle - Type 2



Machine virtuelle - Désavantages

Parce que les machines virtuelles contiennent une copie complète du système d'exploitation, la taille de chaque machine peut être grande (plusieurs GBs).

Conséquemment, le temps pour démarrer une machine virtuelle est non négligeable.

Il existe une alternative plus légère pour créer plusieurs environnements indépendants s'exécutant sur les mêmes ressources matérielles: les conteneurs.

Conteneur

Un système de conteneurs utilise une abstraction au niveau du système d'exploitation (qui est partagé par tous les conteneurs).

Le conteneur contient alors seulement l'application à exécuter et l'ensemble de ses dépendances.

L'image d'un conteneur est beaucoup plus petite comparé à une machine virtuelle qui contient l'OS.

Pour une architecture par services, l'utilisation de conteneur permet de réduire de beaucoup les ressources nécessaires.

Si le conteneur est utilisé comme unité de déploiement, un développeur peut exécuter plusieurs conteneurs en parallèle pour avoir une version locale du système complet.

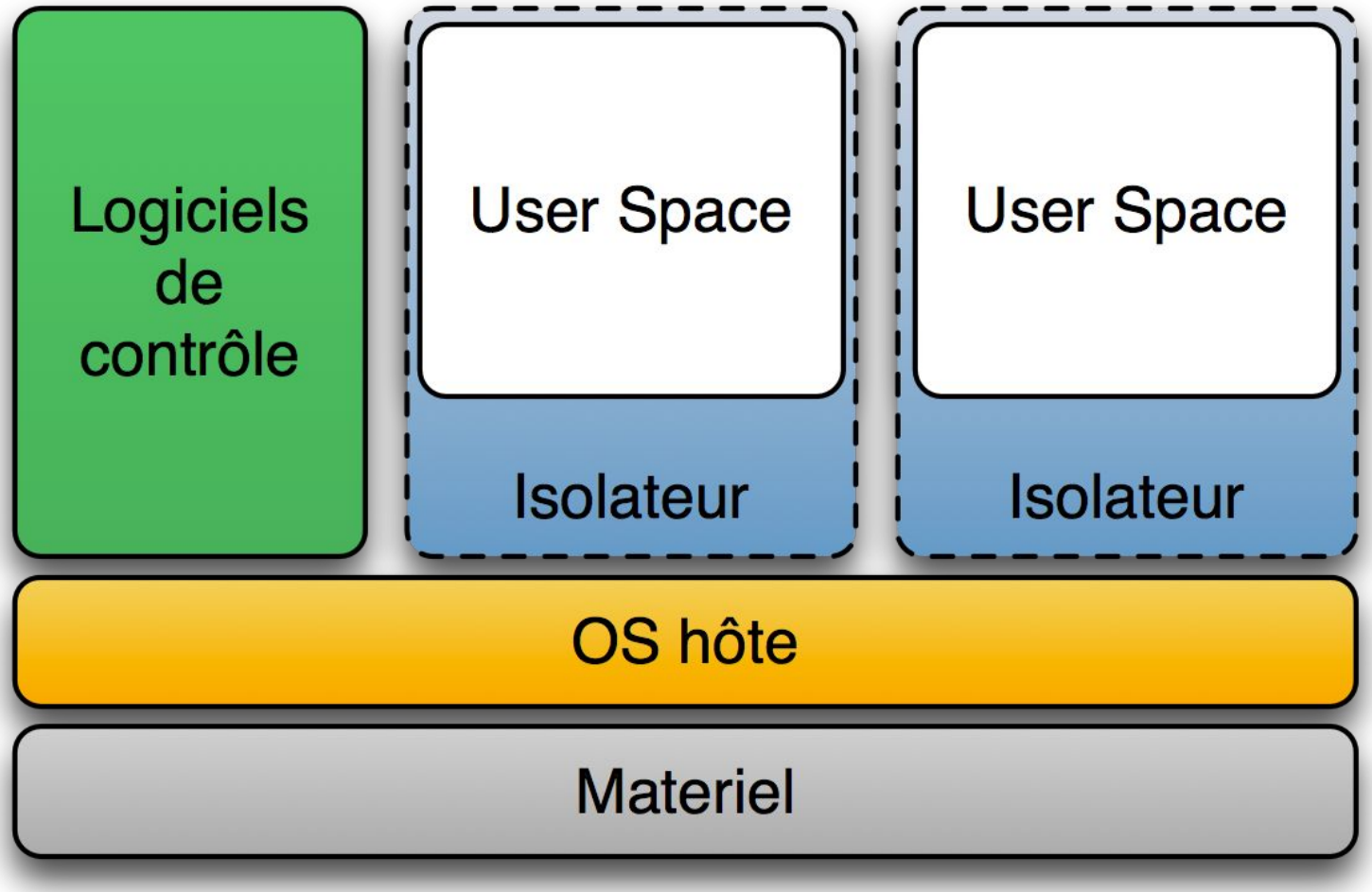
Image vs Conteneur

Une **image** est une définition immuable qui sert de gabarit pour créer les conteneurs.

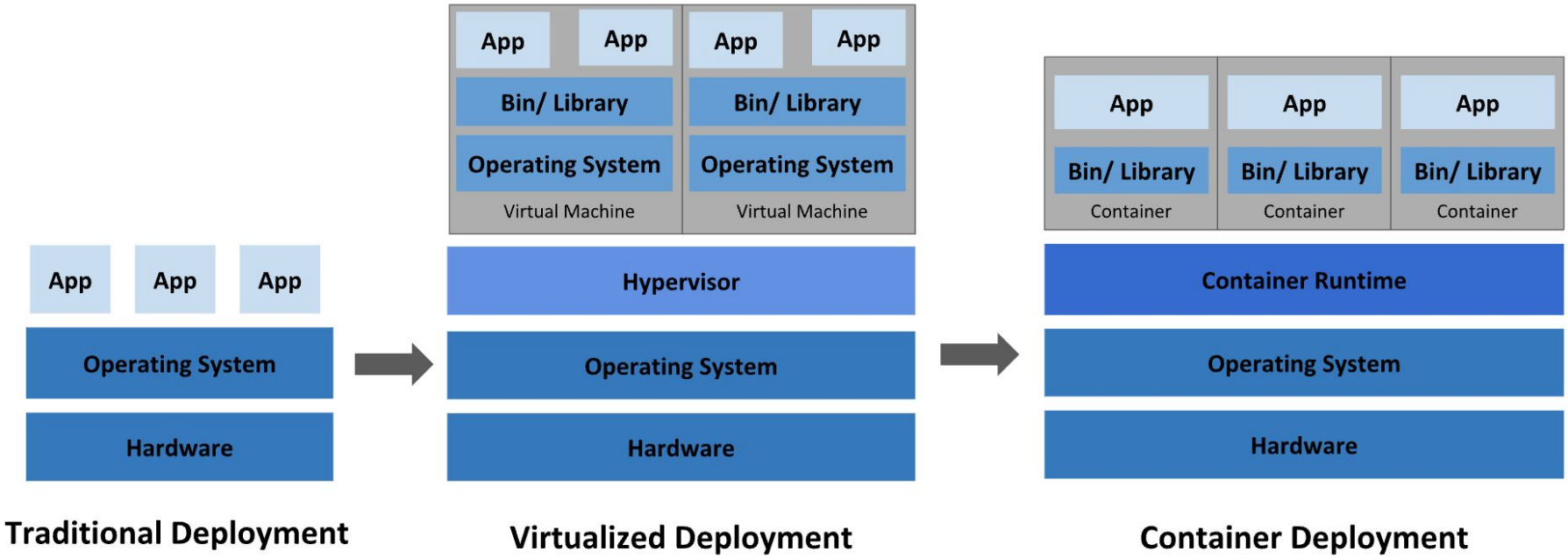
Un **conteneur**

- est une instance exécutable d'une image (avec un nom)
- il est possible de créer, démarrer, arrêter, déplacer et effacer un conteneur
- un conteneur peut s'exécuter localement, sur une machine virtuelle ou déployé dans le cloud
- un conteneur est isolé des autres conteneurs (namespace, cgroup)

Conteneur


Logiciels de contrôle
User Space
Isolateur
User Space
Isolateur
OS hôte
Materiel

Résumé



https://kubernetes.io/docs/concepts/overview/

Docker

Docker (https://www.docker.com/) est une plateforme qui vous permet de créer des images, de les partager et d'exécuter des conteneurs.

1. Définir les étapes de de création d'une image en définissant un Dockerfile
2. Construire une image en utilisant docker (docker build)
3. Exécuter un conteneur (docker run)
4. Publier votre image (docker push)

Orchestration de conteneurs

Lorsqu'on doit gérer un système bâti selon une architecture par service et que le nombre de services (et d'instances pour chaque service) est élevé, il devient rapidement ardu de déployer, mettre à jour, mettre à l'échelle, démarrer un conteneur, etc.

Il existe donc des solutions d'orchestration de conteneurs qui permettent de simplifier la tâche.

Voici certaines des fonctionnalités des solutions d'orchestration de conteneurs:

- découverte de services et balancement de charge
- redémarrer un conteneur en panne
- modifier le nombres d'instances
- gestion de configuration
- mettre à jour de façon graduelle un système

Kubernetes

Kubernetes (https://kubernetes.io/) est un outil d'orchestration de conteneurs (open source) puissant et généralement offert par les fournisseurs de service d'infonuagique.

Un cluster Kubernetes est composé de

- Noeuds - machines virtuelles
- Chaque noeud contient 1 ou plusieurs Pods
- Un Pod est la plus petite unité déployable et gérable
- Chaque Pod contient un ou plusieurs conteneurs

Kubernetes utilise différents concepts pour travailler à un niveau d'abstraction plus élevé:

- Service - interface réseau et balancement de charge sur plusieurs Pods
- Volume - interface pour stocker des données
- Deployment - description de l'état désirée du cluster (nb replicas, etc)

Déploiement en infonuagique

Il est possible de gérer soit même les serveurs sur lesquels l'application web sera exécutée ou de louer un espace dans un centre de données.

Ceci offre plusieurs avantages au niveau du contrôle du matériel, du logiciel et des coûts.

Par contre, cette solution implique que vous êtes responsable de la gestion du matériel (achat, installation, réparation).

Vous êtes aussi responsable de la configuration et des mises à jour du système d'exploitation ce qui demande des compétences spécifiques qui ne sont souvent pas liées directement à l'application à bâtir.

C'est pourquoi les solutions de déploiement en infonuagique sont intéressantes.

Infrastructure en tant que service (IaaS)

Dans un modèle d'infrastructure en tant que service (Infrastructure as a Service), le fournisseur gère le réseau et la partie matériel (machine et disk, incluant la virtualisation) et, parfois, le système d’exploitation des serveurs.

Vous n’avez donc qu’à gérer la couche de logiciel spécifique à votre application web: serveur web, environnement d’exécution, etc.

Exemple: AWS, Azure, Google Compute Engine

Plateforme en tant que service (PaaS)

Dans une solution de plateforme en tant que service (Platform as a Service), le fournisseur gère tout sauf votre application.

Une solution PaaS peut même effectuer la mise à l’échelle automatiquement.

Exemple: AWS Elastic Beanstalk, Google App Engine

Fonction en tant que service (FaaS)

Dans une solution de fonction en tant que service (Function as a Service), le fournisseur gère tout et vous n’avez qu’à déployer des morceaux de code qui seront exécutés selon certains déclencheurs (appel HTTP, cron, modification dans firestore, etc).

La solution FaaS effectue la mise à l’échelle automatiquement.

Exemple: AWS Lambda, Google Cloud Functions

Logiciel en tant que service (SaaS)

Pour une solution de logiciel en tant que service (Software as a Service), le fournisseur gère la gestion de l’ensemble du matériel et du logiciel.

Il suffit ensuite de s’y connecter (e.g. API, base de données) ou simplement d’utiliser le logiciel (e.g. une application de gestion de site web).

Coûts

Le coût des solutions infonuagiques varie en fonction de plusieurs paramètres:

- caractéristiques des machines (CPU, RAM, etc)
- bande passante
- utilisation CPU
- uptime
- utilisation RAM
- espace disque utilisé
- etc

Il n'est donc pas toujours facile de prévoir les coûts. De plus, un changement dans le code de l'application ou dans son patron d'utilisation peut amener des modifications au niveau des coûts.

Il est important de développer un modèle de coûts pour éviter les surprises.

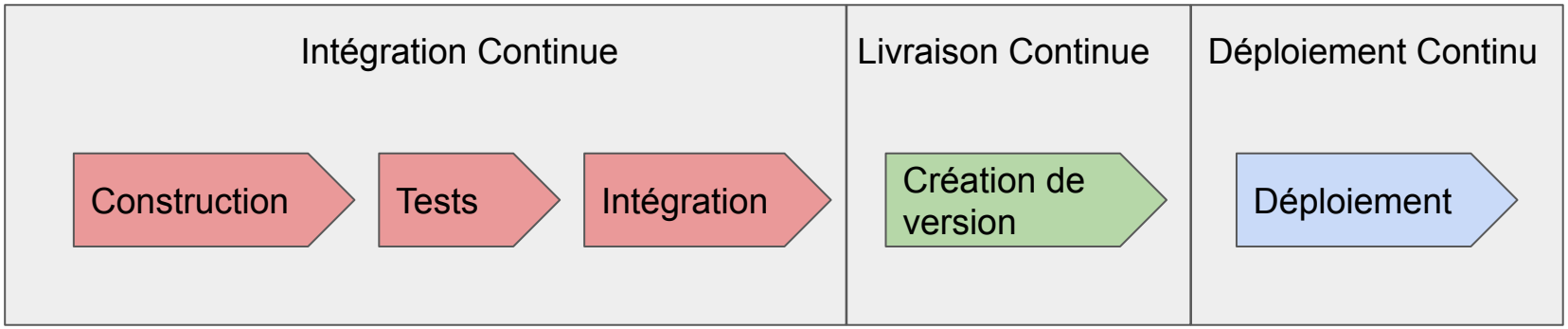
Exemple de calculateur: https://cloud.google.com/products/calculator?hl=fr

Mariage

Lorsqu'on choisit une solution infonuagique pour déployer notre application, l'application devient souvent fortement couplée avec l'écosystème du fournisseur choisi.

Il faut donc en être conscient à chaque étape du développement, car ce peut être un risque non négligeable.

Pipeline CI/CD



Processus d’intégration

Historiquement, les développeurs pouvaient travailler longtemps sur une ou plusieurs fonctionnalités avant d’intégrer leurs changements dans le système de gestion de version.

Cela pouvait être une conséquence de l’outil de gestion de versions utilisé (qui pouvait rendre le processus d’intégration lourd et incitait à limiter les intégrations) ou une problématique au niveau du processus de développement de logiciel (long cycle de développement).

L’intégration était souvent un moment pénible juste avant la création d’une nouvelle version, où l'équipe de développement devait trouver comment régler correctement tous les conflits qui s’étaient accumulés depuis le dernier point d’intégration.

Intégration continue

*“L'**intégration continue** est un ensemble de pratiques utilisées en génie logiciel consistant à **vérifier à chaque modification de code source que le résultat des modifications ne produit pas de régression dans l'application développée**.*

Le principal but de cette pratique est de détecter les problèmes d'intégration au plus tôt lors du développement.”

- https://fr.wikipedia.org/wiki/Int%C3%A9gration_continue

L’utilisation d’outils de gestion de version moderne simplifie l’intégration de code. Ceci, combiné aux tests unitaires exécutés automatiquement, à permis de simplifier le processus d’intégration.

L’intégration continue constitue un élément important des techniques modernes de développement de logiciel.

Revue de code

Les revues de code sont un élément du processus de développement qui ne peut pas être automatisé.

Malgré tout, les revues de code ont une grande valeur:

- elle permettent de trouver des bogues (et les corrections)
- elle assure une consistance dans la base de code
- elle partage de connaissances et bonnes pratiques
- elle augmente la cohésion de l'équipe

Pour ce faire, il faut prendre le temps de bien réviser le code et de commenter de façon constructive.

Il faut aussi apprendre à se dissocier personnellement du code que l'on a écrit.

Processus de livraison

Historiquement, le cycle de livraison de version pouvait être très long (plusieurs mois).

Une nouvelle version contenait alors tellement de changements qu’il était rare qu’une nouvelle version soit stable dès le départ.

Livraison continue

*“La **livraison continue** (en anglais : continuous delivery, CD) est une approche d’ingénierie logicielle dans laquelle l**es équipes produisent des logiciels dans des cycles courts, ce qui permet de le mettre à disposition à n’importe quel moment.** Le but est de construire, tester et diffuser un logiciel plus rapidement.”*

- https://fr.wikipedia.org/wiki/Livraison_continue

Processus de déploiement

Historiquement, déployer une nouvelle version était toujours un moment stressant.

La raison étant que, comme le cycle entre les versions était long, le nombre de modifications s'accumulaient pendant longtemps avant d'être déployées.

Souvent, la première version d'un nouveau cycle introduisait, malgré les longs cycles de tests, de nombreuses instabilités dans le système au grand malheur de l'équipe d'opération (et de l'équipe de développement).

Déploiement continu

*“Le **déploiement continu** (en anglais : continuous deployment, CD) est une approche d'ingénierie logicielle dans laquelle les **fonctionnalités logicielles sont livrées fréquemment par le biais de déploiements automatisés**.”*

https://fr.wikipedia.org/wiki/D%C3%A9ploiement_continu#:~:text=Le%20d%C3%A9ploiement%20continu%20.

En déployant de façon régulière, le nombre de changements entre deux versions est limité. Les chances qu’une instabilité soit présente sont limitées et si jamais c’est le cas, il est plus rapide de trouver la cause, car le nombre de modifications à analyser est restreint.