Comprendre et implémenter des microfrontends sur AWS

AWS Conseils prescriptifs

AWS Conseils prescriptifs: Comprendre et implémenter des microfrontends sur AWS

Table of Contents

Comprendre et implémenter des micro-frontends sur AWS

Amazon Web Services (contributeurs)

Juillet 2024 (historique du document)

Alors que les entreprises recherchent l'agilité et l'évolutivité, l'architecture monolithique classique devient souvent un goulot d'étranglement, entravant le développement et le déploiement rapides. Les micro-frontends atténuent ce problème en décomposant les interfaces utilisateur complexes en composants indépendants plus petits qui peuvent être développés, testés et déployés de manière autonome. Cette approche améliore l'efficacité des équipes de développement et facilite la collaboration entre le backend et le frontend, favorisant ainsi l' end-to-end alignement des systèmes distribués.

Ce guide normatif est conçu pour aider les responsables informatiques, les propriétaires de produits et les architectes de divers domaines professionnels à comprendre l'architecture micro-frontend et à créer des applications micro-frontend sur Amazon Web Services ().AWS

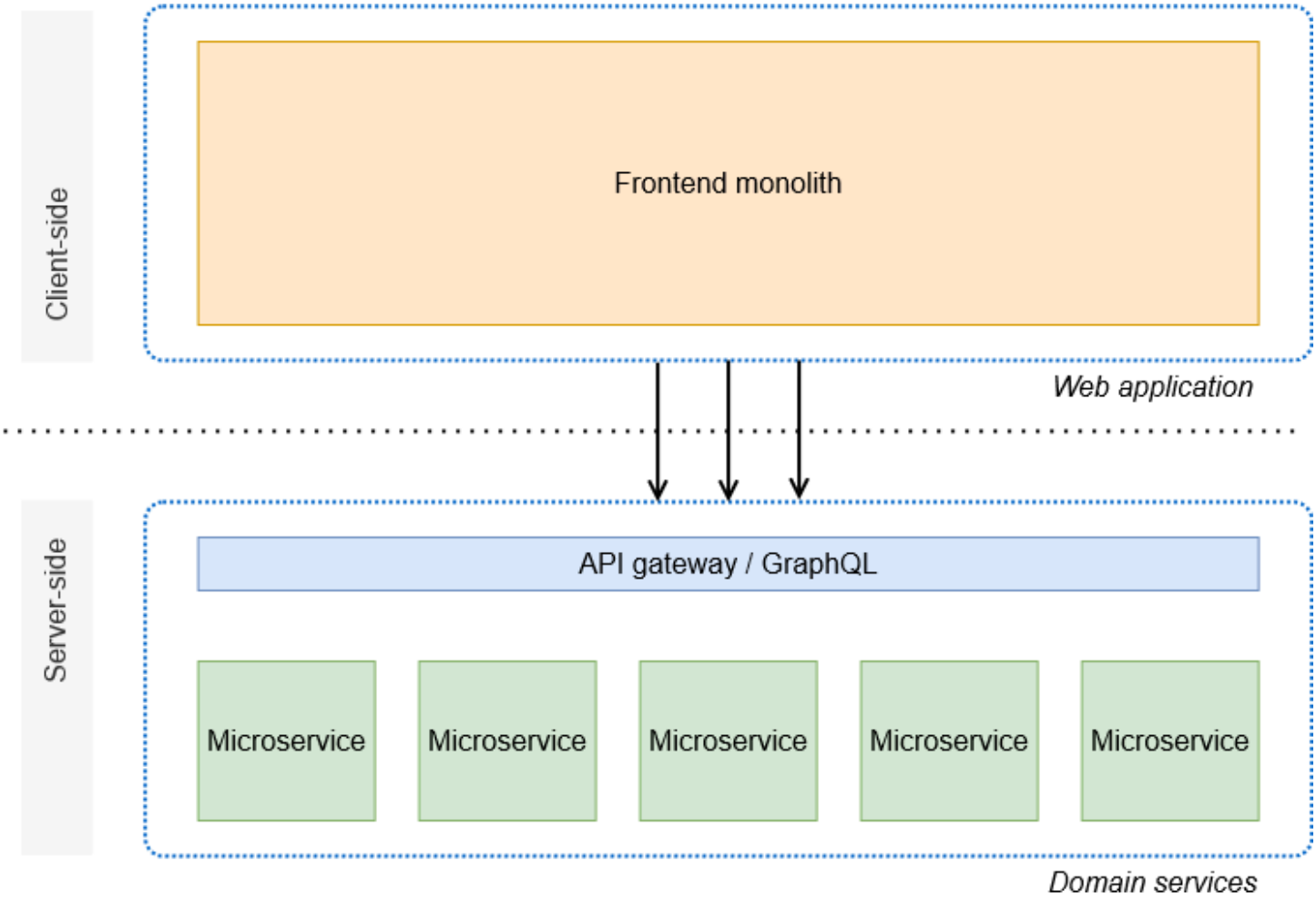
Présentation

Les micro-frontends sont une architecture basée sur la décomposition des frontends d'applications en artefacts développés et déployés indépendamment. Lorsque vous divisez de grandes interfaces en artefacts logiciels autonomes, vous pouvez encapsuler la logique métier et réduire les dépendances. Cela permet une livraison plus rapide et plus fréquente des incréments de produits.

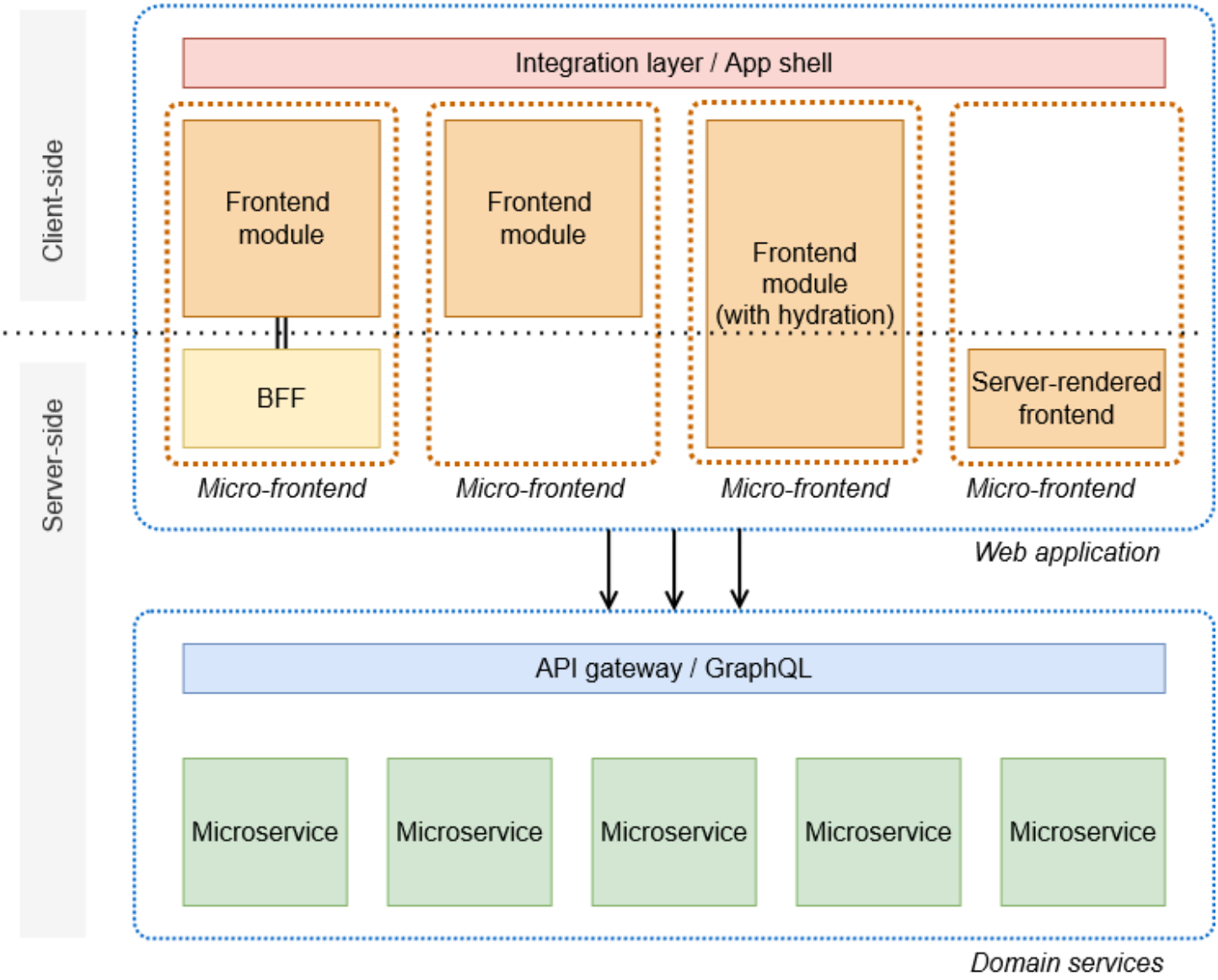
Les micro-frontends sont similaires aux microservices. En fait, le terme micro-frontend est dérivé du terme microservice et vise à exprimer la notion de microservice en tant que frontend. Alors qu'une architecture de microservices combine généralement un système distribué dans le backend avec un frontend monolithique, les micro-frontends sont des services frontaux distribués autonomes. Ces services peuvent être configurés de deux manières :

- Interface uniquement, intégration à une couche d'API partagée derrière laquelle s'exécute une architecture de microservices
- Full-stack, ce qui signifie que chaque micro-frontend possède sa propre implémentation de backend.

Le schéma suivant montre une architecture de microservices traditionnelle, avec un monolithe frontal qui utilise une passerelle d'API pour se connecter aux microservices principaux.



Le schéma suivant montre une architecture de micro-frontend avec différentes implémentations de microservices.



Comme le montre le schéma précédent, vous pouvez utiliser des micro-frontends avec des architectures de rendu côté client ou côté serveur :

- Les micro-frontends rendues côté client peuvent directement consommer les API exposées par une API Gateway centralisée.
- L'équipe peut créer un backend-for-frontend (BFF) dans le contexte délimité afin de réduire le niveau de conversation entre le frontend et les API.
- Du côté serveur, les micro-frontends peuvent être exprimés avec une approche côté serveur augmentée côté client en utilisant une technique appelée hydratation. Lorsqu'une page est affichée par le navigateur, la page associée JavaScript est hydratée pour permettre les interactions avec les éléments de l'interface utilisateur, par exemple en cliquant sur un bouton.

- Les micro-frontends peuvent s'afficher sur le backend et utiliser des hyperliens pour rediriger vers une nouvelle partie d'un site Web.

Les micro-frontends conviennent parfaitement aux organisations qui souhaitent effectuer les opérations suivantes :

- Évoluez avec plusieurs équipes travaillant sur le même projet.
- Optez pour la décentralisation de la prise de décision, en permettant aux développeurs d'innover dans les limites des systèmes identifiés.

Cette approche réduit considérablement la charge cognitive des équipes, car celles-ci deviennent responsables de parties spécifiques du système. Cela améliore l'agilité de l'entreprise, car des modifications peuvent être apportées à une partie du système sans perturber le reste.

Les micro-frontends constituent une approche architecturale distincte. Bien qu'il existe différentes manières de créer des micro-frontends, elles ont toutes des points communs :

- Une architecture de micro-frontend est composée de plusieurs éléments indépendants. La structure est similaire à la modularisation qui se produit avec les microservices sur le backend.
- Un micro-frontend est entièrement responsable de l'implémentation du frontend dans son contexte limité, qui comprend les éléments suivants :
  - Interface utilisateur
  - Données
  - État ou session
  - Logique métier
  - Flux

Un contexte délimité est un système interne cohérent avec des limites soigneusement conçues qui déterminent ce qui peut entrer et sortir. Un micro-frontend doit partager le moins possible de logique métier et de données avec d'autres micro-frontends. Partout où le partage doit avoir lieu, il s'effectue via des interfaces clairement définies, telles que des événements personnalisés ou des flux réactifs. Cependant, lorsqu'il s'agit de préoccupations transversales telles qu'un système de conception ou des bibliothèques de journalisation, le partage intentionnel est le bienvenu.

Un modèle recommandé consiste à créer des micro-frontends en faisant appel à des équipes interfonctionnelles. Cela signifie que chaque micro-frontend est développé par la même

équipe travaillant du backend au frontend. La propriété de l'équipe est cruciale, du codage à l'opérationnalisation du système en production.

Ce guide ne vise pas à recommander une approche en particulier. Il aborde plutôt les différents modèles, les meilleures pratiques, les compromis et les considérations architecturales et organisationnelles.

Concepts fondamentaux

L'architecture du micro-frontend est fortement inspirée de trois concepts architecturaux antérieurs :

- La conception axée sur le domaine est le modèle mental permettant de structurer des applications complexes en domaines cohérents.
- Les systèmes distribués constituent une approche permettant de créer des applications sous la forme de sous-systèmes faiblement couplés, développés indépendamment et exécutés sur leur propre infrastructure dédiée.
- Le cloud computing est une approche permettant de gérer une infrastructure informatique sous forme de services avec un pay-as-you-go modèle.

Conception axée sur le domaine

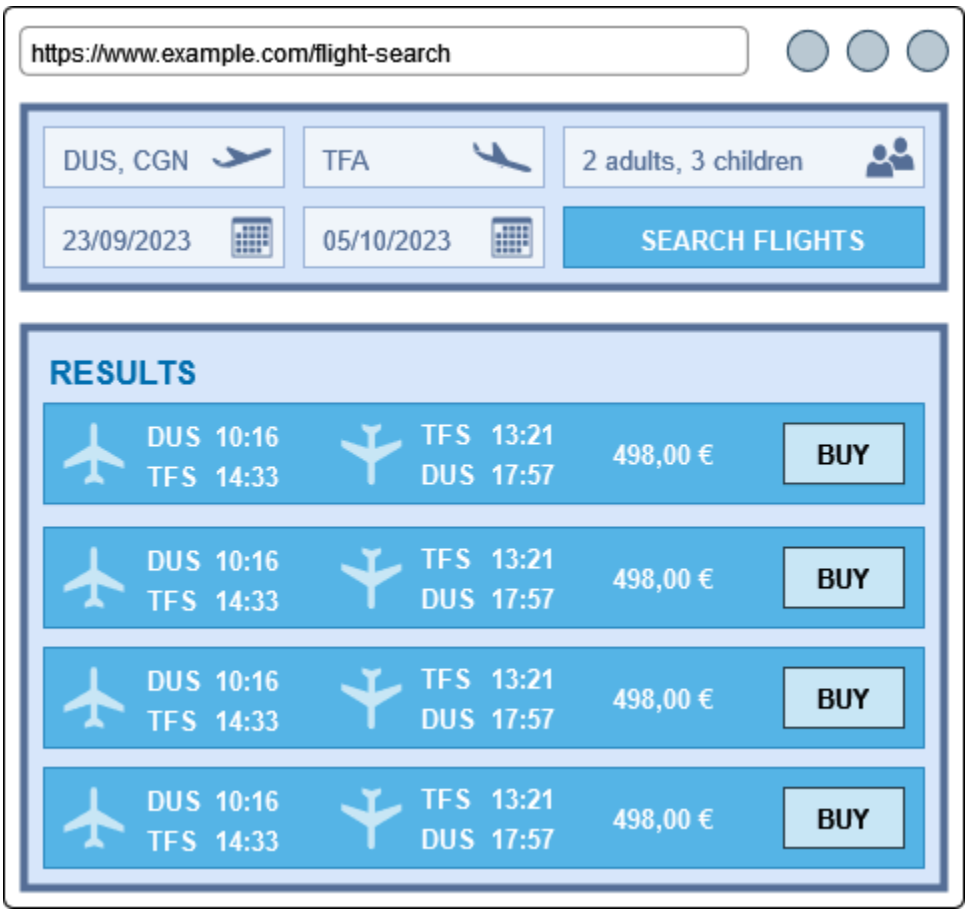
Le design piloté par domaine (DDD) est un paradigme développé par Eric Evans. Dans son livre de 2003 intitulé Domain-Driven Design : Tackling Complexity in the Heart of Software, Evans postule que le développement de logiciels doit être guidé par des préoccupations commerciales plutôt que par des préoccupations techniques. Evans propose que les projets informatiques développent d'abord un langage omniprésent qui aide les experts techniques et spécialisés à trouver une compréhension commune. Sur la base de ce langage, ils peuvent formuler un modèle mutuellement compréhensible de la réalité commerciale.

Aussi évidente que puisse être cette approche, de nombreux projets logiciels souffrent d'un décalage entre l'entreprise et l'informatique. Ces déconnexions sont souvent à l'origine de malentendus importants, qui se traduisent par des dépassements de budget, une baisse de la qualité ou l'échec du projet.

Evans introduit plusieurs autres termes importants, dont le contexte limité. Un contexte limité est un segment autonome d'une application informatique de grande envergure contenant la solution ou la mise en œuvre répondant exactement à un problème commercial. Une application de grande envergure sera composée de plusieurs contextes délimités qui sont faiblement couplés par le biais de modèles d'intégration. Ces contextes délimités peuvent même avoir leurs propres dialectes de la langue omniprésente. Par exemple, un utilisateur dans le contexte du paiement d'une application peut avoir des aspects différents de ceux d'un utilisateur dans le contexte de la livraison, car la notion d'expédition ne serait pas pertinente lors du paiement.

Evans ne définit pas la taille ou la taille d'un contexte délimité. La taille est déterminée par le projet logiciel et peut évoluer au fil du temps. Les bons indicateurs des limites d'un contexte sont le degré de cohésion entre les entités (objets du domaine) et la logique métier.

Dans le contexte des micro-frontends, la conception axée sur le domaine peut être illustrée par l'exemple d'une page Web complexe telle qu'une page de réservation de vol.



Sur cette page, les principaux éléments de base sont un formulaire de recherche, un panneau de filtres et la liste des résultats. Pour identifier les limites, vous devez identifier des contextes fonctionnels indépendants. Tenez également compte des aspects non fonctionnels, tels que la réutilisabilité, les performances et la sécurité. L'indicateur le plus important des « choses qui vont de pair » est leur mode de communication. Si certains éléments d'une architecture doivent communiquer fréquemment et échanger des informations complexes, ils partagent probablement le même contexte limité.

Les éléments individuels de l'interface utilisateur tels que les boutons ne sont pas des contextes limités, car ils ne sont pas indépendants du point de vue fonctionnel. De plus, la page entière ne convient pas à un contexte limité, car elle peut être décomposée en contextes indépendants plus

petits. Une approche raisonnable consiste à traiter le formulaire de recherche comme un contexte limité et à traiter la liste des résultats comme le second contexte délimité. Chacun de ces deux contextes délimités peut désormais être implémenté en tant que micro-frontend distinct.

Systèmes distribués

Pour faciliter la maintenance et favoriser la capacité d'évolution, la majorité des solutions informatiques non triviales sont modulaires. Dans ce cas, la modularité signifie que les systèmes informatiques sont constitués d'éléments de base identifiables qui sont découplés par des interfaces afin de séparer les préoccupations.

En plus d'être modulaires, les systèmes distribués doivent être des systèmes indépendants à part entière. Dans un système simplement modulaire, chaque module est parfaitement encapsulé et expose ses fonctions par le biais d'interfaces, mais il ne peut pas être déployé indépendamment ni même fonctionner seul. De plus, les modules suivent généralement le même cycle de vie que les autres modules faisant partie du même système. Les éléments constitutifs d'un système distribué, en revanche, ont chacun leur propre cycle de vie. En appliquant le paradigme de conception axé sur le domaine, chaque élément de base concerne un domaine ou un sous-domaine commercial et vit dans son propre contexte délimité.

Lorsque les systèmes distribués interagissent pendant la phase de construction, une approche courante consiste à développer des mécanismes permettant d'identifier rapidement les problèmes. Par exemple, vous pourriez adopter des langages dactylographiés et investir massivement dans les tests unitaires. Plusieurs équipes peuvent collaborer au développement et à la maintenance de modules, souvent distribués sous forme de bibliothèques que les systèmes peuvent utiliser avec des outils tels que npm, Apache Maven et NuGet pip.

Pendant l'exécution, les systèmes distribués en interaction sont généralement détenus par des équipes individuelles. La consommation de dépendances entraîne une complexité opérationnelle en raison de la gestion des erreurs, de l'équilibrage des performances et de la sécurité. Les investissements dans les tests d'intégration et l'observabilité sont essentiels pour réduire les risques.

Les exemples les plus populaires de systèmes distribués aujourd'hui sont les microservices. Dans les architectures de microservices, les services principaux sont pilotés par le domaine (plutôt que par des problèmes techniques tels que l'interface utilisateur ou l'authentification) et détenus par des équipes autonomes. Les micro-frontends partagent les mêmes principes, étendant la portée de la solution au frontend.

L'informatique en nuage

Le cloud computing est un moyen d'acheter une infrastructure informatique sous forme de services avec un pay-as-you-go modèle au lieu de créer vos propres centres de données et d'acheter du matériel pour les faire fonctionner sur site. Le cloud computing présente plusieurs avantages :

- Votre entreprise gagne en agilité commerciale en étant en mesure d'expérimenter de nouvelles technologies sans avoir à prendre des engagements financiers importants et à long terme dès le départ.
- En faisant appel à un fournisseur de cloud tel que AWS, votre entreprise peut accéder à un large portefeuille de services nécessitant peu de maintenance et hautement intégrables (tels que des passerelles d'API, des bases de données, l'orchestration de conteneurs et des fonctionnalités cloud). L'accès à ces services permet à votre personnel de se concentrer sur le travail qui différencie votre organisation de la concurrence.
- Lorsque votre entreprise est prête à déployer une solution à l'échelle mondiale, vous pouvez la déployer sur l'infrastructure cloud du monde entier.

Le cloud computing prend en charge les micro-frontends en fournissant une infrastructure hautement gérée. Cela facilite end-to-end la prise en main par les équipes interfonctionnelles. Bien que l'équipe doive posséder de solides connaissances opérationnelles, les tâches manuelles liées au provisionnement de l'infrastructure, aux mises à jour du système d'exploitation et à la mise en réseau seraient une distraction.

Les micro-frontends évoluant dans des contextes limités, les équipes peuvent choisir le service le plus approprié pour les exécuter. Par exemple, les équipes peuvent choisir entre des fonctions cloud et des conteneurs pour le calcul, et elles peuvent choisir entre différentes versions de bases de données SQL et NoSQL ou de caches en mémoire. Les équipes peuvent même créer leurs micro-frontends sur une boîte à outils hautement intégrée AWS Amplify, telle que celle fournie avec des blocs de construction préconfigurés pour une infrastructure sans serveur.

Comparaison des micro-frontends avec des architectures alternatives

Comme pour toutes les stratégies architecturales, la décision d'adopter des micro-frontends doit être basée sur des critères d'évaluation guidés par les principes de votre organisation. Les micro-frontends présentent des avantages et des inconvénients. Si votre organisation décide d'utiliser des micro-frontends, vous devez mettre en place des stratégies pour relever les défis des systèmes distribués

Lors du choix d'une architecture d'application, les alternatives les plus populaires aux micro-frontends sont les monolithes, les applications n-tier et les microservices associés à une interface d'application mono-page (SPA). Toutes ces approches sont valables, et chacune d'elles présente des avantages et des inconvénients.

Monolithes

Une petite application qui ne nécessite pas de modifications fréquentes peut être livrée très rapidement sous forme de monolithe. Même dans les situations où une croissance significative est attendue, un monolithe est une première étape naturelle. Plus tard, le monolithe peut être retiré ou transformé en une structure plus flexible. En commençant par un monolithe, votre entreprise peut se lancer sur le marché, recueillir les commentaires des clients et améliorer le produit plus rapidement.

Cependant, les applications monolithiques ont tendance à se dégrader si elles ne sont pas soigneusement entretenues ou si la taille de la base de code augmente au fil du temps. Lorsque plusieurs équipes contribuent de manière significative à la même base de code, elles contribuent rarement toutes à sa maintenance et à son fonctionnement. Cela entraîne un déséquilibre des responsabilités, ce qui a un impact sur la rapidité et entraîne des inefficiences. Dans le même temps, le couplage involontaire entre les modules d'un monolithe entraîne des effets secondaires imprévus à mesure que la base de code évolue. Ces effets secondaires peuvent entraîner des dysfonctionnements et des pannes.

Applications de niveau N

Une application plus complexe dont le rythme d'évolution est relativement statique peut être construite sous la forme d'une architecture à trois niveaux (présentation, application, données), avec

une couche REST ou GraphQL entre le frontend et le backend. Cela est beaucoup plus flexible et les équipes des différents niveaux peuvent se développer indépendamment dans une certaine mesure. L'inconvénient d'une application n-tier est qu'il est beaucoup plus difficile de déployer des fonctionnalités. Le frontend et le backend étant découplés par un contrat d'API, les modifications majeures doivent être déployées ensemble ou l'API doit être versionnée.

Envisagez le scénario courant suivant : si le lancement d'une nouvelle fonctionnalité nécessite une modification du schéma de données, les responsables du produit peuvent mettre plusieurs jours à s'entendre sur un ensemble de fonctionnalités avec une équipe de frontend. Ensuite, l'équipe du frontend demandera à l'équipe du backend de développer et de publier les fonctionnalités de son côté. L'équipe principale travaillera avec les propriétaires des données pour publier une mise à jour du schéma de base de données. Ensuite, l'équipe du backend publiera une nouvelle version de l'API, afin que l'équipe du frontend puisse développer et publier ses modifications. Dans ce scénario, la propagation de toutes les modifications vers la production peut prendre des semaines, voire des mois, car chaque équipe dispose de son propre backlog, de ses propres priorités et de ses propres mécanismes en matière de développement, de test et de publication des modifications.

Microservices

Dans une architecture de microservices, le backend est décomposé en petits services, chacun répondant à un problème commercial particulier dans un contexte limité. Chaque microservice est également fortement découplé des autres services en exposant un contrat d'interface clairement défini.

Il convient de mentionner que les contextes limités et les contrats d'interface devraient également exister dans des monolithes bien conçus et des architectures n-tier. Dans une architecture de microservices, toutefois, la communication s'effectue via le réseau, généralement via le protocole HTTP, et les services disposent d'une infrastructure d'exécution dédiée. Cela permet le développement, la fourniture et le fonctionnement indépendants de chaque service principal.

Choix de l'approche adaptée à vos besoins

Les monolithes et les architectures n-tier regroupent plusieurs domaines en un seul artefact technique. Cela facilite la gestion d'aspects tels que les dépendances et le flux de données interne, mais complique la fourniture de nouvelles fonctionnalités. Pour maintenir une base de code cohérente, une équipe investit souvent du temps dans le refactoring et le découplage en raison de la grande base de code qu'elle doit gérer.

Les applications développées par quelques équipes peuvent ne pas avoir besoin de la complexité supplémentaire associée au passage aux micro-frontends. Cela est particulièrement vrai si les équipes ne paient pas les pénalités liées à un couplage élevé et aux longs délais de publication des modifications.

En résumé, les architectures plus complexes et distribuées constituent souvent le bon choix pour les applications complexes et en évolution rapide. Pour les applications de petite ou moyenne taille, une architecture distribuée n'est pas nécessairement supérieure à une architecture monolithique, en particulier si l'application n'évolue pas de façon spectaculaire sur une courte période.

Décisions architecturales dans les micro-frontends

Les équipes qui appliquent un modèle d'architecture micro-frontend à leurs applications doivent prendre plusieurs décisions concernant l'architecture dès le début :

- Identification des micro-frontends et définition des limites
- Composer des pages et des vues à l'aide de micro-frontends
- Routage, gestion des états et communication entre les micro-frontends
- Gérer les dépendances pour des préoccupations transversales

Les sections suivantes abordent ces sujets de manière plus approfondie.

Lorsque vous prenez des décisions en matière d'architecture, il est essentiel de disposer des indicateurs appropriés et de comprendre les modèles d'utilisation, les caractéristiques des applications et les compromis à faire. Par exemple, un site de commerce électronique présente des caractéristiques et des modes d'utilisation différents de ceux d'un outil de montage vidéo ou de tableaux de bord d'observabilité.

Les applications destinées au public présentant un trafic élevé et une faible profondeur de session peuvent être optimisées pour les indicateurs de chargement initial des pages, tels que Time to Interactive (TTI) et First Contentful Paint (FCP). En revanche, une application à laquelle les utilisateurs se connectent au début de leur journée et avec laquelle ils continuent d'interagir tout au long de la journée peut être optimisée pour l'expérience intégrée à l'application. L'équipe chargée de l'application peut optimiser la métrique FID (First Input Delay) après chaque navigation au lieu du chargement initial de la page.

Les sites Web publics doivent s'adapter à différents environnements de navigation. Les applications d'entreprise présentant des contraintes connues sur l'environnement client peuvent optimiser la composition de leur micro-frontend en fonction de leurs contraintes.

Il n'existe pas de bon choix unique pour les décisions en matière d'architecture. Comprenez les compromis, le contexte dans lequel l'entreprise opère, les modèles d'utilisation et les indicateurs pour orienter les décisions adaptées à chaque application individuelle.

Identification des limites du micro-frontend

Pour améliorer l'autonomie des équipes, les fonctionnalités métier fournies par une application peuvent être décomposées en plusieurs micro-frontends avec un minimum de dépendances les unes par rapport aux autres.

En suivant la méthodologie DDD évoquée précédemment, les équipes peuvent décomposer un domaine d'application en sous-domaines métiers et en contextes délimités. Les équipes autonomes peuvent alors s'approprier les fonctionnalités de leurs contextes délimités et fournir ces contextes sous forme de micro-frontends. Pour plus d'informations sur la séparation des préoccupations, consultez le diagramme Serverless Land.

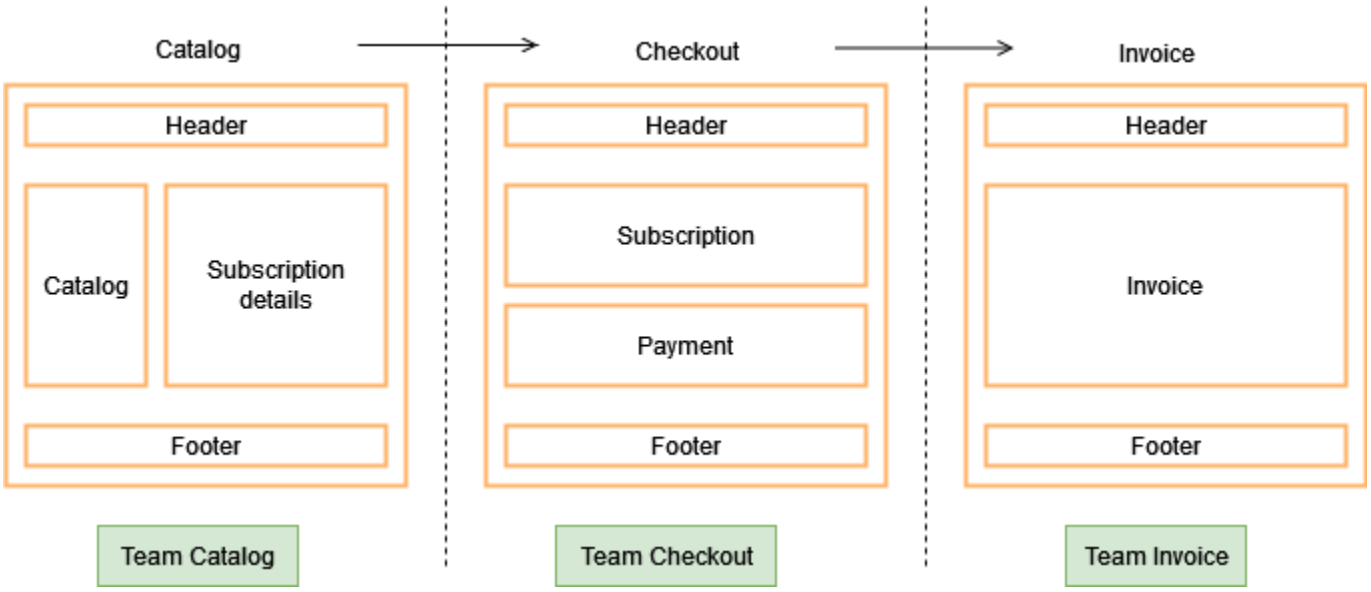
Un contexte délimité bien défini devrait minimiser les chevauchements fonctionnels et la nécessité d'une communication d'exécution entre les contextes. La communication requise peut être mise en œuvre à l'aide de méthodes axées sur les événements. Cela n'est pas différent de l'architecture axée sur les événements pour le développement de microservices.

Une application bien conçue doit également prendre en charge la livraison des futures extensions par les nouvelles équipes afin d'offrir une expérience cohérente aux clients.

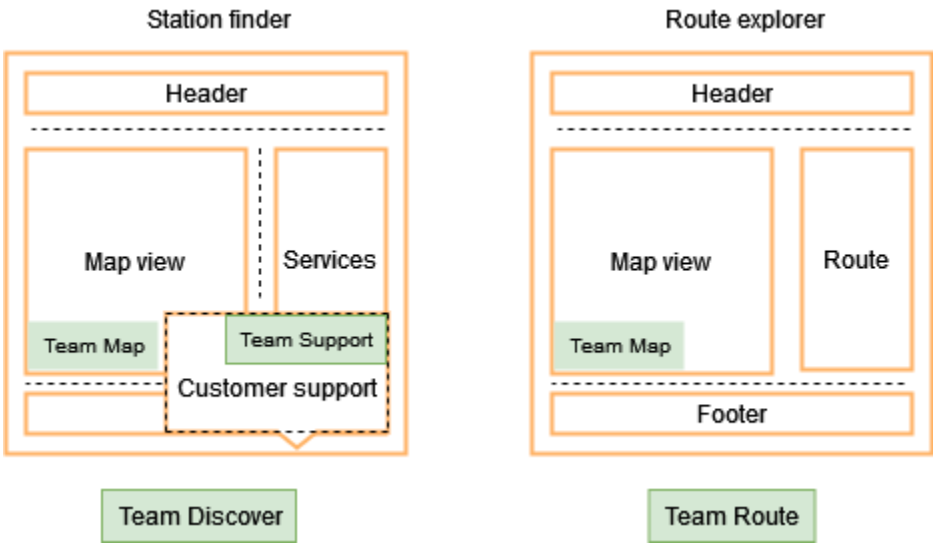
Comment découper une application monolithique en micro-frontends

La section Vue d'ensemble comprenait un exemple d'identification de contextes fonctionnels indépendants sur une page Web. Plusieurs modèles de division des fonctionnalités de l'interface utilisateur apparaissent.

Par exemple, lorsque les domaines commerciaux constituent les étapes d'un parcours utilisateur, une division verticale peut être appliquée sur le frontend, où un ensemble de vues du parcours utilisateur est diffusé sous forme de micro-interfaces. Le schéma suivant montre une division verticale, dans laquelle les étapes du catalogue, du paiement et de la facturation sont exécutées par des équipes distinctes sous la forme de micro-interfaces distinctes.



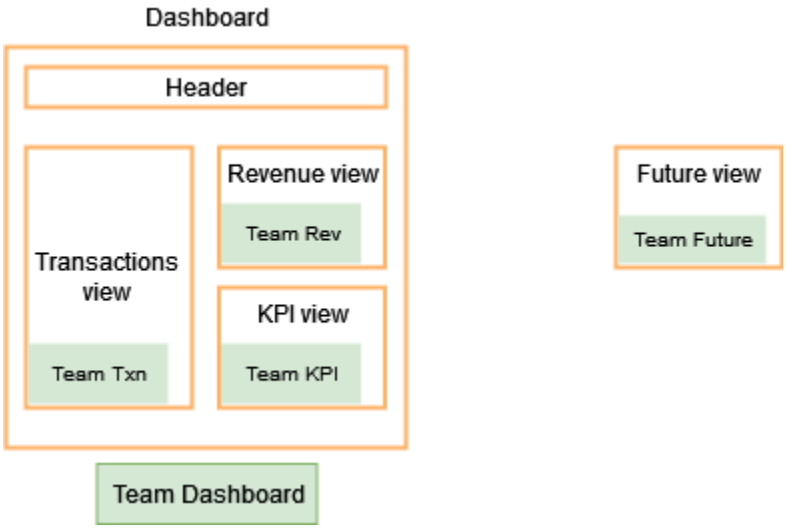
Pour certaines applications, la division verticale seule peut ne pas être suffisante. Par exemple, il peut être nécessaire de fournir certaines fonctionnalités dans de nombreuses vues. Pour ces applications, vous pouvez appliquer un fractionnement mixte. Le schéma suivant montre une solution mixte dans laquelle les micro-interfaces pour Station Finder et Route Explorer utilisent toutes deux la fonctionnalité d'affichage cartographique.



Les applications de type portail ou tableau de bord réunissent généralement les fonctionnalités du frontend dans une seule vue. Dans ces types d'applications, chaque widget peut être fourni sous forme de micro-frontend, et l'application d'hébergement définit les contraintes et les interfaces que les micro-frontends doivent implémenter.

Cette approche fournit un mécanisme permettant aux micro-frontends de gérer des problèmes tels que le dimensionnement de la fenêtre d'affichage, les fournisseurs d'authentification, les paramètres de configuration et les métadonnées. Ces types d'applications optimisent l'extensibilité. De nouvelles fonctionnalités peuvent être développées par de nouvelles équipes afin d'étendre les capacités du tableau de bord.

Le schéma suivant montre une application de tableau de bord développée par trois équipes individuelles faisant partie de Team Dashboard.



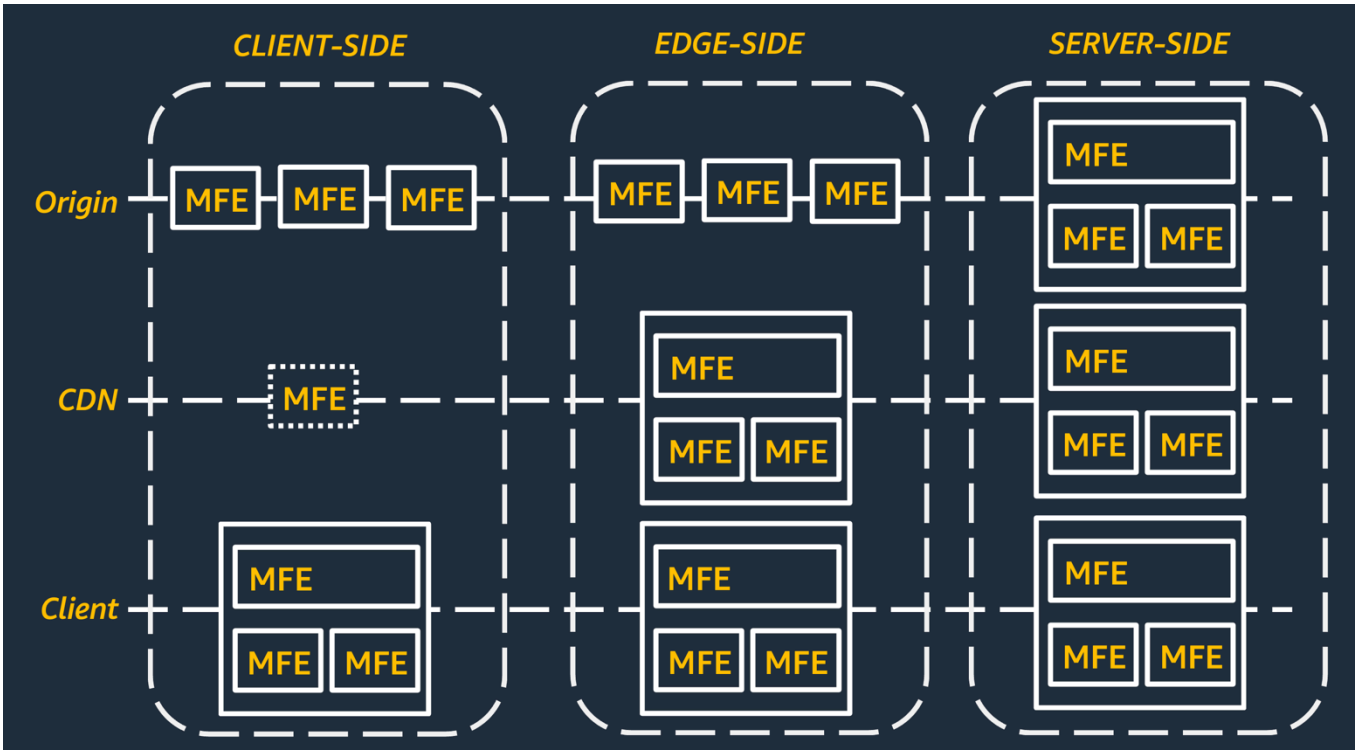
Dans le diagramme, la vue future représente les nouvelles fonctionnalités développées par les nouvelles équipes pour étendre le tableau de bord d'équipe et les fonctionnalités du tableau de bord.

Les applications de portail et de tableau de bord composent généralement les fonctionnalités en utilisant une répartition mixte dans l'interface utilisateur. Les micro-frontends sont configurables avec des paramètres bien définis, notamment des contraintes de position et de taille.

Composer des pages et des vues à l'aide de micro-frontends

Vous pouvez composer des vues d'une application avec une composition côté client, une composition côté bord et une composition côté serveur. Les modèles de composition présentent des caractéristiques différentes en termes de compétences d'équipe nécessaires, de tolérance aux pannes, de performances et de comportement du cache.

Le schéma suivant montre comment la composition se produit au niveau des couches côté client, côté périphérie et côté serveur d'une architecture de micro-frontend.



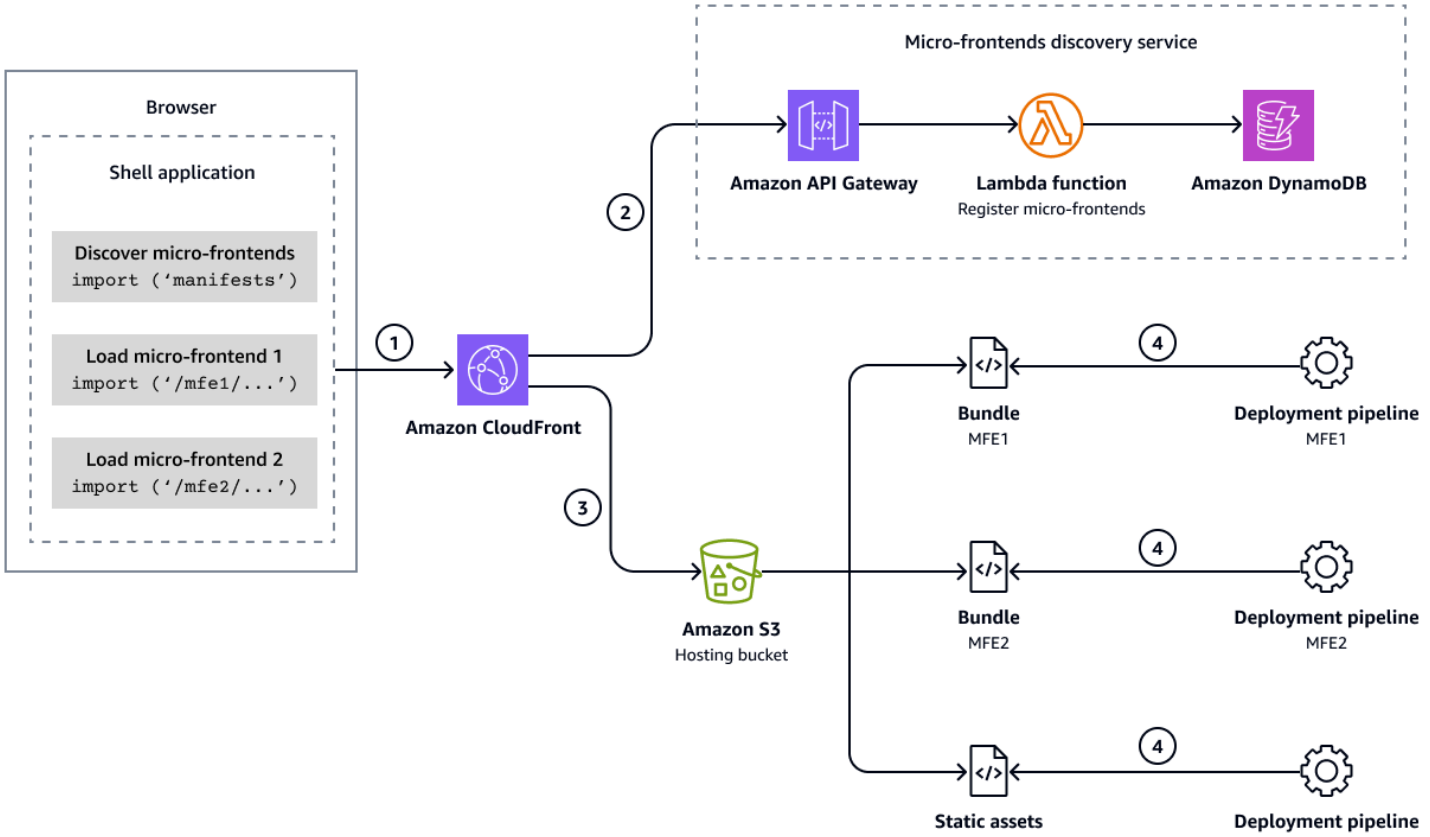
Les couches côté client, côté périphérie et côté serveur sont abordées dans les sections suivantes.

Montage côté client

Chargez et ajoutez dynamiquement des micro-frontends sous forme de fragments DOM (Document Object Model) sur le client (navigateur ou vue Web mobile). Les artefacts du micro-frontend, tels que JavaScript les fichiers CSS, peuvent être chargés à partir de réseaux de diffusion de contenu (CDN) pour réduire la latence. La composition côté client nécessite les éléments suivants :

- Une équipe chargée de posséder et de gérer une application shell ou un framework de micro-frontend pour permettre la découverte, le chargement et le rendu des composants du micro-frontend lors de l'exécution dans le navigateur
- Des niveaux de compétence élevés dans les technologies frontales telles que le HTML, le CSS et JavaScript une compréhension approfondie des environnements de navigateur
- Optimisation de la quantité de données JavaScript chargées dans une page et discipline pour éviter les conflits d'espaces de noms globaux

Le schéma suivant montre un exemple d' AWS architecture pour la composition côté client sans serveur.



La composition côté client s'effectue dans l'environnement du navigateur via une application shell. Le schéma montre les détails suivants :

1. Une fois l'application shell chargée, elle envoie une demande initiale à Amazon pour découvrir les micro-frontends CloudFront à charger via un point de terminaison du manifeste.
2. Les manifestes contiennent des informations sur chaque micro-frontend (par exemple, le nom, l'URL, la version et le comportement de secours). Les manifestes sont servis par le service de découverte des microfrontends. Dans le schéma, ce service de découverte est représenté par Amazon API Gateway, une AWS Lambda fonction, et Amazon DynamoDB. L'application shell utilise les informations du manifeste pour demander à des micro-frontends individuelles de composer la page selon une mise en page donnée.
3. Chaque bundle de micro-frontend est composé de fichiers statiques (tels que JavaScript CSS et HTML). Les fichiers sont hébergés dans un compartiment Amazon Simple Storage Service (Amazon S3) où ils sont servis. CloudFront
4. Les équipes peuvent déployer de nouvelles versions de leurs micro-frontends et mettre à jour les informations du manifeste en utilisant des pipelines de déploiement dont elles sont propriétaires.

Composition côté bord

Utilisez des techniques de transclusion telles que les inclusions côté Edge (ESI) ou les inclusions côté serveur (SSI) prises en charge par certains CDN et proxys devant les serveurs d'origine pour composer une page avant de l'envoyer par câble aux clients. L'ESI requiert les éléments suivants :

- Un CDN avec fonctionnalité ESI ou un déploiement de proxy devant des micro-frontends côté serveur. Les implémentations de proxy telles que HAProxy, Varnish et NGINX prennent en charge le SSI.
- Compréhension de l'utilisation et des limites des implémentations ESI et SSI.

Les équipes qui lancent de nouvelles candidatures ne choisissent généralement pas la composition côté bord comme modèle de composition. Cependant, ce modèle peut fournir une voie pour les applications existantes qui reposent sur la transclusion.

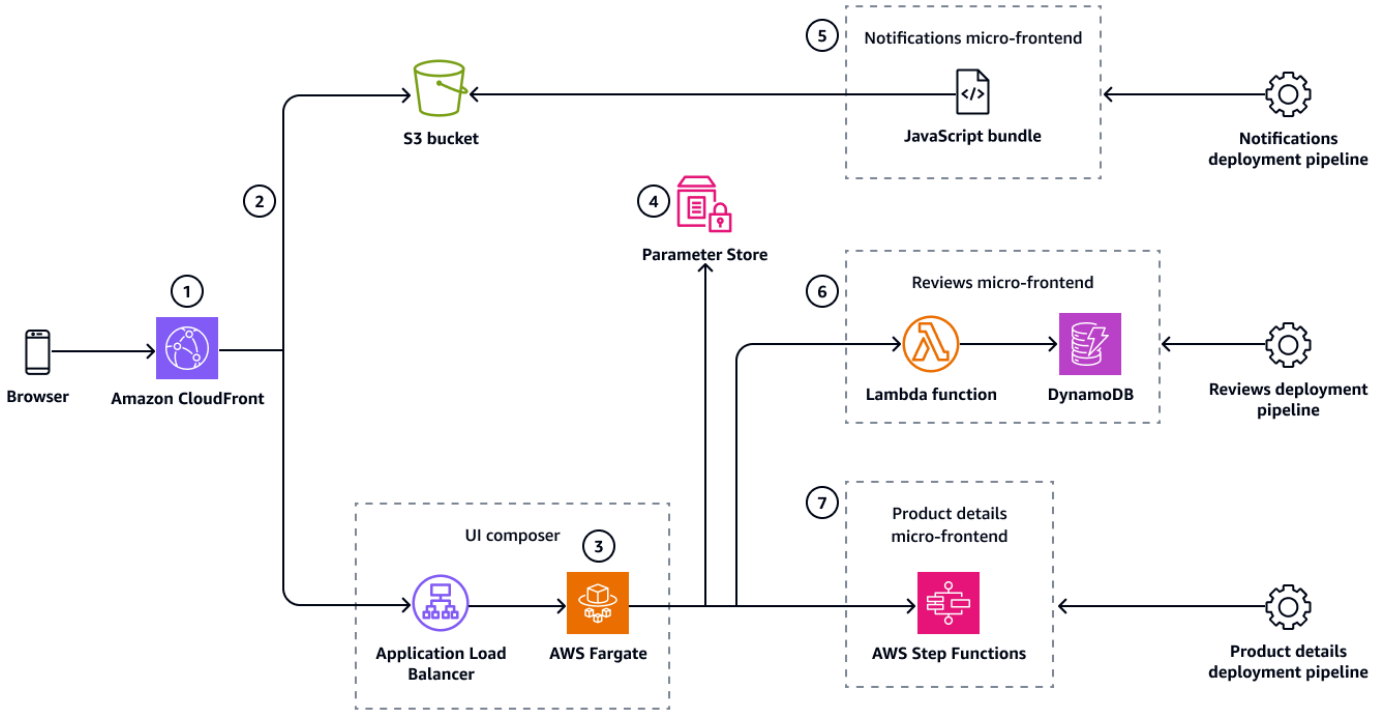
Montage côté serveur

Utilisez les serveurs d'origine pour composer les pages avant qu'elles ne soient mises en cache sur le périphérique. Cela peut être fait à l'aide de technologies traditionnelles, telles que PHP, Jakarta Server Pages (JSP) ou de bibliothèques de modèles, pour composer les pages en incluant des fragments provenant de micro-frontends. Vous pouvez également utiliser JavaScript des frameworks, tels que Next.js, exécutés sur le serveur pour composer des pages sur le serveur avec un rendu côté serveur (SSR).

Une fois les pages affichées sur le serveur, elles peuvent être mises en cache sur des CDN afin de réduire le temps de latence. Lorsque de nouvelles versions de micro-frontends sont déployées, les pages doivent être restituées et le cache doit être mis à jour pour fournir les dernières versions aux clients.

La composition côté serveur nécessite une connaissance approfondie de l'environnement du serveur afin d'établir des modèles de déploiement, de découverte des micro-frontends côté serveur et de gestion du cache.

Le schéma suivant montre la composition côté serveur.



Le diagramme inclut les composants et processus suivants :

1. Amazon CloudFront fournit un point d'entrée unique pour accéder à l'application. La distribution a deux origines : la première pour les fichiers statiques et la seconde pour le compositeur d'interface utilisateur.
2. Les fichiers statiques sont hébergés dans un compartiment Amazon S3. Ils sont utilisés par le navigateur et le compositeur de l'interface utilisateur pour les modèles HTML.
3. Le compositeur d'interface utilisateur s'exécute sur un cluster de conteneurs dans AWS Fargate. Avec une solution conteneurisée, vous pouvez utiliser les fonctionnalités de streaming et de rendu multithread si nécessaire.
4. Parameter Store, une fonctionnalité de AWS Systems Manager, est utilisé comme système de découverte de microfrontends de base. Cette fonctionnalité fournit un magasin clé-valeur utilisé par le compositeur de l'interface utilisateur pour récupérer les points de terminaison du micro-frontend à consommer.
5. Le micro-frontend de notifications stocke le JavaScript bundle optimisé dans le compartiment S3. Cela se répercute sur le client car il doit réagir aux interactions de l'utilisateur.
6. Le micro-frontend des avis est composé d'une fonction Lambda, et les avis des utilisateurs sont stockés dans DynamoDB. Le micro-frontend des révisions est entièrement rendu côté serveur et produit un fragment HTML.

7. Détails du produit Le micro-frontend est un micro-frontend low-code qui utilise. AWS Step Functions L'Express Workflow peut être invoqué de manière synchrone et contient la logique de rendu du fragment HTML et d'une couche de mise en cache.

Pour plus d'informations sur la composition côté serveur, consultez le billet de blog intitulé Micro-frontends de rendu côté serveur : l'architecture.

Routage et communication via des micro-frontends

Les options de routage dépendent de l'approche de composition. La communication peut être optimisée en réduisant le couplage entre les composants du frontend.

Routage

Les applications qui utilisent la composition côté client avec division verticale peuvent utiliser le routage côté serveur (application multipage) ou le routage côté client (application d'une seule page). S'ils utilisent une répartition mixte pour la composition de l'interface utilisateur, le routage côté client est nécessaire pour prendre en charge des hiérarchies de routage plus approfondies des micro-frontends sur une page.

Les applications qui utilisent la composition côté périphérie et la composition côté serveur s'alignent mieux sur le routage côté serveur ou sur le routage avec le calcul périphérique tel que Lambda @Edge avec Amazon. CloudFront

Communication entre micro-frontends

Avec les architectures de micro-frontend, nous recommandons de réduire le couplage entre les composants frontaux. Une approche pour réduire le couplage consiste à abandonner les appels de fonction synchrones au profit de la messagerie asynchrone.

Les temps d'exécution du navigateur et les interactions avec les utilisateurs sont asynchrones par nature. Les événements peuvent être échangés entre producteurs et consommateurs par le biais de messages. Les événements fournissent une interface bien définie pour la communication entre les micro-frontends.

Si vous suivez les pratiques DDD pour identifier vos contextes délimités pour les micro-frontends, l'étape suivante consiste à identifier les événements qui doivent être communiqués au-delà des limites.

Le mécanisme de messagerie pour les événements peut être des événements DOM natifs (`CustomEvents`), des émetteurs d' JavaScript événements ou des bibliothèques de flux réactifs fournies par les équipes de la plateforme. Les micro-frontends publient des événements et s'abonnent à des événements pertinents pour leur contexte limité. Avec cette méthode, les éditeurs et les abonnés n'ont pas besoin de se connaître. Le contrat est la définition de l'événement. Pour une représentation visuelle de cela, consultez la section Communiquer avec les événements du diagramme Contexte limité avec architectures d'événements.

Gérer les dépendances pour des préoccupations transversales

La gestion consciente des dépendances est essentielle au succès d'une architecture distribuée telle que les micro-frontends. La gestion des dépendances est l'un des aspects les plus difficiles du développement du micro-frontend.

Dans une architecture micro-frontend, deux aspects importants de la gestion des dépendances sont la baisse de performance liée au transfert d'artefacts de code volumineux vers le client et la surcharge des ressources de calcul. Idéalement, votre organisation doit définir la manière dont les dépendances sont maintenues dans une architecture frontale distribuée.

Trois stratégies viables pour imposer la maintenance des dépendances sont Share Nothing, en utilisant des standards Web tels que l'importation de cartes et la fédération de modules. D'autres approches sont des anti-modèles car elles violent les principes de base des architectures distribuées.

Ne partagez rien, dans la mesure du possible

L'approche « ne rien partager » postule qu'aucune dépendance entre des artefacts logiciels indépendants ne doit être partagée, ou du moins pas lors de l'intégration ou de l'exécution. Cela signifie que si deux micro-frontends dépendent de la même bibliothèque, chacune doit être intégrée à la bibliothèque au moment de la construction et expédiée séparément. De plus, chaque micro-frontend doit valider que la bibliothèque ne pollue pas les espaces de noms globaux et les ressources partagées.

Cela entraîne des licenciements, mais il s'agit d'un compromis délibéré avec une agilité maximale. En l'absence de dépendances d'exécution partagées, les équipes disposent d'une flexibilité maximale pour faire évoluer le logiciel de la manière qu'elles jugent utile, à condition de le faire dans le cadre de leur solution et de ne rompre aucun contrat d'interface.

Sur une plate-forme où les micro-frontends suivent le principe de ne rien partager, il est important de garder les micro-frontends aussi légers que possible. Cela nécessite des développeurs compétents

et assidus dans l'optimisation des performances de leurs micro-frontends et qui ne sacrifient pas l'expérience utilisateur au profit de l'expérience des développeurs.

Lorsque vous partagez du code

Lorsque vous décidez de partager du code, vous pouvez le partager sous forme de bibliothèques ou de modules d'exécution. Par exemple, l'équipe principale du frontend fournit des bibliothèques destinées à la consommation du micro-frontend via des CDN. Les équipes chargées de la valeur commerciale peuvent charger les bibliothèques au moment de l'exécution ou utiliser des référentiels de packages pour publier leurs bibliothèques. Les équipes de micro-frontend peuvent développer sur une version spécifique de la bibliothèque packagée au moment de la création, de la même manière que les applications mobiles utilisant des frameworks hybrides.

Une troisième option consiste à utiliser un registre de paquets privé pour prendre en charge l'intégration des bibliothèques communes au moment de la création. Cela réduit le risque qu'une modification définitive du contrat de bibliothèque entraîne des erreurs lors de l'exécution. Cependant, cette approche plus conservatrice nécessite la mise en place d'une plus grande gouvernance pour synchroniser tous les micro-frontends avec les nouvelles versions des bibliothèques.

Pour améliorer les temps de chargement des pages, les micro-frontends peuvent externaliser les dépendances de bibliothèque à charger à partir de fragments mis en cache à partir d'un CDN tel qu'Amazon. CloudFront

Pour gérer les dépendances d'exécution, les micro-frontends peuvent utiliser des cartes d'importation (ou des bibliothèques telles que`System.js`) pour spécifier l'endroit d'où chaque module est chargé au moment de l'exécution. La fédération de modules webpack est une autre approche qui permet de pointer vers une version hébergée d'un module distant et de résoudre les dépendances communes entre les micro-frontends indépendants.

Une autre approche consiste à faciliter le chargement dynamique des cartes d'importation avec une demande initiale adressée à un point de terminaison de découverte.

État partagé

Pour réduire le couplage des micro-frontends, il est important d'éviter une gestion globale des états accessible depuis tous les micro-frontends dans une même vue, comme dans le cas des architectures monolithiques. Par exemple, le fait de disposer d'un magasin Redux mondial accessible depuis tous les micro-frontends augmente le couplage.

Un modèle pour éliminer l'état partagé consiste à l'encapsuler dans des micro-frontends et à communiquer avec des messages asynchrones, comme indiqué précédemment.

Lorsque cela est absolument nécessaire, introduisez des interfaces bien définies pour l'état global et optez pour le partage en lecture seule afin d'éviter tout comportement inattendu :

- En cas de division verticale, vous pouvez utiliser les composants URL et le stockage du navigateur pour accéder aux informations de l'environnement hôte.
- Lorsque vous avez une répartition mixte, vous pouvez également utiliser les événements ou JavaScript bibliothèques personnalisés standard du DOM, tels que des émetteurs d'événements ou des flux bidirectionnels, pour transmettre des informations aux micro-frontends.

Si vous devez partager plusieurs informations entre des micro-frontends, nous vous recommandons de revoir les limites des micro-frontends. Le besoin de partager peut être dû à l'évolution de l'entreprise ou à une conception initiale médiocre.

Il est également possible d'utiliser des sessions côté serveur, où chaque micro-frontend récupère les données requises à l'aide d'un identifiant de session. Pour réduire le couplage, il est important d'éliminer l'état partagé et de séparer les données de session spécifiques au micro-frontend.

Cadres et outils

Les frameworks frontaux, tels que Angular et Next.js, ne manquent pas, mais la plupart d'entre eux ne sont pas créés en pensant aux micro-frontends. Par conséquent, il leur manque parfois des mécanismes pour relever les défis de l'architecture micro-frontend.

Considérations générales relatives au cadre

Ce guide n'a pas pour but de recommander ou de comparer des cadres individuels. Comme plusieurs micro-frontends s'exécutent souvent sur la même page d'application Web, le chargement et les performances d'exécution sont des préoccupations majeures. Il est important de choisir un framework qui introduit le moins de frais possible.

Les frameworks sont divisés en fonction de la couche de rendu :

- Rendu côté client (CSR)
- Rendu côté serveur (SSR)

Les architectures frontales incluent d'autres fonctionnalités, telles que la génération de sites statiques (SSG). Cependant, le SSG n'est effectué qu'une seule fois. Les micro-frontends étant principalement composés au moment de l'exécution, CSR et SSR sont les principales options.

Rendu côté client

Pour la RSE, il existe deux options populaires :

- Cadre SPA unique
- Fédération de modules

Le Single SPA est un choix léger pour composer des micro-frontends. Il résout les problèmes les plus courants des architectures de micro-frontend, tels que la composition de plusieurs micro-frontends sur la même page et la prévention des conflits de dépendances.

Module Federation a commencé comme un plugin, proposé par Webpack 5, et il résout la grande majorité des défis des architectures de micro-frontend, y compris la gestion des dépendances entre différents artefacts. Module Federation 2.0 fonctionne nativement avec Rspack, webpack, esbuild, et maintenant avec. JavaScript

Envisagez de ne pas utiliser de framework du tout. Les navigateurs modernes, dont la part de marché globale est de 98 % selon caniuse.com, offrent des fonctionnalités telles que des éléments personnalisés de manière native, et ils conviennent parfaitement à une application de micro-frontends. Si nécessaire, combinez des éléments personnalisés avec des bibliothèques légères pour la propagation d'événements, l'internationalisation ou d'autres préoccupations spécifiques.

Rendu côté serveur

Du côté de la SSR, les deux principales options sont plus compliquées :

- Adoptez un framework existant tel que Next.js et appliquez le principe des micro-frontends qui utilise la fédération de modules.
- Utilisez HTML- over-the-wire pour échanger des fragments HTML qui représentent des micro-frontends, et composez ces fragments dans un modèle lors de l'exécution. Podium est un exemple de cette approche.

Intégration d'API – Backend pour frontend

Le modèle Backends for Frontends (BFF) est généralement utilisé dans les environnements de microservices. Dans le contexte des micro-frontends, un BFF est un service côté serveur qui appartient à un micro-frontend. Tous les micro-frontends n'ont pas besoin d'un BFF. Toutefois, si vous utilisez un BFF, il doit s'exécuter dans le même contexte délimité et ne pas être partagé entre d'autres contextes délimités.

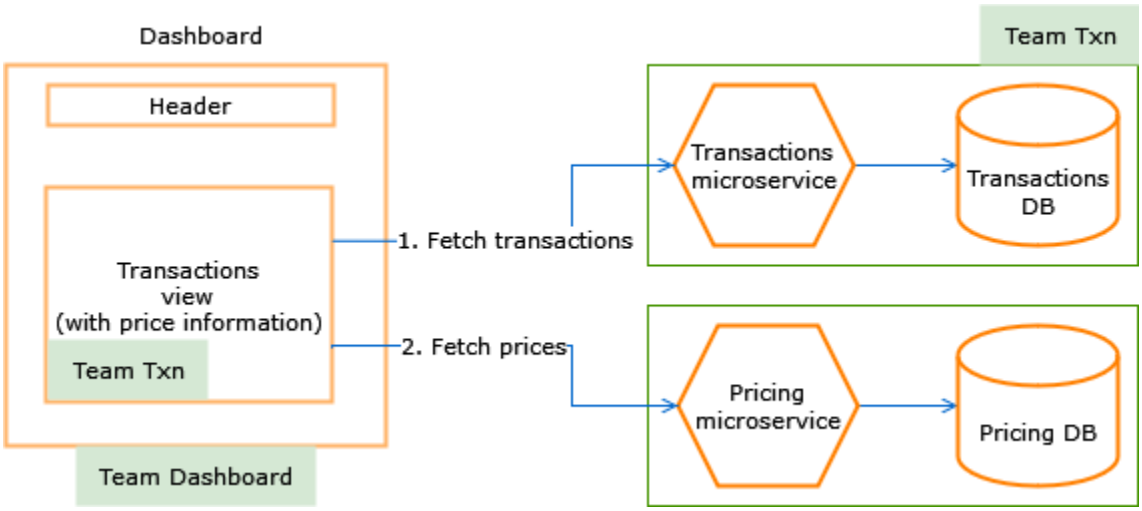
Contrairement à un service traditionnel, un BFF ne suit pas de modèle de domaine. Il s'agit plutôt d'une couche d'API permettant au micro-frontend de prétraiter les données avant qu'elles n'atteignent le client. Cela peut s'avérer utile dans les domaines suivants :

- Autorisation vers des API privées
- Agrégation de données provenant de différentes sources
- Transformation des données pour réduire la charge du réseau et faciliter la consommation de données par le client

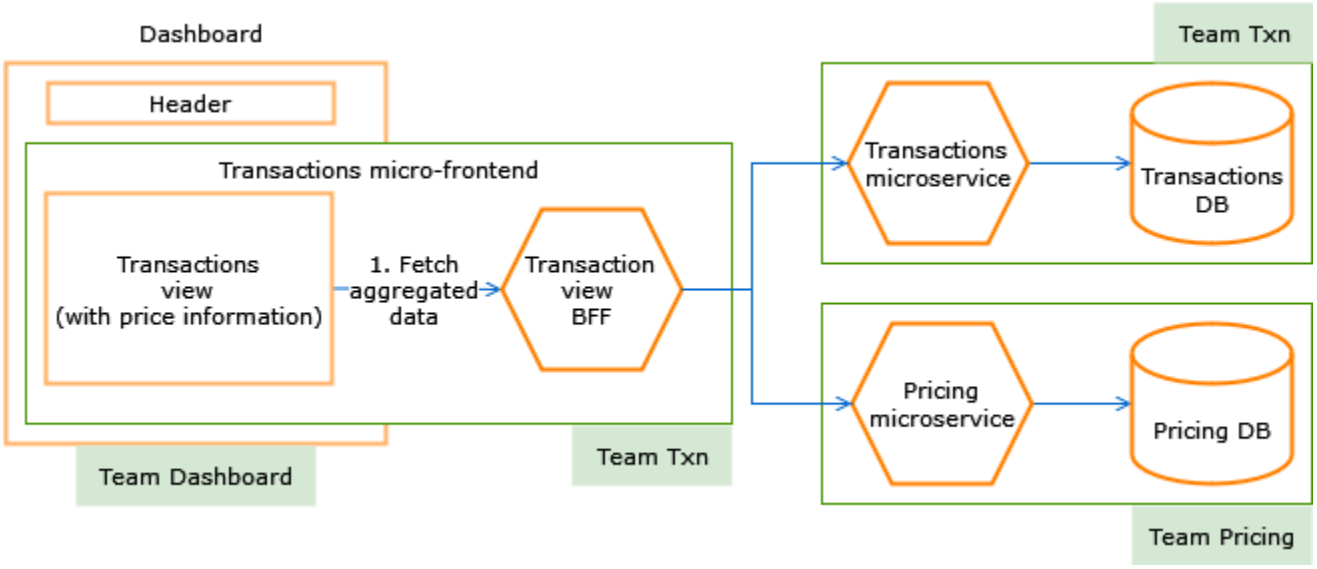
En tant que tel, un BFF appartient au micro-frontend, et non au niveau de service du domaine. Les BFF peuvent être déployés en utilisant les méthodes suivantes :

- API AWS AppSync GraphQL
- Un ensemble de fonctions AWS Lambda
- En tant que conteneur exécuté sur Amazon ECS, Amazon EKS ou AWS AppRunner

Le schéma suivant montre que sans le modèle BFF, les micro-frontends doivent se connecter à des points de terminaison d'API de microservice individuels pour récupérer et agréger des données.



Au lieu de cela, avec le modèle BFF du schéma suivant, les micro-frontends peuvent communiquer avec leur propre backend et récupérer des données agrégées.



Les équipes peuvent développer des meilleurs amis pour différents canaux tels que le mobile, le Web ou des vues spécifiques, en veillant à optimiser les interactions entre les backend en réduisant le nombre de bavardages.

Style et CSS

Les feuilles de style en cascade (CSS) sont un langage qui permet de déterminer de manière centralisée la présentation d'un document au lieu de coder en dur le formatage du texte et des objets. La fonctionnalité en cascade du langage est conçue pour contrôler les priorités entre les styles en utilisant l'héritage. Lorsque vous travaillez sur des micro-frontends et que vous créez une stratégie pour gérer les dépendances, la fonctionnalité en cascade du langage peut s'avérer un défi.

Par exemple, deux micro-frontends coexistent sur la même page, chacune définissant son propre style pour l'`body`élément HTML. Si chacun récupère son propre fichier CSS et l'attache au DOM à l'aide d'une `style` balise, le fichier CSS remplace le premier s'ils ont tous deux une définition d'éléments HTML communs, de noms de classe ou d'ID d'élément. Il existe différentes stratégies pour résoudre ces problèmes, en fonction de la stratégie de dépendance que vous choisissez pour gérer les styles.

Actuellement, l'approche la plus populaire pour équilibrer les performances, la cohérence et l'expérience des développeurs consiste à développer et à maintenir un système de conception.

Systèmes de conception – Une approche axée sur le partage

Cette approche utilise un système pour partager le style le cas échéant, tout en prenant en charge les divergences occasionnelles afin d'équilibrer la cohérence, les performances et l'expérience des développeurs. Un système de conception est un ensemble de composants réutilisables, guidé par des normes claires. Le développement du système de conception est généralement piloté par une seule équipe avec l'apport et les contributions de nombreuses équipes. Concrètement, un système de conception est un moyen de partager des éléments de bas niveau qui peuvent être exportés sous forme de JavaScript bibliothèque. Les développeurs de micro-frontend peuvent utiliser la bibliothèque comme dépendance pour créer des interfaces simples en composant des ressources disponibles prédéfinies et comme point de départ pour créer de nouvelles interfaces.

Prenons l'exemple d'un micro-frontend qui a besoin d'un formulaire. L'expérience typique des développeurs consiste à utiliser des composants prédéfinis disponibles dans le système de conception pour composer des zones de texte, des boutons, des listes déroulantes et d'autres éléments de l'interface utilisateur. Le développeur n'a pas besoin d'écrire de style pour les composants eux-mêmes, mais uniquement pour leur apparence. Le système à créer et à publier peut utiliser Webpack Module Federation ou une approche similaire pour déclarer le système de

conception en tant que dépendance externe, de sorte que la logique du formulaire soit empaquetée sans inclure le système de conception.

Plusieurs micro-frontends peuvent alors faire de même pour répondre à des préoccupations communes. Lorsque les équipes développent de nouveaux composants qui peuvent être partagés entre plusieurs micro-frontends, ces composants sont ajoutés au système de conception une fois arrivés à maturité.

L'un des principaux avantages de l'approche du système de conception est le haut niveau de cohérence. Bien que les micro-frontends puissent écrire des styles et parfois remplacer ceux du système de conception, cela n'est pas nécessaire. Les principaux éléments de bas niveau ne changent pas souvent et offrent des fonctionnalités de base extensibles par défaut. Un autre avantage est la performance. Avec une bonne stratégie de création et de publication, vous pouvez produire des ensembles partagés minimaux instrumentés par le shell de l'application. Vous pouvez encore vous améliorer lorsque plusieurs ensembles spécifiques au micro-frontend sont chargés de manière asynchrone à la demande, avec un encombrement minimal en termes de bande passante réseau. Enfin, l'expérience du développeur est idéale, car les utilisateurs peuvent se concentrer sur la création d'interfaces riches sans avoir à réinventer la roue (comme l'écriture JavaScript et le CSS chaque fois qu'un bouton doit être ajouté à une page).

L'inconvénient est qu'un système de conception, quel qu'il soit, est une dépendance, il doit donc être maintenu et parfois mis à jour. Si plusieurs micro-frontends nécessitent une nouvelle version d'une dépendance partagée, vous pouvez utiliser l'une des méthodes suivantes :

- Un mécanisme d'orchestration qui peut parfois récupérer plusieurs versions de cette dépendance partagée sans conflit
- Une stratégie partagée pour déplacer toutes les personnes dépendantes vers une nouvelle version

Par exemple, si tous les micro-frontends dépendent de la version 3.0 d'un système de conception et qu'il existe une nouvelle version appelée 3.1 à utiliser de manière partagée, vous pouvez implémenter des indicateurs de fonctionnalité pour que tous les microfrontends migrent avec un minimum de risques. Pour plus d'informations, consultez la section Feature flags. Un autre inconvénient potentiel est que les systèmes de design ne se limitent généralement pas au style. Ils incluent également des JavaScript pratiques et des outils. Ces aspects nécessitent de parvenir à un consensus par le biais du débat et de la collaboration.

La mise en œuvre d'un système de conception est un bon investissement à long terme. C'est une approche populaire, et elle devrait être envisagée par tous ceux qui travaillent sur une architecture

frontend complexe. Cela nécessite généralement que les ingénieurs frontaux et les équipes de produits et de conception collaborent et définissent des mécanismes pour interagir les uns avec les autres. Il est important de prévoir le temps nécessaire pour atteindre l'état souhaité. Il est également important d'avoir le soutien de la direction afin que les gens puissent construire quelque chose de fiable, bien entretenu et performant à long terme.

CSS entièrement encapsulé – Une approche qui ne partage rien

Chaque micro-frontend utilise des conventions et des outils pour surmonter la fonctionnalité en cascade du CSS. Par exemple, assurez-vous que le style de chaque élément est toujours associé à un nom de classe plutôt qu'à l'ID de l'élément, et que les noms de classe sont toujours uniques. De cette façon, tout est limité aux micro-frontends individuels et le risque de conflits indésirables est minimisé. Le shell de l'application est généralement chargé de charger les styles des micro-frontends après leur chargement dans le DOM, bien que certains outils regroupent les styles en utilisant. JavaScript

Le principal avantage de ne rien partager est la réduction du risque d'introduire des conflits entre les micro-frontends. Un autre avantage est l'expérience du développeur. Chaque micro-frontend ne partage rien avec les autres micro-frontends. La publication et le test de manière isolée sont plus simples et plus rapides.

L'un des principaux inconvénients de l'approche « ne rien partager » est le manque potentiel de cohérence. Aucun système n'est en place pour évaluer la cohérence. Même si l'objectif est de dupliquer ce qui est partagé, cela devient difficile lorsqu'il s'agit de trouver un équilibre entre rapidité de publication et collaboration. Une mesure d'atténuation courante consiste à créer des outils pour mesurer la cohérence. Par exemple, vous pouvez créer un système permettant de prendre des captures d'écran automatisées de plusieurs micro-frontends affichées sur une page avec un navigateur sans en-tête. Vous pouvez ensuite consulter manuellement les captures d'écran avant de les publier. Toutefois, cela exige discipline et gouvernance. Pour plus d'informations, consultez la section Équilibrer l'autonomie avec l'alignement.

Selon le cas d'utilisation, les performances constituent un autre inconvénient potentiel. Si toutes les micro-interfaces utilisent une grande quantité de style, le client doit télécharger une grande partie du code dupliqué. Cela aura un impact négatif sur l'expérience utilisateur.

Cette approche consistant à ne rien partager ne doit être envisagée que pour les architectures de micro-frontend qui n'impliquent que quelques équipes, ou pour les micro-frontends qui peuvent tolérer une faible cohérence. Il peut également s'agir d'une première étape naturelle lorsqu'une organisation travaille sur un système de conception.

CSS global partagé – Une approche de partage de tous

Avec cette approche, tout le code lié au style est stocké dans un référentiel central où les contributeurs écrivent du CSS pour tous les micro-frontends en travaillant sur des fichiers CSS ou en utilisant des préprocesseurs tels que Sass. Lorsque des modifications sont apportées, un système de compilation crée un bundle CSS unique qui peut être hébergé dans un CDN et inclus dans chaque micro-frontend par le shell de l'application. Les développeurs de micro-frontend peuvent concevoir et créer leurs applications en exécutant leur code via un shell d'application hébergé localement.

Outre l'avantage évident de réduire le risque de conflits entre les micro-frontends, les avantages de cette approche sont la cohérence et les performances. Cependant, en découplant les styles du balisage et de la logique, il est plus difficile pour les développeurs de comprendre comment les styles sont utilisés, comment ils peuvent évoluer et comment ils peuvent être déconseillés. Par exemple, il peut être plus rapide d'introduire un nouveau nom de classe que de se renseigner sur la classe existante et les conséquences de la modification de ses propriétés. Les inconvénients de la création d'un nouveau nom de classe sont l'augmentation de la taille des ensembles, qui affecte les performances, et l'introduction potentielle d'incohérences dans l'expérience utilisateur.

Bien qu'un CSS global partagé puisse être le point de départ d'une monolith-to-micro-frontends migration, il est rarement bénéfique pour les architectures de micro-frontend impliquant plus d'une ou deux équipes collaborant ensemble. Nous vous recommandons d'investir dans un système de conception dès que possible et de mettre en œuvre une approche de non-partage pendant le développement du système de conception.

Organisation et méthodes de travail

Comme toutes les stratégies architecturales, les micro-frontends ont des implications qui vont bien au-delà de la technologie qu'une organisation choisit de mettre en œuvre. La décision de créer des applications micro-frontales doit être conforme à l'entreprise, au produit, à l'organisation, aux opérations et même à la culture (par exemple, en responsabilisant les équipes et en décentralisant la prise de décision). En retour, ce type d'architecture micro-frontend permet un développement réellement agile et axé sur le produit, car il réduit considérablement les frais de communication entre des équipes par ailleurs indépendantes.

Développement agile

L'idée du développement logiciel agile est devenue si universelle ces dernières années que pratiquement toutes les organisations prétendent travailler de manière agile. Bien qu'une définition concluante de l'agile dépasse le cadre de cette stratégie, il convient de passer en revue les éléments clés pertinents pour le développement de micro-frontend.

Le paradigme agile repose sur le Manifeste agile (2001), qui postule quatre principes principaux (par exemple, « Les individus et les interactions plutôt que les processus et les outils ») et douze principes. Des cadres de processus tels que Scrum et le Scaled Agile Framework (SAFe) sont apparus autour du Manifeste Agile et ont trouvé leur place dans les pratiques quotidiennes. Cependant, la philosophie qui les sous-tend est largement mal comprise ou ignorée.

Dans le contexte de l'architecture micro-frontend, il est important d'adopter les principes agiles suivants :

- « Fournissez fréquemment des logiciels fonctionnels, de quelques semaines à quelques mois, en privilégiant les délais les plus courts. »

  Ce principe souligne à quel point il est important de travailler par étapes et de livrer des logiciels à la production aussi régulièrement et aussi souvent que possible. D'un point de vue technologique, cela fait référence à l'intégration continue et à la livraison continue (CI/CD). Dans CI/CD, les outils et les processus de création, de test et de déploiement font partie intégrante de chaque projet logiciel. Le principe implique également que l'infrastructure d'exécution et les responsabilités opérationnelles doivent appartenir à l'équipe. Cette propriété est particulièrement importante dans les systèmes distribués où les sous-systèmes indépendants peuvent avoir des exigences très différentes en matière d'infrastructure et d'exploitation.

- « Construisez des projets autour de personnes motivées. Donnez-leur l'environnement et le soutien dont ils ont besoin, et faites-leur confiance pour accomplir leur travail. »

  « Les meilleures architectures, exigences et conceptions sont le fruit d'équipes auto-organisées. »

  Ces deux principes mettent l'accent sur les avantages de la propriété, de l'indépendance et de end-to-end la responsabilité. Une architecture de micro-frontend sera efficace lorsque (et seulement lorsque) les équipes seront véritablement propriétaires de leurs micro-frontends. La nd-to-end responsabilité, depuis la conception jusqu'à la livraison et à l'exploitation, en passant par la conception et la mise en œuvre, garantit que l'équipe peut réellement s'approprier. Cette indépendance est requise, tant sur le plan technique qu'organisationnel, pour que l'équipe soit autonome quant à l'orientation stratégique. Nous ne recommandons pas l'utilisation d'une plateforme micro-frontend dans une organisation centralisée qui utilise un modèle de développement en cascade.

Composition et taille de l'équipe

Pour qu'une équipe logicielle puisse exercer ses responsabilités, elle doit se gouverner elle-même, y compris comment et ce qu'elle fournit, dans les limites imposées par l'organisation.

Pour être efficaces, les équipes doivent être en mesure de fournir des logiciels de manière indépendante et avoir le pouvoir de décider de la meilleure façon de les fournir. Une équipe qui reçoit les exigences relatives aux fonctionnalités d'un chef de produit externe ou des conceptions d'interface utilisateur d'un concepteur externe, sans être impliquée dans la planification de ces éléments, ne peut pas être considérée comme autonome. Les fonctionnalités peuvent enfreindre les contrats ou fonctionnalités existants. De telles violations nécessiteront de nouvelles discussions et négociations, avec le risque de retarder la livraison et d'introduire des conflits inutiles entre les équipes.

Dans le même temps, les équipes ne doivent pas devenir trop grandes. Alors qu'une équipe plus nombreuse dispose de plus de ressources et peut s'adapter aux absences individuelles, la complexité de la communication augmente de façon exponentielle avec chaque nouveau membre. Il n'est pas possible de définir une taille d'équipe maximale universellement valide. Le nombre de personnes nécessaires à un projet dépend de facteurs tels que la maturité de l'équipe, la complexité technique, le rythme d'innovation et l'infrastructure. Par exemple, Amazon applique la règle des deux pizzas : une équipe trop nombreuse pour être nourrie de deux pizzas doit être divisée en équipes plus petites. Cela peut être un défi. La scission doit se faire selon des limites naturelles et donner à chaque équipe l'autonomie et la maîtrise de son travail.

DevOps culture

DevOps fait référence à une pratique de génie logiciel dans laquelle les étapes du cycle de développement sont étroitement intégrées du point de vue organisationnel et technique. Contrairement à la croyance populaire, DevOps c'est une question de culture et d'état d'esprit, et très peu de rôles et d'outils.

Traditionnellement, une organisation logicielle devait disposer d'équipes de spécialistes, notamment pour la conception, la mise en œuvre, les tests, le déploiement et les opérations. Chaque fois qu'une équipe terminait son travail, elle confiait le projet à l'équipe suivante. Cependant, la livraison de logiciels par le biais de silos d'équipes spécialisées entraîne des frictions lors des transferts. Dans le même temps, lorsque les spécialistes sont contraints de travailler dans une optique étroite, ils manquent de connaissances dans les domaines voisins et n'ont pas de vision systémique du produit. Ces déficits peuvent entraîner une faible cohérence du produit logiciel.

Par exemple, lorsqu'un architecte logiciel conçoit une solution qui doit être mise en œuvre par un membre d'une autre équipe, il peut négliger certains aspects inhérents à la mise en œuvre (tels que l'inadéquation des dépendances). Les développeurs prennent ensuite des raccourcis (comme un patch en forme de singe), ou une formalisation back-and-forth est initiée entre l'architecte et l'équipe de développement. En raison de la surcharge liée à la gestion de ces processus, le développement n'est plus agile (au sens de flexible, adaptatif, progressif et informel).

Bien que le terme DevOps se rapporte principalement à la culture, il implique les technologies et les processus qui le rendent DevOps possible dans la pratique. DevOps est étroitement lié au CI/CD. Lorsqu'un développeur a fini d'implémenter un incrément de logiciel, il le commet dans un système de contrôle de version tel que Git. Traditionnellement, un système de build créait et intégrait ensuite le logiciel, puis le testait et le publiait dans le cadre d'un processus plus ou moins unifié et centralisé. Avec CI/CD, la création, l'intégration, les tests et la publication du logiciel sont inhérents et automatisés. Idéalement, le processus fait partie du projet logiciel lui-même par le biais de fichiers de configuration spécifiquement adaptés au projet donné.

Le plus grand nombre possible d'étapes sont automatisées. Par exemple, les pratiques de test manuel devraient être réduites, car presque tous les types de tests peuvent être automatisés. Lorsque le projet est configuré de cette manière, les mises à jour d'un produit logiciel peuvent être fournies plusieurs fois par jour en toute confiance. Une autre technologie prise en charge DevOps est l'infrastructure en tant que code (IaC).

Traditionnellement, la configuration et la maintenance de l'infrastructure informatique nécessitaient le travail manuel d'installation et de maintenance du matériel (configuration des câbles et des

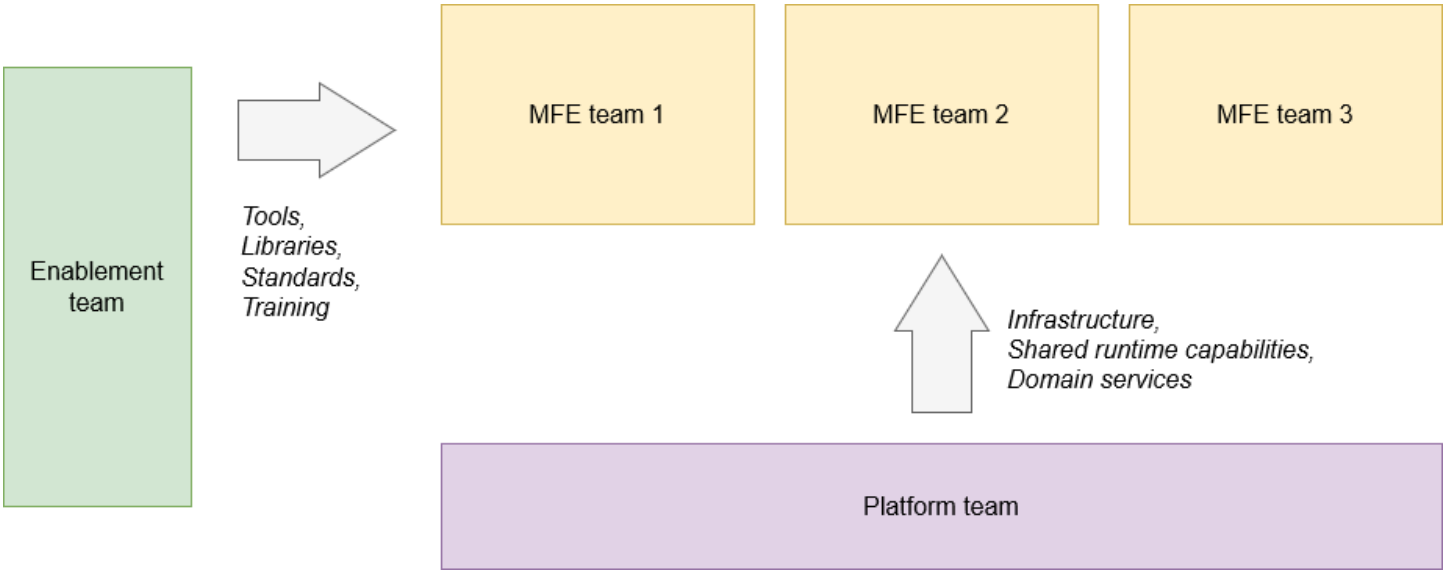
serveurs dans un centre de données) et des logiciels opérationnels. C'était nécessaire, mais cela présentait de nombreux inconvénients. L'installation prenait beaucoup de temps et était source d'erreurs. Le matériel était souvent surprovisionné ou sous-approvisionné, ce qui entraînait des dépenses excessives ou une dégradation des performances. En utilisant iAC, vous pouvez décrire les exigences d'infrastructure d'un système informatique par le biais d'un fichier de configuration à partir duquel les services cloud peuvent être déployés et mis à jour automatiquement.

Qu'est-ce que tout cela a à voir avec les micro-frontends ? DevOps, CI/CD et IaC sont des compléments idéaux à une architecture de micro-frontend. Les avantages des micro-frontends reposent sur des processus de livraison rapides et fluides. Une DevOps culture ne peut prospérer que dans des environnements où les équipes sont end-to-end responsables de projets logiciels.

Orchestrer le développement du micro-frontend entre plusieurs équipes

Lorsque le développement du micro-frontend est étendu à plusieurs équipes interfonctionnelles, deux problèmes apparaissent : tout d'abord, les équipes commencent à développer leur propre interprétation du paradigme, à faire des choix en matière de framework et de bibliothèque, et à créer leurs propres bibliothèques d'outils et d'assistance. Deuxièmement, les équipes totalement autonomes doivent assumer la responsabilité des fonctionnalités génériques telles que la gestion de l'infrastructure de bas niveau. Il est donc logique d'introduire deux équipes supplémentaires dans une organisation micro-frontend à plusieurs équipes : une équipe d'habilitation et une équipe de plateforme. Ces concepts sont largement adoptés dans les organisations informatiques modernes dotées de systèmes distribués et sont bien documentés dans Team Topologies.

Le schéma suivant montre l'équipe d'habilitation fournissant des outils, des bibliothèques, des normes et des tests à trois équipes de micro-frontend. L'équipe de la plateforme fournit une infrastructure, des fonctionnalités d'exécution partagées et des services de domaine à ces trois mêmes équipes de micro-frontend.



L'équipe de la plateforme soutient les équipes du micro-frontend en les libérant du levage de charges lourdes indifférenciées. Ce support peut inclure des services d'infrastructure tels que les temps d'exécution des conteneurs, les pipelines CI/CD, les outils de collaboration et la surveillance. Cependant, la mise en place d'une équipe chargée de la plateforme ne doit pas aboutir à une organisation dans laquelle le développement est dissocié des opérations. Le contraire est vrai : l'équipe de la plateforme propose un produit d'ingénierie, et les équipes du micro-frontend sont propriétaires et responsables de l'exécution de leurs services sur la plateforme.

L'équipe d'habilitation fournit un soutien en se concentrant sur la gouvernance et en garantissant la cohérence entre les équipes du micro-frontend. (L'équipe de la plateforme ne doit pas être impliquée dans cette affaire.) L'équipe d'habilitation gère des ressources partagées telles qu'une bibliothèque d'interface utilisateur et crée des normes telles que le choix du framework, les budgets de performance et les conventions d'interopérabilité. Dans le même temps, il fournit aux nouvelles équipes ou aux nouveaux membres de l'équipe une formation à l'application des normes et des outils tels que définis par la gouvernance.

Déploiement

La clé de voûte de l'autonomie des équipes de micro-frontend est de disposer d'un pipeline automatisé avec un chemin de production indépendant des autres équipes de micro-frontend. Les équipes qui suivent le principe de ne rien partager peuvent mettre en œuvre des pipelines indépendants. Les équipes qui partagent des bibliothèques ou dépendent d'une équipe de plateforme doivent décider comment gérer les dépendances dans les pipelines de déploiement.

En général, chaque pipeline effectue les opérations suivantes :

- Construit des actifs frontaux
- Déploie les actifs vers l'hébergement à des fins de consommation
- Veille à ce que les registres et les caches soient mis à jour afin que les nouvelles versions puissent être livrées aux clients

Les étapes réelles du pipeline varient en fonction de la technologie et de l'approche de composition des pages.

Pour la composition côté client, cela implique de télécharger des ensembles d'applications vers des compartiments d'hébergement et de les distribuer à la consommation via la mise en cache sur un CDN. Les applications qui utilisent la mise en cache du navigateur avec les agents de service doivent également mettre en œuvre des moyens de mettre à jour les caches des agents de service.

Pour la composition côté serveur, cela signifie généralement le déploiement de la nouvelle version du composant serveur et la mise à jour du registre du micro-frontend pour rendre la nouvelle version détectable. Vous pouvez utiliser des modèles de déploiement bleu/vert ou canari pour déployer progressivement de nouvelles versions.

Gouvernance

Plusieurs personas travaillent généralement sur des micro-frontends, et chacun travaille avec des contraintes différentes pour atteindre des objectifs commerciaux communs. Bien que la communication et la collaboration entre les personnes soient essentielles au succès, la surcommunication et la mise en œuvre de processus trop complexes ralentissent le cycle de développement. Cela entraîne une baisse du moral et abaisse la barre de qualité.

Les entreprises les plus performantes qui mettent en œuvre des micro-interfaces en utilisant plusieurs équipes créent des mécanismes pour équilibrer autonomie et alignement. Ils permettent aux décideurs de prendre des mesures au niveau local et de monter hiérarchiquement en cas de besoin. Les mécanismes sont notamment les suivants :

- Contrats d'API
- Interactivité croisée à l'aide d'événements
- Équilibrer autonomie et alignement
- Drapeaux caractéristiques
- Découverte de service

Contrats d'API

Chaque micro-frontend est un système capable d'encapsuler les opinions, la logique et la complexité. Les préoccupations transversales incluent généralement les suivantes :

- Systèmes de conception – Outils pour développer des interfaces utilisateur distribuées sous forme de bibliothèques
- Composition – La façon dont un micro-frontend doit interagir avec le shell d'une application pour afficher et hériter de son contexte
- Gestion logique – Interaction avec les API pour gérer l'état persistant
- Interactivité avec d'autres micro-frontends – Scénarios tels que la publication et la consommation d'événements ou la navigation d'un micro-frontend à un autre

Pour accélérer la consommation et le dépannage, il est courant d'investir dans la standardisation de la manière dont ces interfaces sont déclarées et documentées, y compris les dépendances des micro-frontend. Les wikis créés par des humains sont un bon début. Une approche plus évolutive

consiste à stocker ces informations sous forme de métadonnées structurées dans le code. Vous pouvez ensuite le centraliser pour la consommation en utilisant l'automatisation pour suivre les modifications historiques et fournir une recherche en texte intégral.

Lorsque les micro-frontends impliquent un grand nombre d'équipes, vous avez besoin d'une stratégie de coordination entre les équipes. Le partage unifié des contrats d'API devient indispensable car il réduit les frais de communication et améliore l'expérience des développeurs.

OpenAPI est un langage de spécification pour les API HTTP qui permet de définir des interfaces API et des contrats de manière unifiée. Vous pouvez implémenter des API REST à l'aide d'OpenAPI dans Amazon API Gateway. Vous pouvez également utiliser une grande variété de frameworks open source que vous pouvez héberger dans des conteneurs ou des machines virtuelles. Un avantage significatif est qu'OpenAPI peut générer automatiquement de la documentation dans un format cohérent, ce qui permet à plusieurs équipes de partager leurs connaissances avec un investissement initial minimal.

Lorsque plusieurs équipes travaillent sur des micro-frontends, elles forment souvent des groupes. Dans ces groupes, les gens peuvent se rencontrer et apprendre les uns des autres tout en réfléchissant et en contribuant à la vision d'ensemble. Ces initiatives définissent et documentent généralement les limites de propriété, discutent des préoccupations transversales et identifient dès le début toute duplication des efforts pour résoudre des problèmes communs.

Interactivité croisée à l'aide d'événements

Dans certains scénarios, plusieurs micro-frontends peuvent avoir besoin d'interagir les uns avec les autres pour réagir aux changements d'état ou aux actions des utilisateurs. Par exemple, plusieurs micro-frontends sur la page peuvent inclure des menus pliables. Un menu apparaît lorsque l'utilisateur choisit un bouton. Le menu est masqué lorsque l'utilisateur clique sur un autre menu, y compris sur un autre menu affiché dans un micro-frontend différent.

Techniquement, une bibliothèque d'état partagée telle que Redux peut être utilisée par plusieurs micro-frontends et coordonnée par un shell. Cependant, cela crée un couplage important entre les applications, ce qui rend le code plus difficile à tester et peut ralentir les performances lors du rendu.

Une approche courante et efficace consiste à développer un bus d'événements distribué sous forme de bibliothèque, orchestré par le shell de l'application et utilisé par plusieurs micro-frontends. Ainsi, chaque micro-frontend publie et écoute des événements particuliers de manière asynchrone, en basant son comportement uniquement sur son propre état interne. Plusieurs équipes peuvent ensuite

gérer une page wiki partagée qui décrit les événements et documente les comportements approuvés par les concepteurs d'expérience utilisateur.

Dans une implémentation de l'exemple de bus d'événements, un composant déroulant utilise le bus partagé pour publier un événement appelé `drop-down-open-menu` avec une charge utile de. `{"id": "homepage-aboutus-button"}` Le composant ajoute un écouteur à l'`drop-down-open-menu`événement pour garantir que si un événement est déclenché pour un nouvel identifiant, le composant déroulant est affiché de manière à masquer sa section pliable. De cette façon, le micro-frontend peut réagir aux changements de manière asynchrone avec des performances accrues et une meilleure encapsulation, ce qui permet à plusieurs équipes de concevoir et de tester plus facilement les comportements.

Nous vous recommandons d'utiliser des API standard implémentées nativement par les navigateurs modernes pour améliorer la simplicité et la maintenabilité. La référence MDN Event fournit des informations sur l'utilisation d'événements avec des applications rendues côté client.

Équilibrer autonomie et alignement

Les architectures micro-frontend sont fortement orientées vers l'autonomie des équipes. Cependant, il est important de faire la distinction entre les domaines qui peuvent favoriser la flexibilité et la diversité des approches pour résoudre les problèmes, et les domaines dans lesquels la normalisation est nécessaire pour parvenir à l'alignement. Les hauts dirigeants et les architectes doivent identifier ces domaines dès le début et prioriser les investissements afin d'équilibrer la sécurité, les performances, l'excellence opérationnelle et la fiabilité des micro-frontends. Trouver cet équilibre implique les étapes suivantes : création, test, publication, journalisation, surveillance et alerte de micro-frontend.

Création de micro-frontends

Idéalement, toutes les équipes sont parfaitement alignées afin de maximiser les avantages en termes de performances pour les utilisateurs finaux. Dans la pratique, cela peut être difficile et demander plus d'efforts. Nous vous recommandons de commencer par des directives écrites auxquelles plusieurs équipes peuvent contribuer dans le cadre d'un débat ouvert et transparent. Les équipes peuvent ensuite progressivement adopter le modèle logiciel Cookiecutter, qui permet de créer des outils qui fournissent un moyen unifié de structurer un projet.

En utilisant cette approche, vous pouvez intégrer des opinions et des contraintes. L'inconvénient est que ces outils nécessitent des investissements importants pour leur création et leur maintenance,

et pour garantir que les bloqueurs sont traités rapidement sans affecter la productivité des développeurs.

nd-to-end Test électronique pour les micro-frontends

Les tests unitaires peuvent être laissés aux propriétaires. Nous vous recommandons de mettre en œuvre une stratégie dès le début pour tester de manière croisée les micro-frontends exécutés sur un shell unique. La stratégie inclut la capacité de tester les applications avant et après une version de production. Nous recommandons de développer des processus et de la documentation pour le personnel technique et non technique afin de tester manuellement les fonctionnalités critiques.

Il est important de veiller à ce que les modifications ne dégradent pas l'expérience client fonctionnelle ou non fonctionnelle. Une stratégie idéale consiste à investir progressivement dans des tests automatisés, à la fois pour les fonctionnalités clés et pour les caractéristiques de l'architecture telles que la sécurité et les performances.

Libérer les micro-frontends

Chaque équipe peut avoir sa propre façon de déployer son code, de recueillir des opinions et de disposer de sa propre infrastructure. Le coût de la complexité de la maintenance de tels systèmes est généralement dissuasif. Nous recommandons plutôt d'investir dès le début pour mettre en œuvre une stratégie partagée pouvant être appliquée par des outils partagés.

Développez des modèles avec la plateforme CI/CD de votre choix. Les équipes peuvent ensuite utiliser les modèles préapprouvés et l'infrastructure partagée pour apporter des modifications à la production. Vous pouvez commencer à investir dans ce travail de développement très tôt, car ces systèmes ont rarement besoin de mises à jour importantes après une période initiale de test et de consolidation.

Journalisation et surveillance

Chaque équipe peut avoir différents indicateurs commerciaux et systèmes qu'elle souhaite suivre à des fins opérationnelles ou analytiques. Le modèle du logiciel Cookiecutter peut également être appliqué ici. La diffusion des événements peut être résumée et mise à disposition sous forme de bibliothèque que plusieurs micro-frontends peuvent utiliser. Pour trouver un équilibre entre flexibilité et autonomie, développez des outils permettant de consigner des métriques personnalisées et de créer des tableaux de bord ou des rapports personnalisés. Les rapports favorisent une collaboration étroite avec les responsables des produits et réduisent la boucle de feedback du client final.

En normalisant la livraison, plusieurs équipes peuvent collaborer pour suivre les indicateurs. Par exemple, un site Web de commerce électronique peut suivre le parcours de l'utilisateur depuis le micro-frontend « Détails du produit » au micro-frontend « Panier », en passant par le micro-frontend « Acheter » pour mesurer l'engagement, le taux de désabonnement et les problèmes. Si chaque micro-frontend enregistre les événements à l'aide d'une seule bibliothèque, vous pouvez utiliser ces données dans leur ensemble, les explorer de manière globale et identifier des tendances pertinentes.

Alerte

À l'instar de la journalisation et de la surveillance, les alertes bénéficient de la standardisation avec une certaine flexibilité. Les différentes équipes peuvent réagir différemment aux alertes fonctionnelles et non fonctionnelles. Cependant, si toutes les équipes disposent d'un moyen consolidé pour lancer des alertes sur la base de mesures collectées et analysées sur une plateforme partagée, l'entreprise peut identifier les problèmes entre les équipes. Cette fonctionnalité est utile lors d'événements liés à la gestion des incidents. Par exemple, les alertes peuvent être déclenchées par les moyens suivants :

- Nombre élevé d'exceptions JavaScript côté client sur une version de navigateur donnée
- Temps nécessaire pour obtenir une dégradation significative au-delà d'un seuil donné
- Nombre élevé de codes d'état 5xx lors de l'utilisation d'une API particulière

En fonction de la maturité de votre système, vous pouvez équilibrer vos efforts sur différentes parties de votre infrastructure, comme indiqué dans le tableau suivant.

Adoption	Recherche et développement	Ascension	Maturité
Créez des micro-frontends.	Expérimentez, documentez et partagez les apprentissages.	Investissez dans de l'outillage pour construire de nouvelles micro-frontends. Évangéliser l'adoption.	Consolidez l'outillage pour les échafaudages. Faites pression pour l'adoption.
Testez les micro-frontends	Mettez en œuvre des mécanismes pour tester manuellement	Investissez dans des outils pour automatiser les tests de sécurité et de performance. Étudiez les indicateurs de fonctionn	Consolidez l'outillage pour la découverte des services, les tests en production et les end-to-end tests automatisés.

Adoption	Recherche et développement	Ascension	Maturité
de bout en bout.	tous les micro-frontends associés.	alités et la découverte de services.	
Relâchez les micro-frontends.	Investissez dans une infrastructure CI/CD partagée et dans des versions automatisées dans plusieurs environnements. Évangéliser l'adoption.	Consolider l'outillage pour l'infrastructure CI/CD Mettre en œuvre des mécanismes de restauration manuels. Faites pression pour l'adoption.	Créez des mécanismes pour lancer des annulations automatisées en plus des indicateurs et alertes du système et de l'entreprise.
Observez les performances du micro-frontend.	Investissez dans une infrastructure de surveillance et une bibliothèque partagées pour une journalisation cohérente du système et des événements commerciaux.	Consolidez les outils de surveillance et d'alerte. Implémentez des tableaux de bord inter-équipes pour surveiller l'état général et améliorer la gestion des incidents.	Standardisez les schémas de journalisation. Optimisez en fonction des coûts. Mettez en œuvre des alertes basées sur des indicateurs commerciaux complexes.

Drapeaux caractéristiques

Les indicateurs de fonctionnalités peuvent être implémentés dans les micro-frontends afin de faciliter la coordination des tests et de la publication des fonctionnalités dans plusieurs environnements. La technique des indicateurs de fonctionnalité consiste à centraliser les décisions dans un magasin basé sur le booléen et à définir le comportement en fonction de cela. Il est souvent utilisé pour propager

silencieusement des modifications qui peuvent être masquées jusqu'à un moment précis, tout en débloquant de nouvelles versions pour de nouvelles fonctionnalités qui seraient autrement bloquées, réduisant ainsi la vélocité de l'équipe.

Prenons l'exemple d'équipes travaillant sur une fonctionnalité de micro-frontend qui sera lancée à une date précise. La fonctionnalité est prête, mais elle doit être publiée en même temps qu'une modification sur un autre micro-frontend publié indépendamment. Le blocage de la libération des deux micro-frontends serait considéré comme un anti-modèle et augmenterait les risques lors du déploiement.

Les équipes peuvent plutôt créer un indicateur de fonctionnalité booléen dans une base de données qu'elles utilisent toutes les deux pendant le rendu (par exemple via un appel HTTP à une API Feature Flags partagée). Les équipes peuvent même publier la modification dans un environnement de test où la valeur booléenne est définie `True` pour vérifier les exigences fonctionnelles et non fonctionnelles entre les projets avant le lancement en production.

Un autre exemple d'utilisation d'un indicateur de fonctionnalité consiste à implémenter un mécanisme permettant de remplacer la valeur d'un indicateur en définissant une valeur spécifique via le `QueryString` paramètre ou en stockant une chaîne de test particulière dans un cookie. Les responsables du produit peuvent modifier les fonctionnalités sans bloquer la sortie d'autres fonctionnalités ou les corrections de bogues jusqu'à la date de lancement. À la date donnée, la modification de la valeur du drapeau sur la base de données rend instantanément la modification visible en production, sans qu'il soit nécessaire de coordonner les publications entre les équipes. Après la sortie d'une fonctionnalité, les équipes de développement nettoient le code pour supprimer l'ancien comportement.

Parmi les autres cas d'utilisation, citons la publication d'un système d'indicateurs de fonctionnalités basé sur le contexte. Par exemple, si un seul site Web propose des services aux clients dans plusieurs langues, une fonctionnalité peut être disponible uniquement pour les visiteurs d'un pays en particulier. Le système d'indicateurs de fonctionnalité peut dépendre du fait que le consommateur envoie le contexte du pays (par exemple en utilisant l'en-tête `Accept-Language` HTTP), et le comportement peut être différent en fonction de ce contexte.

Bien que les indicateurs de fonctionnalité soient un outil puissant pour faciliter la collaboration entre les développeurs et les propriétaires de produits, ils dépendent de la diligence des utilisateurs pour éviter une dégradation significative de la base de code. Le fait de maintenir les indicateurs actifs sur plusieurs fonctionnalités peut accroître la complexité lors de la résolution des problèmes, augmenter la taille du JavaScript bundle et, en fin de compte, accumuler des dettes techniques. Les activités d'atténuation courantes sont les suivantes :

- Test unitaire de chaque fonctionnalité derrière un drapeau afin de réduire la probabilité de bogues, ce qui peut introduire des boucles de rétroaction plus longues dans les pipelines CI/CD automatisés qui exécutent les tests
- Création d'outils pour mesurer l'augmentation de la taille des paquets lors des modifications de code, ce qui peut être atténué lors des révisions de code

AWS propose une gamme de solutions pour optimiser les tests A/B en périphérie à l'aide des CloudFront fonctions Amazon ou Lambda @Edge. Ces approches permettent de réduire la complexité de l'intégration d'une solution ou du produit SaaS existant que vous utilisez pour affirmer vos hypothèses. Pour plus d'informations, consultez la section Test A/B.

Découverte de service

Le modèle de découverte du frontend améliore l'expérience de développement lors du développement, du test et de la fourniture de micro-frontends. Le modèle utilise une configuration partageable qui décrit le point d'entrée des micro-frontends. La configuration partageable inclut également des métadonnées supplémentaires qui sont utilisées pour des déploiements sécurisés dans chaque environnement à l'aide des versions de Canary.

Le développement de frontend modernes implique l'utilisation d'une grande variété d'outils et de bibliothèques pour prendre en charge la modularité pendant le développement. Traditionnellement, ce processus consistait à regrouper le code dans des fichiers individuels pouvant être hébergés sur un CDN dans le but de minimiser les appels réseau pendant l'exécution, y compris le chargement initial (lorsqu'une application s'ouvre dans un navigateur) et l'utilisation (lorsqu'un client effectue des actions telles que le choix de boutons ou l'insertion d'informations).

Séparer des lots

Les architectures micro-frontend résolvent les problèmes de performance causés par les très grands ensembles générés par le regroupement individuel d'un grand nombre de fonctionnalités. Par exemple, un très gros site Web de commerce électronique peut être regroupé dans un JavaScript fichier de 6 Mo. Malgré la compression, la taille de ce fichier peut avoir un impact négatif sur l'expérience utilisateur lors du chargement de l'application et du téléchargement du fichier à partir d'un CDN optimisé pour les périphériques.

Si vous divisez l'application en page d'accueil, en détails sur le produit et en micro-frontends de panier, vous pouvez utiliser un mécanisme de regroupement pour produire trois offres groupées

individuelles de 2 Mo. Cette modification peut améliorer les performances lors du premier chargement de 300 % lorsque les utilisateurs consultent la page d'accueil. Les packs de micro-frontends du produit ou du panier sont chargés de manière asynchrone uniquement si et lorsque l'utilisateur visite la page produit d'un article et décide de l'acheter.

De nombreux frameworks et bibliothèques sont disponibles sur la base de cette approche, qui présente des avantages tant pour les clients que pour les développeurs. Pour identifier les limites commerciales susceptibles de découpler les dépendances dans le code, vous pouvez mapper différentes fonctions commerciales à plusieurs équipes. La propriété distribuée apporte indépendance et agilité.

Lorsque vous divisez des packages de construction, vous pouvez utiliser une configuration pour cartographier les micro-frontends et piloter l'orchestration pour le chargement initial et pour la navigation après le chargement. Ensuite, la configuration peut être utilisée pendant l'exécution plutôt que pendant la construction. Par exemple, le code frontal côté client ou le code principal côté serveur peut effectuer un appel réseau initial à une API pour récupérer dynamiquement la liste des micro-frontends. Il récupère également les métadonnées nécessaires à la composition et à l'intégration. Vous pouvez configurer des stratégies de basculement et de mise en cache pour des raisons de fiabilité et de performance. Le mappage des micro-frontends permet de rendre les déploiements individuels de micro-frontends détectables par des micro-frontends précédemment déployés et orchestrés par une application shell.

Versions de Canary

Une version Canary est un modèle bien établi et populaire pour le déploiement de microservices. Les versions de Canary répartissent les utilisateurs cibles d'une version en plusieurs groupes et publient les modifications progressivement, au lieu de les remplacer immédiatement (également connu sous le nom de déploiement bleu/vert). Un exemple de stratégie de publication de Canary consiste à appliquer une nouvelle modification à 10 % des utilisateurs cibles et à en ajouter 10 % par minute, avec une durée totale de 10 minutes pour atteindre 100 %.

L'objectif d'une version de Canary est d'obtenir un feedback précoce sur les modifications, en surveillant le système afin de réduire l'impact des éventuels problèmes. Lorsque l'automatisation est en place, les indicateurs de l'entreprise ou du système peuvent être surveillés par un système interne qui peut arrêter le déploiement ou démarrer une restauration.

Par exemple, une modification peut introduire un bogue qui, dans les premières minutes d'une publication, entraîne une perte de revenus ou une dégradation des performances. La surveillance automatisée peut déclencher une alarme. Grâce au modèle de découverte de services, cette alarme

peut arrêter le déploiement et revenir en arrière immédiatement, affectant seulement 20 % des utilisateurs au lieu de 100 %. L'entreprise bénéficie de la réduction de l'ampleur du problème.

Pour un exemple d'architecture qui utilise DynamoDB comme stockage pour implémenter une API d'administration REST, consultez la solution Frontend Service Discovery on AWS sur. GitHub Utilisez le AWS CloudFormation modèle pour intégrer l'architecture dans vos propres pipelines CI/CD. La solution inclut une API REST Consumer pour intégrer la solution à vos applications frontales.

Avez-vous besoin d'une équipe chargée de la plateforme ?

Certaines entreprises ont une équipe chargée de posséder et de maintenir le code, l'infrastructure et les processus adoptés par d'autres équipes pour travailler sur des micro-frontends. Les responsabilités communes incluent :

- Créez et gérez un pipeline CI/CD qui peut être utilisé avec des référentiels contenant des micro-frontends. Créez et testez des modifications de code, puis publiez-les dans plusieurs environnements.
- Créez et gérez des outils liés à l'observabilité tels que des tableaux de bord partagés, des mécanismes d'alerte et des systèmes pour réagir aux problèmes.
- Créez et gérez des bibliothèques partagées pour la gestion des événements, la consommation de services partagés et les dépendances tierces.
- Créez et gérez des outils qui surveillent en permanence les qualités non fonctionnelles telles que les performances, la sécurité et la fiabilité du système.
- Créez et maintenez des systèmes de conception.
- Créez, gérez et soutenez le shell de l'application pour le système micro-frontend.

Selon l'ampleur du projet, vous pouvez gérer ces responsabilités en utilisant l'une des approches suivantes :

- Créez une équipe dédiée à la plateforme dont la seule responsabilité est de travailler sur des outils partagés.
- Créez un groupe composé de membres de plusieurs équipes. Les membres du groupe partagent leur temps entre travailler sur des micro-frontends et travailler sur des outils partagés. Ceci est également connu sous le nom d'équipe de tigres.

Bien que l'approche de l'équipe de tigres soit un moyen efficace de rester centrée sur le client, une équipe de tigres évolue souvent vers une équipe de plateforme si le projet gagne en popularité et en responsabilités. Tant pour les équipes de plateforme que pour les équipes de type Tiger, les entreprises les plus performantes travaillant sur des micro-frontends forment ces équipes afin que plusieurs personnes aux parcours et compétences multiples puissent apporter leur contribution. Les membres de l'équipe peuvent inclure des ingénieurs backend, des ingénieurs frontaux, des concepteurs d'expérience utilisateur (UX) et des chefs de produits techniques. Cette diversité

pousse les gens à s'engager continuellement dans des débats sains et à concevoir dans un souci de simplicité.

Étapes suivantes

Ce guide traitait des modèles architecturaux et organisationnels, des compromis à faire pour les décisions clés et des problèmes de gouvernance liés aux micro-frontends. Les tableaux résument les compromis des pratiques abordées dans ce document en fonction des dimensions suivantes :

- Autonomie – La capacité de chaque équipe de micro-frontend à faire évoluer indépendamment sa mise en œuvre et à la mettre à la disposition des utilisateurs finaux.
- Cohérence – L'expérience globale de l'application où chaque micro-frontend se comporte comme prévu. Une cohérence élevée signifie que les micro-frontends sont cohérents avec le reste de l'application et ne nuisent pas à l'expérience utilisateur de l'application dans son ensemble.
- Complexité – La quantité d'infrastructure, de code et d'efforts nécessaires pour implémenter et tester les micro-frontends, l'application globale et les contrôles de gouvernance.

Pratique	Autonomie	Cohérence	Complexité
Construire avec des micro-frontends au lieu d'applications monolithiques	Élevée	Medium	Élevée

Pratiques de partage de code	Autonomie	Cohérence	Complexité
Ne rien partager	Élevée	Faible	Faible
Partagez des préoccupations transversales	Medium	Élevée	Medium
Partagez la logique métier	Faible	Élevée	Medium
Partagez via des bibliothèques au moment de la création	Medium	Élevée	Faible
Partager au moment de l'exécution	Élevée	Élevée	Élevée

Pratiques de découverte par micro-frontend	Autonomie	Cohérence	Complexité
Configuration lors de la création de l'application	Faible	Élevée	Faible
Découverte côté serveur	Élevée	Élevée	Medium
Découverte côté client (exécution)	Élevée	Élevée	Medium

Voir les pratiques de composition	Autonomie	Cohérence	Complexité
Montage côté serveur	Élevée	Medium	Élevée
Composition côté bord	Medium	Medium	Élevée
Montage côté client	Élevée	Medium	Medium

Pour en savoir plus sur les concepts présentés dans ce guide, consultez la section Ressources.

Ressources

- Micro-frontends en contexte
- Conception axée sur le domaine
- Visuels EDA
- Découverte du frontend
- Frontend Service Discovery sur AWS
- Manifeste Agile
- Référence d'événement MDN
- OpenAPI

Collaborateurs

Les personnes suivantes ont contribué à ce guide.

- Matteo Figus, architecte de solutions principal, AWS
- Alexander Guensche, architecte de solutions senior, AWS
- Harun Hasdal, architecte de solutions senior, AWS
- Luca Mezzalira, architecte principal de solutions Serverless UK, spécialiste de la mise en marché AWS

Historique du document

Le tableau suivant décrit les modifications importantes apportées à ce guide. Pour être averti des mises à jour à venir, abonnez-vous à un fil RSS.

Modification	Description	Date
Publication initiale	—	12 juillet 2024

AWS Glossaire des recommandations

Les termes suivants sont couramment utilisés dans les politiques, les guides et les modèles fournis par AWS les recommandations. Pour suggérer des entrées, veuillez utiliser le lien Faire un commentaire à la fin du glossaire.

Nombres

7 R

Sept politiques de migration courantes pour transférer des applications vers le cloud. Ces politiques s'appuient sur les 5 R identifiés par Gartner en 2011 et sont les suivantes :

- Refactorisation/réarchitecture : transférez une application et modifiez son architecture en tirant pleinement parti des fonctionnalités natives cloud pour améliorer l'agilité, les performances et la capacité de mise à l'échelle. Cela implique généralement le transfert du système d'exploitation et de la base de données. Exemple : migrez votre base de données Oracle sur site vers Amazon Aurora Édition SQL compatible.
- Replateformer (déplacer et remodeler) : transférez une application vers le cloud et introduisez un certain niveau d'optimisation pour tirer parti des fonctionnalités du cloud. Exemple : migrez votre base de données Oracle sur site vers Amazon Relational Database Service (RDSAmazon) for Oracle dans le. AWS Cloud
- Racheter (rachat) : optez pour un autre produit, généralement en passant d'une licence traditionnelle à un modèle SaaS. Exemple : migrez votre système de gestion de la relation client (CRM) vers Salesforce.com.
- Réhéberger (lift and shift) : transférez une application vers le cloud sans apporter de modifications pour tirer parti des fonctionnalités du cloud. Exemple : migrez votre base de données Oracle sur site vers Oracle sur une EC2 instance dans le AWS Cloud.
- Relocaliser (lift and shift au niveau de l'hyperviseur) : transférez l'infrastructure vers le cloud sans acheter de nouveau matériel, réécrire des applications ou modifier vos opérations existantes. Vous migrez des serveurs d'une plateforme sur site vers un service cloud pour la même plateforme. Exemple : Migrer une Microsoft Hyper-V application à AWS.
- Retenir : conservez les applications dans votre environnement source. Il peut s'agir d'applications nécessitant une refactorisation majeure, que vous souhaitez retarder, et d'applications existantes que vous souhaitez retenir, car rien ne justifie leur migration sur le plan commercial.

- Retirer : mettez hors service ou supprimez les applications dont vous n'avez plus besoin dans votre environnement source.

A

ABAC

Voir contrôle d'accès basé sur les attributs.

services abstraits

Consultez la section Services gérés.

ACID

Voir atomicité, cohérence, isolement, durabilité.

migration active-active

Méthode de migration de base de données dans laquelle la synchronisation des bases de données source et cible est maintenue (à l'aide d'un outil de réplication bidirectionnelle ou d'opérations d'écriture double), tandis que les deux bases de données gèrent les transactions provenant de la connexion d'applications pendant la migration. Cette méthode prend en charge la migration par petits lots contrôlés au lieu d'exiger un basculement ponctuel. Elle est plus flexible, mais demande plus de travail qu'une migration active-passive.

migration active-passive

Méthode de migration de base de données dans laquelle la synchronisation des bases de données source et cible est maintenue, mais seule la base de données source gère les transactions provenant de la connexion d'applications pendant que les données sont répliquées vers la base de données cible. La base de données cible n'accepte aucune transaction pendant la migration.

Fonction d'agrégation

SQLFonction qui agit sur un groupe de lignes et calcule une valeur de retour unique pour le groupe. Des exemples de fonctions d'agrégation incluent `SUM` et`MAX`.

AI

Voir intelligence artificielle.

AIOps

Voir les opérations d'intelligence artificielle.

anonymisation

Processus de suppression définitive d'informations personnelles dans un ensemble de données. L'anonymisation peut contribuer à protéger la vie privée. Les données anonymisées ne sont plus considérées comme des données personnelles.

anti-modèle

Solution fréquemment utilisée pour un problème récurrent lorsque la solution est contre-productive, inefficace ou moins efficace qu'une solution alternative.

contrôle des applications

Une approche de sécurité qui permet d'utiliser uniquement des applications approuvées afin de protéger un système contre les logiciels malveillants.

portefeuille d'applications

Ensemble d'informations détaillées sur chaque application utilisée par une organisation, y compris le coût de génération et de maintenance de l'application, ainsi que sa valeur métier. Ces informations sont essentielles pour le processus de découverte et d'analyse du portefeuille et permettent d'identifier et de prioriser les applications à migrer, à moderniser et à optimiser.

intelligence artificielle (IA)

Domaine de l'informatique consacré à l'utilisation des technologies de calcul pour exécuter des fonctions cognitives généralement associées aux humains, telles que l'apprentissage, la résolution de problèmes et la reconnaissance de modèles. Pour plus d'informations, veuillez consulter Qu'est-ce que l'intelligence artificielle ?

opérations d'intelligence artificielle (AIOps)

Processus consistant à utiliser des techniques de machine learning pour résoudre les problèmes opérationnels, réduire les incidents opérationnels et les interventions humaines, mais aussi améliorer la qualité du service. Pour plus d'informations sur la façon dont AIOps elle est utilisée dans la stratégie de AWS migration, veuillez consulter le guide d'intégration des opérations.

chiffrement asymétrique

Algorithme de chiffrement qui utilise une paire de clés, une clé publique pour le chiffrement et une clé privée pour le déchiffrement. Vous pouvez partager la clé publique, car elle n'est pas utilisée pour le déchiffrement, mais l'accès à la clé privée doit être très restreint.

atomicité, cohérence, isolement, durabilité () ACID

Ensemble de propriétés logicielles garantissant la validité des données et la fiabilité opérationnelle d'une base de données, même en cas d'erreur, de panne de courant ou d'autres problèmes.

contrôle d'accès basé sur les attributs () ABAC

Pratique qui consiste à créer des autorisations détaillées en fonction des attributs de l'utilisateur, tels que le service, le poste et le nom de l'équipe. Pour plus d'informations, consultez ABAC AWS la documentation AWS Identity and Access Management (IAM).

source de données faisant autorité

Emplacement où vous stockez la version principale des données, considérée comme la source d'information la plus fiable. Vous pouvez copier les données de la source de données officielle vers d'autres emplacements à des fins de traitement ou de modification des données, par exemple en les anonymisant, en les expurgant ou en les pseudonymisant.

Zone de disponibilité

Emplacement distinct au sein d'une Région AWS qui est à l'abri des dysfonctionnements d'autres zones de disponibilité et offre une connectivité réseau peu coûteuse et de faible latence par rapport aux autres zones de disponibilité de la même région.

AWS Cadre d'adoption du cloud (AWS CAF)

Un cadre de directives et de bonnes pratiques d' AWS pour aider les entreprises à élaborer un plan efficient et efficace pour réussir leur migration vers le Cloud. AWS CAForganise les conseils en six domaines prioritaires appelés perspectives : l'entreprise, les personnes, la gouvernance, la plateforme, la sécurité et les opérations. Les perspectives d'entreprise, de personnes et de gouvernance mettent l'accent sur les compétences et les processus métier, tandis que les perspectives relatives à la plateforme, à la sécurité et aux opérations se concentrent sur les compétences et les processus techniques. Par exemple, la perspective liée aux personnes cible les parties prenantes qui s'occupent des ressources humaines (RH), des fonctions de dotation en personnel et de la gestion des personnes. Dans cette perspective, AWS CAF fournit des conseils pour le développement du personnel, la formation et les communications afin de préparer l'organisation à une adoption réussie du cloud. Pour plus d'informations, consultez le AWS CAFsite Web et le AWS CAFlivre blanc.

AWS Workload Qualification Framework (AWS WQF)

Outil qui évalue les charges de travail de migration de base de données, recommande des politiques de migration et fournit des estimations de travail. AWS WQFest inclus avec AWS

Schema Conversion Tool (AWS SCT). Il analyse les schémas de base de données et les objets de code, le code d'application, les dépendances et les caractéristiques de performance, et fournit des rapports d'évaluation.

B

robot erroné

Un bot destiné à perturber ou à nuire à des individus ou à des organisations.

BCP

Consultez la section Planification de la continuité des activités.

graphique de comportement

Vue unifiée et interactive des comportements des ressources et des interactions au fil du temps. Vous pouvez utiliser un graphique de comportement avec Amazon Detective pour examiner les tentatives de connexion infructueuses, les API appels suspects et les actions similaires. Pour plus d'informations, veuillez consulter Data in a behavior graph dans la documentation Detective.

système de poids fort

Système qui stocke d'abord l'octet le plus significatif. Voir aussi endianité.

classification binaire

Processus qui prédit un résultat binaire (l'une des deux classes possibles). Par exemple, votre modèle de machine learning peut avoir besoin de prévoir des problèmes tels que « Cet e-mail est-il du spam ou non ? » ou « Ce produit est-il un livre ou une voiture ? ».

filtre de Bloom

Structure de données probabiliste et efficace en termes de mémoire qui est utilisée pour tester si un élément fait partie d'un ensemble.

déploiement bleu/vert

Stratégie de déploiement dans laquelle vous créez deux environnements distincts mais identiques. Vous exécutez la version actuelle de l'application dans un environnement (bleu) et la nouvelle version de l'application dans l'autre environnement (vert). Cette stratégie vous permet de revenir rapidement en arrière avec un impact minimal.

bot

Application logicielle qui exécute des tâches automatisées sur Internet et simule l'activité ou l'interaction humaine. Certains robots sont utiles ou bénéfiques, comme les robots d'indexation qui indexent des informations sur Internet. D'autres robots, appelés « bots malveillants », sont destinés à perturber ou à nuire à des individus ou à des organisations.

réseau d'ordinateurs zombies

Réseaux de robots infectés par des logiciels malveillants et contrôlés par une seule entité, connue sous le nom d'herder ou d'opérateur de bots. Les botnets sont le mécanisme le plus connu pour faire évoluer les bots et leur impact.

branche

Zone contenue d'un référentiel de code. La première branche créée dans un référentiel est la branche principale. Vous pouvez créer une branche à partir d'une branche existante, puis développer des fonctionnalités ou corriger des bogues dans la nouvelle branche. Une branche que vous créez pour générer une fonctionnalité est communément appelée branche de fonctionnalités. Lorsque la fonctionnalité est prête à être publiée, vous fusionnez à nouveau la branche de fonctionnalités dans la branche principale. Pour plus d'informations, consultez À propos des branches (GitHub documentation).

accès par brise-vitre

Dans des circonstances exceptionnelles et par le biais d'un processus approuvé, c'est un moyen rapide pour un utilisateur d'accéder à un accès auquel Compte AWS il n'est généralement pas autorisé. Pour plus d'informations, consultez l'indicateur Implementation break-glass procedures dans le guide Well-Architected AWS .

stratégie existante (brownfield)

L'infrastructure existante de votre environnement. Lorsque vous adoptez une stratégie existante pour une architecture système, vous concevez l'architecture en fonction des contraintes des systèmes et de l'infrastructure actuels. Si vous étendez l'infrastructure existante, vous pouvez combiner des politiques brownfield (existantes) et greenfield (inédites).

cache de tampon

Zone de mémoire dans laquelle sont stockées les données les plus fréquemment consultées.

capacité métier

Ce que fait une entreprise pour générer de la valeur (par exemple, les ventes, le service client ou le marketing). Les architectures de microservices et les décisions de développement

peuvent être dictées par les capacités métier. Pour plus d'informations, veuillez consulter la section Organisation en fonction des capacités métier du livre blanc Exécution de microservices conteneurisés sur AWS.

planification de la continuité des activités (BCP)

Plan qui tient compte de l'impact potentiel d'un événement perturbateur, tel qu'une migration à grande échelle, sur les opérations, et qui permet à une entreprise de reprendre ses activités rapidement.

C

CAF

Voir le cadre d'adoption du AWS cloud.

déploiement de Canary

Diffusion lente et progressive d'une version pour les utilisateurs finaux. Lorsque vous êtes sûr, vous déployez la nouvelle version et remplacez la version actuelle dans son intégralité.

CCoE

Voir Centre d'excellence cloud.

CDC

Consultez la section Capture des données de modification.

capture de données modifiées (CDC)

Processus de suivi des modifications apportées à une source de données, telle qu'une table de base de données, et d'enregistrement des métadonnées relatives à ces modifications. Vous pouvez l'utiliser à diverses CDC fins, telles que l'audit ou la réplication des modifications dans un système cible afin de maintenir la synchronisation.

ingénierie du chaos

Introduire intentionnellement des défaillances ou des événements perturbateurs pour tester la résilience d'un système. Vous pouvez utiliser AWS Fault Injection Service (AWS FIS) pour effectuer des expériences qui stressent vos AWS charges de travail et évaluer leur réponse.

CI/CD

Voir intégration continue et livraison continue.

classification

Processus de catégorisation qui permet de générer des prédictions. Les modèles de ML pour les problèmes de classification prédisent une valeur discrète. Les valeurs discrètes se distinguent toujours les unes des autres. Par exemple, un modèle peut avoir besoin d'évaluer la présence ou non d'une voiture sur une image.

chiffrement côté client

Chiffrement des données en local, avant que le cible ne les Service AWS reçoive.

Centre d'excellence cloud (CCoE)

Une équipe multidisciplinaire qui dirige les efforts d'adoption du cloud au sein d'une organisation, notamment en développant les bonnes pratiques en matière de cloud, en mobilisant des ressources, en établissant des délais de migration et en guidant l'organisation dans le cadre de transformations à grande échelle. Pour plus d'informations, consultez les CCoEpublications sur le blog AWS Cloud Enterprise Strategy.

cloud computing

Technologie cloud généralement utilisée pour le stockage de données à distance et la gestion des appareils IoT. Le cloud computing est généralement lié à la technologie informatique de pointe.

modèle d'exploitation cloud

Dans une organisation informatique, modèle d'exploitation utilisé pour créer, faire évoluer et optimiser un ou plusieurs environnements cloud. Pour plus d'informations, consultez la section Création de votre modèle d'exploitation cloud.

étapes d'adoption du cloud

Les quatre phases que les organisations traversent généralement lorsqu'elles migrent vers AWS Cloud :

- Projet : exécution de quelques projets liés au cloud à des fins de preuve de concept et d'apprentissage
- Base : réaliser des investissements fondamentaux pour mettre à l'échelle l'adoption du cloud (par exemple, en créant une zone de destination, en définissant unCCoE, en établissant un modèle opérationnel)
- Migration : migration d'applications individuelles
- Réinvention : optimisation des produits et services et innovation dans le cloud

Ces étapes ont été définies par Stephen Orban dans le billet de blog The Journey Toward Cloud-First & the Stages of Adoption sur le blog AWS Cloud Enterprise Strategy. Pour en savoir plus sur la façon dont elles sont liées à stratégie de AWS migration, veuillez consulter le guide de préparation à la migration.

CMDB

Consultez la base de données de gestion des configurations.

référentiel de code

Emplacement où le code source et d'autres ressources, comme la documentation, les exemples et les scripts, sont stockés et mis à jour par le biais de processus de contrôle de version. Les référentiels cloud courants incluent GitHub ou AWS CodeCommit. Chaque version du code est appelée branche. Dans une structure de microservice, chaque référentiel est consacré à une seule fonctionnalité. Un seul pipeline CI/CD peut utiliser plusieurs référentiels.

cache passif

Cache tampon vide, mal rempli ou contenant des données obsolètes ou non pertinentes. Cela affecte les performances, car l'instance de base de données doit lire à partir de la mémoire principale ou du disque, ce qui est plus lent que la lecture à partir du cache tampon.

données gelées

Données rarement consultées et généralement historiques. Lorsque vous interrogez ce type de données, les requêtes lentes sont généralement acceptables. Le transfert de ces données vers des niveaux ou des classes de stockage moins performants et moins coûteux peut réduire les coûts.

vision par ordinateur

Domaine de l'IA qui utilise l'apprentissage automatique pour analyser et extraire des informations à partir de formats visuels tels que des images numériques et des vidéos. Par exemple, AWS Panorama propose des appareils qui ajoutent des CV aux réseaux de caméras locaux, et Amazon SageMaker fournit des algorithmes de traitement d'image pour les CV.

dérive de configuration

Pour une charge de travail, une modification de configuration par rapport à l'état attendu. Cela peut entraîner une non-conformité de la charge de travail, et cela est généralement progressif et involontaire.

base de données de gestion des configurations (CMDB)

Référentiel qui stocke et gère les informations relatives à une base de données et à son environnement informatique, y compris les composants matériels et logiciels ainsi que leurs configurations. Vous utilisez généralement les données d'une phase CMDB de découverte et d'analyse du portefeuille de la migration.

pack de conformité

Une collection de AWS Config règles et d'actions correctives que vous pouvez mettre en place pour personnaliser vos contrôles de conformité et de sécurité. Vous pouvez déployer un pack de conformité en tant qu'entité unique dans un Compte AWS et une région, ou au sein d'une organisation, à l'aide d'un YAML modèle. Pour plus d'informations, veuillez consulter Conformance packs dans la AWS Config documentation.

intégration continue et livraison continue (CI/CD)

Processus d'automatisation des étapes source, de génération, de test, intermédiaire et de production du processus de publication du logiciel. CI/CD est communément décrit comme un pipeline. CI/CD peut vous aider à automatiser les processus, à améliorer la productivité, à améliorer la qualité du code et à accélérer les livraisons. Pour plus d'informations, veuillez consulter Avantages de la livraison continue. CD peut également signifier déploiement continu. Pour plus d'informations, veuillez consulter Livraison continue et déploiement continu.

CV

Voir vision par ordinateur.

D

données au repos

Données stationnaires dans votre réseau, telles que les données stockées.

classification des données

Processus permettant d'identifier et de catégoriser les données de votre réseau en fonction de leur sévérité et de leur sensibilité. Il s'agit d'un élément essentiel de toute stratégie de gestion des risques de cybersécurité, car il vous aide à déterminer les contrôles de protection et de conservation appropriés pour les données. La classification des données est une composante du pilier de sécurité du cadre AWS Well-Architected. Pour plus d'informations, veuillez consulter Classification des données.

dérive de données

Une variation significative entre les données de production et les données utilisées pour entraîner un modèle ML, ou une modification significative des données d'entrée au fil du temps. La dérive des données peut réduire la qualité, la précision et l'équité globales des prédictions des modèles ML.

données en transit

Données qui circulent activement sur votre réseau, par exemple entre les ressources du réseau.

maillage de données

Un cadre architectural qui fournit une propriété des données distribuée et décentralisée avec une gestion et une gouvernance centralisées.

minimisation des données

Le principe de collecte et de traitement des seules données strictement nécessaires. La pratique de la minimisation des données AWS Cloud peut réduire les risques liés à la confidentialité, les coûts et l'empreinte carbone de vos analyses.

périmètre de données

Ensemble de garde-fous préventifs dans votre AWS environnement qui permettent de garantir que seules les identités fiables accèdent aux ressources fiables des réseaux attendus. Pour plus d'informations, voir Création d'un périmètre de données sur AWS.

prétraitement des données

Pour transformer les données brutes en un format facile à analyser par votre modèle de ML. Le prétraitement des données peut impliquer la suppression de certaines colonnes ou lignes et le traitement des valeurs manquantes, incohérentes ou en double.

provenance des données

Le processus de suivi de l'origine et de l'historique des données tout au long de leur cycle de vie, par exemple la manière dont les données ont été générées, transmises et stockées.

sujet de données

Personne dont les données sont collectées et traitées.

entrepôt des données

Un système de gestion des données qui prend en charge les informations commerciales, telles que les analyses. Les entrepôts de données contiennent généralement de grandes quantités de données historiques et sont généralement utilisés pour les requêtes et les analyses.

langage de définition de base de données (DDL)

Instructions ou commandes permettant de créer ou de modifier la structure des tables et des objets dans une base de données.

langage de manipulation de base de données (DML)

Instructions ou commandes permettant de modifier (insérer, mettre à jour et supprimer) des informations dans une base de données.

DDL

Voir langage de définition de base de données.

ensemble profond

Sert à combiner plusieurs modèles de deep learning à des fins de prédiction. Vous pouvez utiliser des ensembles profonds pour obtenir une prévision plus précise ou pour estimer l'incertitude des prédictions.

deep learning

Un sous-champ de ML qui utilise plusieurs couches de réseaux neuronaux artificiels pour identifier le mappage entre les données d'entrée et les variables cibles d'intérêt.

defense-in-depth

Approche de la sécurité de l'information dans laquelle une série de mécanismes et de contrôles de sécurité sont judicieusement répartis sur l'ensemble d'un réseau informatique afin de protéger la confidentialité, l'intégrité et la disponibilité du réseau et des données qu'il contient. Lorsque vous adoptez cette stratégie sur AWS, vous ajoutez plusieurs contrôles à différentes couches de la AWS Organizations structure afin de protéger les ressources. Par exemple, une defense-in-depth approche peut combiner l'authentification multifactorielle, la segmentation du réseau et le chiffrement.

administrateur délégué

Dans AWS Organizations, un service compatible peut enregistrer un compte AWS membre pour administrer les comptes de l'organisation et gérer les autorisations pour ce service. Ce compte est

appelé administrateur délégué pour ce service. Pour plus d'informations et une liste des services compatibles, veuillez consulter la rubrique Services qui fonctionnent avec AWS Organizations dans la documentation AWS Organizations .

déploiement

Processus de mise à disposition d'une application, de nouvelles fonctionnalités ou de corrections de code dans l'environnement cible. Le déploiement implique la mise en œuvre de modifications dans une base de code, puis la génération et l'exécution de cette base de code dans les environnements de l'application.

environnement de développement

Voir environnement.

contrôle de détection

Contrôle de sécurité conçu pour détecter, journaliser et alerter après la survenue d'un événement. Ces contrôles constituent une deuxième ligne de défense et vous alertent en cas d'événements de sécurité qui ont contourné les contrôles préventifs en place. Pour plus d'informations, veuillez consulter la rubrique Contrôles de détection dans Implementing security controls on AWS.

cartographie de la chaîne de valeur du développement (DVSM)

Processus utilisé pour identifier et hiérarchiser les contraintes qui nuisent à la rapidité et à la qualité du cycle de vie du développement logiciel. DVSMétend le processus de cartographie de la chaîne de valeur initialement conçu pour les pratiques de production allégée. Il met l'accent sur les étapes et les équipes nécessaires pour créer et transférer de la valeur tout au long du processus de développement logiciel.

jumeau numérique

Représentation virtuelle d'un système réel, tel qu'un bâtiment, une usine, un équipement industriel ou une ligne de production. Les jumeaux numériques prennent en charge la maintenance prédictive, la surveillance à distance et l'optimisation de la production.

table de dimensions

Dans un schéma en étoile, table plus petite contenant les attributs de données relatifs aux données quantitatives d'une table de faits. Les attributs des tables de dimensions sont généralement des champs de texte ou des nombres discrets qui se comportent comme du texte. Ces attributs sont couramment utilisés pour la contrainte des requêtes, le filtrage et l'étiquetage des ensembles de résultats.

catastrophe

Un événement qui empêche une charge de travail ou un système d'atteindre ses objectifs commerciaux sur son site de déploiement principal. Ces événements peuvent être des catastrophes naturelles, des défaillances techniques ou le résultat d'actions humaines, telles qu'une mauvaise configuration involontaire ou une attaque de logiciel malveillant.

reprise après sinistre (DR)

La stratégie et le processus que vous utilisez pour minimiser les temps d'arrêt et les pertes de données causés par un sinistre. Pour plus d'informations, veuillez consulter Reprise après sinistre des charges de travail sur AWS : restauration dans le cloud dans le cadre AWS Well-Architected.

DML

Voir langage de manipulation de base de données.

conception axée sur le domaine

Approche visant à développer un système logiciel complexe en connectant ses composants à des domaines évolutifs, ou objectifs métier essentiels, que sert chaque composant. Ce concept a été introduit par Eric Evans dans son ouvrage Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston : Addison-Wesley Professional, 2003). Pour plus d'informations sur l'utilisation du design piloté par domaine avec le modèle de figuier étrangleur, veuillez consulter Modernizing legacy Microsoft. ASP NET(ASMX) des services Web de manière incrémentielle à l'aide de conteneurs et d'Amazon API Gateway.

DR

Consultez la section Reprise après sinistre.

détection de dérive

Suivi des écarts par rapport à une configuration de référence. Par exemple, vous pouvez l'utiliser AWS CloudFormation pour détecter la dérive des ressources du système ou AWS Control Tower pour détecter les modifications de votre zone d'atterrissage susceptibles d'affecter le respect des exigences de gouvernance.

DVSM

Voir la cartographie de la chaîne de valeur du développement.

E

EDA

Voir analyse exploratoire des données.

informatique de périphérie

Technologie qui augmente la puissance de calcul des appareils intelligents en périphérie d'un réseau IoT. Par rapport au cloud computing, l'informatique de périphérie peut réduire la latence des communications et améliorer le temps de réponse.

chiffrement

Processus informatique qui transforme des données en texte clair, lisibles par l'homme, en texte chiffré.

clé de chiffrement

Chaîne cryptographique de bits aléatoires générée par un algorithme cryptographique. La longueur des clés peut varier, et chaque clé est conçue pour être imprévisible et unique.

endianisme

Ordre selon lequel les octets sont stockés dans la mémoire de l'ordinateur. Les systèmes de poids fort stockent d'abord l'octet le plus significatif. Les systèmes de poids faible stockent d'abord l'octet le moins significatif.

point de terminaison

Voir point de terminaison de service.

service de point de terminaison

Service que vous pouvez héberger sur un cloud privé virtuel (VPC) pour le partager avec d'autres utilisateurs. Vous pouvez créer un service de point de terminaison avec AWS PrivateLink et accorder des autorisations à d'autres Comptes AWS ou à AWS Identity and Access Management (IAM) principaux. Ces comptes ou principaux peuvent se connecter à votre service de point de terminaison de manière privée en créant des points de VPC terminaison d'interface. Pour plus d'informations, veuillez consulter la rubrique Creating an endpoint service dans la documentation Amazon Virtual Private Cloud (AmazonVPC).

planification des ressources d'entreprise (ERP)

Système qui automatise et gère les principaux processus métier (tels que la comptabilité et la gestion de projet) pour une entreprise. MES

chiffrement d'enveloppe

Processus de chiffrement d'une clé de chiffrement à l'aide d'une autre clé de chiffrement. Pour plus d'informations, veuillez consulter la rubrique Envelope encryption dans la documentation AWS Key Management Service (AWS KMS).

environment

Instance d'une application en cours d'exécution. Les types d'environnement les plus courants dans le cloud computing sont les suivants :

- Environnement de développement : instance d'une application en cours d'exécution à laquelle seule l'équipe principale chargée de la maintenance de l'application peut accéder. Les environnements de développement sont utilisés pour tester les modifications avant de les promouvoir dans les environnements supérieurs. Ce type d'environnement est parfois appelé environnement de test.
- Environnements inférieurs : tous les environnements de développement d'une application, tels que ceux utilisés pour les générations et les tests initiaux.
- Environnement de production : instance d'une application en cours d'exécution à laquelle les utilisateurs finaux peuvent accéder. Dans un pipeline CI/CD, l'environnement de production est le dernier environnement de déploiement.
- Environnements supérieurs : tous les environnements accessibles aux utilisateurs autres que l'équipe de développement principale. Ils peuvent inclure un environnement de production, des environnements de préproduction et des environnements pour les tests d'acceptation par les utilisateurs.

épopée

Dans les méthodologies agiles, catégories fonctionnelles qui aident à organiser et à prioriser votre travail. Les épopées fournissent une description détaillée des exigences et des tâches d'implémentation. Par exemple, les épopées en matière de AWS CAF sécurité comprennent la gestion des identités et des accès, les contrôles de détection, la sécurité de l'infrastructure, la protection des données et la réponse aux incidents. Pour plus d'informations sur les épopées dans la stratégie de migration AWS , veuillez consulter le guide d'implémentation du programme.

ERP

Voir Planification des ressources d'entreprise.

analyse exploratoire des données () EDA

Processus d'analyse d'un jeu de données pour comprendre ses principales caractéristiques. Vous collectez ou agrégez des données, puis vous effectuez des enquêtes initiales pour trouver des modèles, détecter des anomalies et vérifier les hypothèses. EDAest réalisée en calculant des statistiques récapitulatives et en créant des visualisations de données.

F

tableau d'informations

La table centrale dans un schéma en étoile. Il stocke des données quantitatives sur les opérations commerciales. Généralement, une table de faits contient deux types de colonnes : celles qui contiennent des mesures et celles qui contiennent une clé étrangère pour une table de dimensions.

échouer rapidement

Une philosophie qui utilise des tests fréquents et progressifs pour réduire le cycle de vie du développement. C'est un élément essentiel d'une approche agile.

limite d'isolation des défauts

Dans le AWS Cloud, une limite telle qu'une zone de disponibilité Région AWS, un plan de contrôle ou un plan de données qui limite l'effet d'une panne et contribue à améliorer la résilience des charges de travail. Pour plus d'informations, consultez AWS Bordures d'isolation des pannes.

branche de fonctionnalités

Voir la succursale.

fonctionnalités

Les données d'entrée que vous utilisez pour faire une prédiction. Par exemple, dans un contexte de fabrication, les fonctionnalités peuvent être des images capturées périodiquement à partir de la ligne de fabrication.

importance des fonctionnalités

Le niveau d'importance d'une fonctionnalité pour les prédictions d'un modèle. Il s'exprime généralement sous la forme d'un score numérique qui peut être calculé à l'aide de différentes techniques, telles que la méthode Shapley Additive Explanations (SHAP) et les gradients intégrés. Pour plus d'informations, veuillez consulter Machine learning model interpretability with :AWS.

transformation de fonctionnalité

Optimiser les données pour le processus de ML, notamment en enrichissant les données avec des sources supplémentaires, en mettant à l'échelle les valeurs ou en extrayant plusieurs ensembles d'informations à partir d'un seul champ de données. Cela permet au modèle de ML de tirer parti des données. Par exemple, si vous décomposez la date « 2021-05-27 00:15:37 » en « 2021 », « mai », « jeudi » et « 15 », vous pouvez aider l'algorithme d'apprentissage à apprendre des modèles nuancés associés à différents composants de données.

FGAC

Voir le contrôle d'accès affiné.

contrôle précis des accès () FGAC

Utilisation de plusieurs conditions pour autoriser ou refuser une demande d'accès.

migration instantanée (flash-cut)

Méthode de migration de base de données qui utilise la réplication continue des données via la capture des données de modification afin de migrer les données dans les plus brefs délais, au lieu d'utiliser une approche progressive. L'objectif est de réduire au maximum les temps d'arrêt.

G

blocage géographique

Voir les restrictions géographiques.

restrictions géographiques (blocage géographique)

Dans Amazon CloudFront, option permettant d'empêcher les utilisateurs de pays spécifiques d'accéder aux distributions de contenu. Vous pouvez utiliser une liste d'autorisation ou une liste de blocage pour spécifier les pays approuvés et interdits. Pour plus d'informations, veuillez consulter la rubrique Restriction de la distribution géographique de votre contenu dans la CloudFront documentation.

Flux de travail Gitflow

Approche dans laquelle les environnements inférieurs et supérieurs utilisent différentes branches dans un référentiel de code source. Le flux de travail Gitflow est considéré comme l'approche héritée, tandis que le flux de travail basé sur les liaisons est l'approche moderne préférée.

stratégie inédite

L'absence d'infrastructures existantes dans un nouvel environnement. Lorsque vous adoptez une stratégie inédite pour une architecture système, vous pouvez sélectionner toutes les nouvelles technologies sans restriction de compatibilité avec l'infrastructure existante, également appelée brownfield. Si vous étendez l'infrastructure existante, vous pouvez combiner des politiques brownfield (existantes) et greenfield (inédites).

barrière de protection

Règle de haut niveau qui permet de régir les ressources, les politiques et la conformité au sein des unités d'organisation (OUs). Les barrières de protection préventives appliquent des politiques pour garantir l'alignement sur les normes de conformité. Elles sont mises en œuvre à l'aide de politiques de contrôle des services et de limites des IAM autorisations. Les barrières de protection de détection détectent les violations des politiques et les problèmes de conformité, et génèrent des alertes pour y remédier. Ils sont implémentés à l'aide d'Amazon AWS Config AWS Security Hub GuardDuty AWS Trusted Advisor, d'Amazon Inspector et de AWS Lambda contrôles personnalisés.

H

HA

Découvrez la haute disponibilité.

migration de base de données hétérogène

Migration de votre base de données source vers une base de données cible qui utilise un moteur de base de données différent (par exemple, Oracle vers Amazon Aurora). La migration hétérogène fait généralement partie d'un effort de réarchitecture, et la conversion du schéma peut s'avérer une tâche complexe. AWS propose AWS SCT qui facilite les conversions de schémas.

haute disponibilité (HA)

Capacité d'une charge de travail à fonctionner en continu, sans intervention, en cas de difficultés ou de catastrophes. Les systèmes HA sont conçus pour basculer automatiquement, fournir constamment des performances de haute qualité et gérer différentes charges et défaillances avec un impact minimal sur les performances.

modernisation de l'historien

Approche utilisée pour moderniser et mettre à niveau les systèmes de technologie opérationnelle (OT) afin de mieux répondre aux besoins de l'industrie manufacturière. Un historien est un type de base de données utilisé pour collecter et stocker des données provenant de diverses sources dans une usine.

migration de base de données homogène

Migration de votre base de données source vers une base de données cible qui partage le même moteur de base de données (par exemple, Microsoft SQL Server vers Amazon RDS for SQL Server). La migration homogène s'inscrit généralement dans le cadre d'un effort de réhébergement ou de replateforme. Vous pouvez utiliser les utilitaires de base de données natifs pour migrer le schéma.

données chaudes

Données fréquemment consultées, telles que les données en temps réel ou les données translationnelles récentes. Ces données nécessitent généralement un niveau ou une classe de stockage à hautes performances pour fournir des réponses rapides aux requêtes.

correctif

Solution d'urgence à un problème critique dans un environnement de production. En raison de son urgence, un correctif est généralement créé en dehors du flux de travail de DevOps publication courant.

période de soins intensifs

Immédiatement après le basculement, période pendant laquelle une équipe de migration gère et surveille les applications migrées dans le cloud afin de résoudre les problèmes éventuels. En règle générale, cette période dure de 1 à 4 jours. À la fin de la période de soins intensifs, l'équipe de migration transfère généralement la responsabilité des applications à l'équipe des opérations cloud.

I

IaC

Considérez l'infrastructure comme un code.

politique basée sur l’identité

Politique attachée à un ou plusieurs IAM principaux qui définit leurs autorisations au sein de l' AWS Cloud environnement.

application inactive

Application dont l'utilisation moyenne CPU de la mémoire se situe entre 5 et 20 % sur une période de 90 jours. Dans un projet de migration, il est courant de retirer ces applications ou de les retenir sur site.

IIoT

Voir internet industriel des objets.

infrastructure immuable

Modèle qui déploie une nouvelle infrastructure pour les charges de travail de production au lieu de mettre à jour, d'appliquer des correctifs ou de modifier l'infrastructure existante. Les infrastructures immuables sont intrinsèquement plus cohérentes, fiables et prévisibles que les infrastructures mutables. Pour plus d'informations, consultez les meilleures pratiques de déploiement à l'aide d'une infrastructure immuable dans le Cadre AWS Well-Architected.

entrant (d'entrée) VPC

Dans une architecture à AWS comptes multiples, VPC qui accepte, inspecte et achemine les connexions réseau depuis l'extérieur d'une application. L'architecture AWS de référence de sécurité recommande de configurer votre compte réseau avec l'entrée, la sortie et l'inspection VPCs afin de protéger l'interface bidirectionnelle entre votre application et Internet en général.

migration incrémentielle

Stratégie de basculement dans le cadre de laquelle vous migrez votre application par petites parties au lieu d'effectuer un basculement complet unique. Par exemple, il se peut que vous ne transfériez que quelques microservices ou utilisateurs vers le nouveau système dans un premier temps. Après avoir vérifié que tout fonctionne correctement, vous pouvez transférer progressivement des microservices ou des utilisateurs supplémentaires jusqu'à ce que vous puissiez mettre hors service votre système hérité. Cette stratégie réduit les risques associés aux migrations de grande ampleur.

Industry 4.0

Un terme introduit par Klaus Schwab en 2016 pour désigner la modernisation des processus de fabrication grâce aux avancées en matière de connectivité, de données en temps réel, d'automatisation, d'analyse et d'IA/ML.

infrastructure

Ensemble des ressources et des actifs contenus dans l'environnement d'une application.

infrastructure en tant que code (IaC)

Processus de mise en service et de gestion de l'infrastructure d'une application via un ensemble de fichiers de configuration. IaC est conçue pour vous aider à centraliser la gestion de l'infrastructure, à normaliser les ressources et à mettre à l'échelle rapidement afin que les nouveaux environnements soient reproductibles, fiables et cohérents.

internet industriel des objets (IIoT)

L'utilisation de capteurs et d'appareils connectés à Internet dans les secteurs industriels tels que la fabrication, l'énergie, l'automobile, les soins de santé, les sciences de la vie et l'agriculture. Pour plus d'informations, veuillez consulter Building an industrial Internet of Things (IIoT) digital transformation strategy.

inspection VPC

Dans une architecture à AWS comptes multiples, système centralisé VPC qui gère les inspections du trafic réseau entre VPCs (dans des identiques ou différentes Régions AWS), Internet et les réseaux sur site. L'architecture AWS de référence de sécurité recommande de configurer votre compte réseau avec l'entrée, la sortie et l'inspection VPCs afin de protéger l'interface bidirectionnelle entre votre application et Internet en général.

Internet des objets (IoT)

Réseau d'objets physiques connectés dotés de capteurs ou de processeurs intégrés qui communiquent avec d'autres appareils et systèmes via Internet ou via un réseau de communication local. Pour plus d'informations, veuillez consulter la section Qu'est-ce que l'IoT ?.

interprétabilité

Caractéristique d'un modèle de machine learning qui décrit dans quelle mesure un être humain peut comprendre comment les prédictions du modèle dépendent de ses entrées. Pour plus d'informations, veuillez consulter Machine learning model interpretability with AWS.

IoT

Voir Internet des objets.

bibliothèque d'informations informatiques (ITIL)

Ensemble de bonnes pratiques pour proposer des services informatiques et les aligner sur les exigences métier. ITILconstitue la base deITSM.

Gestion des services informatiques (ITSM)

Activités associées à la conception, à la mise en œuvre, à la gestion et à la prise en charge de services informatiques d'une organisation. Pour en savoir plus sur l'intégration des opérations cloud aux ITSM outils, veuillez consulter le guide d'intégration des opérations.

ITIL

Consultez la bibliothèque d'informations informatiques.

ITSM

Voir Gestion des services informatiques.

L

contrôle d'accès par étiquette () LBAC

Une implémentation du contrôle d'accès obligatoire (MAC) dans laquelle une valeur d'étiquette de sécurité est explicitement attribuée aux utilisateurs et aux données elles-mêmes. L'intersection entre l'étiquette de sécurité utilisateur et l'étiquette de sécurité des données détermine les lignes et les colonnes visibles par l'utilisateur.

zone de destination

Une zone de destination est un AWS environnement à comptes multiples Well-Architected évolutif et sécurisé. Il s'agit d'un point de départ à partir duquel vos entreprises peuvent rapidement lancer et déployer des charges de travail et des applications en toute confiance dans leur environnement de sécurité et d'infrastructure. Pour plus d'informations sur les zones de destination, veuillez consulter Setting up a secure and scalable multi-account AWS environment.

migration de grande envergure

Migration de 300 serveurs ou plus.

LBAC

Voir contrôle d'accès basé sur des étiquettes.

principe de moindre privilège

Bonne pratique de sécurité qui consiste à accorder les autorisations minimales nécessaires à l'exécution d'une tâche. Pour plus d'informations, veuillez consulter la rubrique Accorder les autorisations de moindre privilège dans la IAM documentation.

lift and shift

Voir 7 Rs.

système de poids faible

Système qui stocke d'abord l'octet le moins significatif. Voir aussi endianité.

environnements inférieurs

Voir environnement.

M

machine learning (ML)

Type d'intelligence artificielle qui utilise des algorithmes et des techniques pour la reconnaissance et l'apprentissage de modèles. Le ML analyse et apprend à partir de données enregistrées, telles que les données de l'Internet des objets (IoT), pour générer un modèle statistique basé sur des modèles. Pour plus d'informations, veuillez consulter Machine Learning.

branche principale

Voir la succursale.

malware

Logiciel conçu pour compromettre la sécurité ou la confidentialité de l'ordinateur. Les logiciels malveillants peuvent perturber les systèmes informatiques, divulguer des informations sensibles ou obtenir un accès non autorisé. Les virus, les vers, les rançongiciels, les chevaux de Troie, les logiciels espions et les enregistreurs de frappe.

services gérés

Services AWS pour lequel AWS fonctionnent la couche d'infrastructure, le système d'exploitation et les plateformes, et vous accédez aux points de terminaison pour stocker et récupérer des données. Amazon Simple Storage Service (Amazon S3) et Amazon DynamoDB sont des exemples de services gérés. Ils sont également connus sous le nom de services abstraits.

système d'exécution de la fabrication (MES)

Un système logiciel pour le suivi, la surveillance, la documentation et le contrôle des processus de production qui convertissent les matières premières en produits finis dans l'atelier.

MAP

Voir Migration Acceleration Program.

mécanisme

Processus complet au cours duquel vous créez un outil, favorisez son adoption, puis inspectez les résultats afin de procéder aux ajustements nécessaires. Un mécanisme est un cycle qui se renforce et s'améliore lorsqu'il fonctionne. Pour plus d'informations, veuillez consulter Building mechanisms in the AWS Well-Architected Framework.

compte membre

Tous les Comptes AWS autres que le compte de gestion qui font partie d'une organisation dans AWS Organizations. Un compte ne peut être membre que d'une seule organisation à la fois.

MES

Voir le système d'exécution de la fabrication.

Transport de télémétrie par file d'attente de messages () MQTT

Protocole de communication léger machine-to-machine (M2M), basé sur le modèle de publication/d'abonnement, pour les appareils IoT aux ressources limitées.

microservice

Petit service indépendant qui communique via un réseau bien défini APIs et qui est généralement détenu par de petites équipes autonomes. Par exemple, un système d'assurance peut inclure des microservices qui mappent à des capacités métier, telles que les ventes ou le marketing, ou à des sous-domaines, tels que les achats, les réclamations ou l'analytique. Les avantages des microservices incluent l'agilité, la flexibilité de la mise à l'échelle, la facilité de déploiement, la réutilisation du code et la résilience. Pour plus d'informations, veuillez consulter Integrating microservices by using AWS serverless services.

architecture de microservices

Approche de création d'une application avec des composants indépendants qui exécutent chaque processus d'application en tant que microservice. Ces microservices communiquent via une interface bien définie à l'aide d'une interface légère. APIs Chaque microservice de cette architecture peut être mis à jour, déployé et mis à l'échelle pour répondre à la demande de fonctions spécifiques d'une application. Pour de plus amples informations, veuillez consulter Implémentation de microservices sur AWS.

Migration Program (MAP)

AWS Programme qui fournit un support de conseil, des formations et des services pour aider les entreprises à générer une base opérationnelle solide pour passer au cloud et pour aider à compenser le coût initial des migrations. MAPinclut une méthodologie de migration pour exécuter les migrations héritées de manière méthodique, ainsi qu'un ensemble d'outils pour automatiser et accélérer les scénarios de migration courants.

migration à grande échelle

Processus consistant à transférer la majeure partie du portefeuille d'applications vers le cloud par vagues, un plus grand nombre d'applications étant déplacées plus rapidement à chaque vague. Cette phase utilise les bonnes pratiques et les enseignements tirés des phases précédentes pour implémenter une usine de migration d'équipes, d'outils et de processus en vue de rationaliser la migration des charges de travail grâce à l'automatisation et à la livraison agile. Il s'agit de la troisième phase de la stratégie de migration AWS.

usine de migration

Équipes interfonctionnelles qui rationalisent la migration des charges de travail grâce à des approches automatisées et agiles. Les équipes de l'usine de migration comprennent généralement des responsables des opérations, des analystes métier et des propriétaires, des ingénieurs de migration, des développeurs et DevOps des professionnels travaillant dans des sprints. Entre 20 et 50 % du portefeuille d'applications d'entreprise est constitué de modèles répétés qui peuvent être optimisés par une approche d'usine. Pour plus d'informations, veuillez consulter la rubrique discussion of migration factories et le guide Cloud Migration Factory dans cet ensemble de contenus.

métadonnées de migration

Informations relatives à l'application et au serveur nécessaires pour finaliser la migration. Chaque modèle de migration nécessite un ensemble de métadonnées de migration différent. Les exemples de métadonnées de migration incluent le sous-réseau cible, le groupe de sécurité et le AWS compte.

modèle de migration

Tâche de migration reproductible qui détaille la stratégie de migration, la destination de la migration et l'application ou le service de migration utilisé. Exemple : réhéberger la migration vers Amazon EC2 avec AWS Application Migration Service.

Évaluation du portefeuille de migration (MPA)

Outil en ligne qui fournit des informations pour valider l'analyse de rentabilisation en faveur de la migration vers. AWS Cloud MPApropose une évaluation détaillée du portefeuille (dimensionnement approprié des serveurs, tarification, TCO comparaison, analyse des coûts de migration), ainsi que la planification de la migration (analyse et collecte des données d'applications, regroupement des applications, priorisation des migrations et planification des vagues). L'MPAoutil (connexion requise) est mis gratuitement à la disposition de tous les AWS consultants et consultants APN partenaires.

Évaluation de la préparation à la migration (MRA)

Processus qui consiste à obtenir des informations sur l'état de préparation au cloud d'une entreprise, à identifier les forces et les faiblesses, ainsi qu'à élaborer un plan d'action pour combler les lacunes identifiées, à l'aide du AWS CAF. Pour plus d'informations, veuillez consulter le guide de préparation à la migration. MRAest la première phase de la stratégie de AWS migration.

stratégie de migration

Approche utilisée pour migrer une charge de travail vers AWS Cloud. Pour plus d'informations, veuillez consulter l'entrée 7 R dans ce glossaire et Mobilize your organization to accelerate large-scale migrations.

ML

Voir apprentissage automatique.

modernisation

Transformation d'une application obsolète (héritée ou monolithique) et de son infrastructure en un système agile, élastique et hautement disponible dans le cloud afin de réduire les coûts, de gagner en efficacité et de tirer parti des innovations. Pour plus d'informations, consultez Strategy for modernizing applications dans le AWS Cloud.

évaluation de la préparation à la modernisation

Évaluation qui permet de déterminer si les applications d'une organisation sont prêtes à être modernisées, d'identifier les avantages, les risques et les dépendances, et qui détermine dans quelle mesure l'organisation peut prendre en charge l'état futur de ces applications. Le résultat de l'évaluation est un plan de l'architecture cible, une feuille de route détaillant les phases de développement et les étapes du processus de modernisation, ainsi qu'un plan d'action

pour combler les lacunes identifiées. Pour plus d'informations, veuillez consulter Evaluating modernizing approach for applications in the AWS Cloud.

applications monolithiques (monolithes)

Applications qui s'exécutent en tant que service unique avec des processus étroitement couplés. Les applications monolithiques ont plusieurs inconvénients. Si une fonctionnalité de l'application connaît un pic de demande, l'architecture entière doit être mise à l'échelle. L'ajout ou l'amélioration des fonctionnalités d'une application monolithique devient également plus complexe lorsque la base de code s'élargit. Pour résoudre ces problèmes, vous pouvez utiliser une architecture de microservices. Pour plus d'informations, veuillez consulter Decomposing monoliths into microservices.

MPA

Voir Évaluation du portefeuille de migration.

MQTT

Voir Transport de télémétrie par file d'attente de messages.

classification multi-classes

Processus qui permet de générer des prédictions pour plusieurs classes (prédiction d'un résultat parmi plus de deux). Par exemple, un modèle de ML peut demander « Ce produit est-il un livre, une voiture ou un téléphone ? » ou « Quelle catégorie de produits intéresse le plus ce client ? ».

infrastructure mutable

Modèle qui met à jour et modifie l'infrastructure existante pour les charges de travail de production. Pour améliorer la cohérence, la fiabilité et la prévisibilité, le AWS Well-Architected Framework recommande l'utilisation d'une infrastructure immuable comme meilleure pratique.

O

OAC

Voir Contrôle d'accès à l'origine.

OAI

Voir l'identité d'accès à l'origine.

OCM

Voir gestion du changement organisationnel.

migration hors ligne

Méthode de migration dans laquelle la charge de travail source est supprimée au cours du processus de migration. Cette méthode implique un temps d'arrêt prolongé et est généralement utilisée pour de petites charges de travail non critiques.

OI

Consultez la section Intégration des opérations.

OLA

Voir accord au niveau opérationnel.

migration en ligne

Méthode de migration dans laquelle la charge de travail source est copiée sur le système cible sans être mise hors ligne. Les applications connectées à la charge de travail peuvent continuer à fonctionner pendant la migration. Cette méthode implique un temps d'arrêt nul ou minimal et est généralement utilisée pour les charges de travail de production critiques.

OPC-États-Unis

Voir Open Process Communications - Architecture unifiée.

Communications par processus ouvert - Architecture unifiée (OPC-UA)

Un protocole de communication machine-to-machine (M2M) pour l'automatisation industrielle. OPC-UA fournit une norme d'interopérabilité avec des schémas de cryptage, d'authentification et d'autorisation des données.

accord au niveau opérationnel () OLA

Accord qui précise ce que les groupes informatiques fonctionnels s'engagent à fournir les uns aux autres, afin de prendre en charge un contrat de niveau de service (). SLA

examen de l'état de préparation opérationnelle (ORR)

Une liste de questions et de bonnes pratiques associées qui vous aident à comprendre, à évaluer, à prévenir ou à réduire l'ampleur des incidents et des défaillances possibles. Pour plus d'informations, voir Operational Readiness Reviews (ORR) dans le AWS Well-Architected Framework.

technologie opérationnelle (OT)

Systèmes matériels et logiciels qui fonctionnent avec l'environnement physique pour contrôler les opérations, les équipements et les infrastructures industriels. Dans le secteur manufacturier, l'intégration des systèmes OT et des technologies de l'information (IT) est au cœur des transformations de l'industrie 4.0.

intégration des opérations (OI)

Processus de modernisation des opérations dans le cloud, qui implique la planification de la préparation, l'automatisation et l'intégration. Pour en savoir plus, veuillez consulter le guide d'intégration des opérations.

journal de suivi d'organisation

Journal de suivi créé par AWS CloudTrail qui journalise tous les événements pour tous les Comptes AWS dans une organisation dans AWS Organizations. Ce journal de suivi est créé dans chaque Compte AWS qui fait partie de l'organisation et suit l'activité de chaque compte. Pour plus d'informations, veuillez consulter la rubrique Creating a trail for an organization dans la CloudTrail documentation.

gestion du changement organisationnel (OCM)

Cadre pour gérer les transformations métier majeures et perturbatrices du point de vue des personnes, de la culture et du leadership. OCMaide les organisations à se préparer et à effectuer la transition vers de nouveaux systèmes et de nouvelles politiques en accélérant l'adoption des changements, en abordant les problèmes de transition et en favorisant des changements culturels et organisationnels. Dans la stratégie de AWS migration, ce cadre s'appelle accélération des personnes, en raison de la rapidité du changement requise dans les projets d'adoption du cloud. Pour plus d'informations, consultez le OCMguide.

contrôle d'accès d'origine (OAC)

Dans CloudFront, option améliorée qui permet de restreindre l'accès à votre contenu Amazon Simple Storage Service (Amazon S3). OACprend en charge tous les compartiments S3 Régions AWS, le chiffrement côté serveur avec AWS KMS (SSE-KMS) et les `DELETE` requêtes dynamiques `PUT` adressées au compartiment S3.

identité d'accès d'origine (OAI)

Dans CloudFront, option qui permet de restreindre l'accès à votre contenu Amazon S3. Lorsque vous utilisezOAI, CloudFront crée un principal auprès duquel Amazon S3 peut s'authentifier. Les

principaux authentifiés peuvent accéder au contenu dans un compartiment S3 uniquement via une distribution spécifique CloudFront . Voir également OAC, qui fournit un contrôle d'accès plus précis et amélioré.

ORR

Voir l'examen de l'état de préparation opérationnelle.

DE

Voir technologie opérationnelle.

sortant (de sortie) VPC

Dans une architecture à AWS comptes multiples, VPC qui gère les connexions réseau initiées depuis une application. L'architecture AWS de référence de sécurité recommande de configurer votre compte réseau avec l'entrée, la sortie et l'inspection VPCs afin de protéger l'interface bidirectionnelle entre votre application et Internet en général.

P

limite des autorisations

Politique de IAM gestion attachée IAM aux principaux pour définir les autorisations maximales que peut avoir l'utilisateur ou le rôle. Pour plus d'informations, veuillez consulter la rubrique Limites des autorisations dans la IAM documentation.

données d'identification personnelle (PII)

Informations qui, lorsqu'elles sont consultées directement ou associées à d'autres données connexes, peuvent être utilisées pour déduire raisonnablement l'identité d'une personne. PIILes exemples incluent les noms, les adresses et les coordonnées.

PII

Voir les informations personnelles identifiables.

manuel stratégique

Ensemble d'étapes prédéfinies qui capturent le travail associé aux migrations, comme la fourniture de fonctions d'opérations de base dans le cloud. Un manuel stratégique peut revêtir la forme de scripts, de runbooks automatisés ou d'un résumé des processus ou des étapes nécessaires au fonctionnement de votre environnement modernisé.

PLC

Voir contrôleur logique programmable.

PLM

Consultez la section Gestion du cycle de vie des produits.

politique

Objet qui permet de définir des autorisations (voir Politique basée sur l'identité), de spécifier les conditions d'accès (voir Politique de contrôle de service) ou de définir les autorisations maximales pour tous les comptes d'une organisation dans AWS Organizations (voir Politique de contrôle de service).

persistance polyglotte

Choix indépendant de la technologie de stockage de données d'un microservice en fonction des modèles d'accès aux données et d'autres exigences. Si vos microservices utilisent la même technologie de stockage de données, ils peuvent rencontrer des difficultés d'implémentation ou présenter des performances médiocres. Les microservices sont plus faciles à mettre en œuvre, atteignent de meilleures performances, ainsi qu'une meilleure capacité de mise à l'échelle s'ils utilisent l'entrepôt de données le mieux adapté à leurs besoins. Pour plus d'informations, veuillez consulter Enabling data persistence in microservices.

évaluation du portefeuille

Processus de découverte, d'analyse et de priorisation du portefeuille d'applications afin de planifier la migration. Pour plus d'informations, veuillez consulter Evaluating migration readiness.

predicate

Une condition de requête qui renvoie `true` ou`false`, généralement située dans une `WHERE` clause.

pushdown de predicate

Technique d'optimisation des requêtes de base de données qui filtre les données de la requête avant le transfert. Cela permet de réduire la quantité de données devant être extraites et traitées à partir de la base de données relationnelle, et d'améliorer les performances des requêtes.

contrôle préventif

Contrôle de sécurité conçu pour empêcher qu'un événement ne se produise. Ces contrôles constituent une première ligne de défense pour empêcher tout accès non autorisé ou toute

modification indésirable de votre réseau. Pour plus d'informations, veuillez consulter Preventative controls dans Implementing security controls on AWS.

principal

Une entité d' AWS qui peut exécuter des actions et accéder à des ressources. Cette entité est généralement un utilisateur root pour un Compte AWS, un IAM rôle ou un utilisateur. Pour plus d'informations, consultez les termes et concepts de Principal in Roles dans la IAM documentation.

Protection de la confidentialité dès la conception

Une approche de l'ingénierie des systèmes qui prend en compte la confidentialité tout au long du processus d'ingénierie.

zones hébergées privées

Conteneur qui contient des informations concernant la façon dont vous souhaitez qu'Amazon Route 53 réponde aux DNS requêtes concernant un domaine et ses sous-domaines dans un ou plusieursVPCs. Pour plus d'informations, veuillez consulter Working with private hosted zones dans la documentation Route 53.

contrôle proactif

Utilisation d'un contrôle de sécurité conçu pour empêcher le déploiement de ressources non conformes. Ces contrôles analysent les ressources avant qu'elles ne soient provisionnées. Si la ressource n'est pas conforme au contrôle, elle n'est pas provisionnée. Pour plus d'informations, consultez le guide de référence sur les contrôles dans la AWS Control Tower documentation et consultez la section Contrôles proactifs dans Implémentation des contrôles de sécurité sur AWS.

gestion du cycle de vie des produits (PLM)

Gestion des données et des processus d'un produit tout au long de son cycle de vie, depuis la conception, le développement et le lancement, en passant par la croissance et la maturité, jusqu'au déclin et au retrait.

environnement de production

Voir environnement.

contrôleur logique programmable (PLC)

Dans le secteur manufacturier, un ordinateur hautement fiable et adaptable qui surveille les machines et automatise les processus de fabrication.

pseudonymisation

Processus de remplacement des identifiants personnels dans un ensemble de données par des valeurs fictives. La pseudonymisation peut contribuer à protéger la vie privée. Les données pseudonymisées sont toujours considérées comme des données personnelles.

publier/souscrire (pub/sub)

Modèle qui permet les communications asynchrones entre les microservices afin d'améliorer l'évolutivité et la réactivité. Par exemple, dans un environnement basé sur des microservices MES, un microservice peut publier des messages d'événements sur un canal auquel d'autres microservices peuvent s'abonner. Le système peut ajouter de nouveaux microservices sans modifier le service de publication.

Q

plan de requête

Série d'étapes, telles que des instructions, utilisées pour accéder aux données d'un système de base de données SQL relationnelle.

régression du plan de requêtes

Le cas où un optimiseur de service de base de données choisit un plan moins optimal qu'avant une modification donnée de l'environnement de base de données. Cela peut être dû à des changements en termes de statistiques, de contraintes, de paramètres d'environnement, de liaisons de paramètres de requêtes et de mises à jour du moteur de base de données.

R

RACImatrice

Voir responsable, redevable, consulté et informé (RACI).

rançongiciel

Logiciel malveillant conçu pour bloquer l'accès à un système informatique ou à des données jusqu'à ce qu'un paiement soit effectué.

RASCImatrice

Voir responsable, redevable, consulté et informé (RACI).

RCAC

Voir contrôle d'accès aux lignes et aux colonnes.

réplica en lecture

Copie d'une base de données utilisée en lecture seule. Vous pouvez acheminer les requêtes vers le réplica de lecture pour réduire la charge sur votre base de données principale.

réarchitecture

Voir 7 Rs.

objectif de point de récupération (RPO)

Durée maximale acceptable depuis le dernier point de récupération des données. Il détermine ce qui est considéré comme étant une perte de données acceptable entre le dernier point de reprise et l'interruption du service.

objectif de temps de récupération (RTO)

Délai maximal acceptable entre l'interruption du service et son rétablissement.

refactoriser

Voir 7 Rs.

Région

Ensemble de AWS ressources dans une zone géographique. Chaque Région AWS est isolée et indépendante des autres pour assurer la tolérance aux pannes, la stabilité et la résilience. Pour plus d'informations, voir Spécifier ce que Régions AWS votre compte peut utiliser.

régression

Technique de ML qui prédit une valeur numérique. Par exemple, pour résoudre le problème « Quel sera le prix de vente de cette maison ? », un modèle de ML pourrait utiliser un modèle de régression linéaire pour prédire le prix de vente d'une maison sur la base de faits connus à son sujet (par exemple, la superficie en mètres carrés).

réhébergement

Voir 7 Rs.

version

Dans un processus de déploiement, action visant à promouvoir les modifications apportées à un environnement de production.

déplacer

Voir 7 Rs.

replateforme

Voir 7 Rs.

rachat

Voir 7 Rs.

résilience

La capacité d'une application à résister aux perturbations ou à s'en remettre. La haute disponibilité et la reprise après sinistre sont des considérations courantes lors de la planification de la résilience dans le AWS Cloud. Pour de plus amples informations, veuillez consulter AWS Cloud Résilience.

politique basée sur les ressources

Politique attachée à une ressource, comme un compartiment Amazon S3, un point de terminaison ou une clé de chiffrement. Ce type de politique précise les principaux auxquels l'accès est autorisé, les actions prises en charge et toutes les autres conditions qui doivent être remplies.

matrice responsable, redevable, consulté et informé (RACI)

Une matrice qui définit les rôles et les responsabilités de toutes les parties impliquées dans les activités de migration et les opérations cloud. Le nom de la matrice est dérivé des types de responsabilité définis dans la matrice : responsable (R), responsable (A), consulté (C) et informé (I). Le type de support (S) est facultatif. Si vous incluez le support, la matrice est appelée RASCImatrice, et si vous l'excluez, elle est appelée RACImatrice.

contrôle réactif

Contrôle de sécurité conçu pour permettre de remédier aux événements indésirables ou aux écarts par rapport à votre référence de sécurité. Pour plus d'informations, veuillez consulter la rubrique Responsive controls dans Implementing security controls on AWS.

retain

Voir 7 Rs.

se retirer

Voir 7 Rs.

rotation

Processus où vous mettez à jour le secret périodiquement pour rendre l'accès au informations d'identification plus difficile pour un pirate informatique.

contrôle d'accès aux lignes et aux colonnes (RCAC)

L'utilisation d'SQLexpressions simples et flexibles qui ont défini des règles d'accès. RCACconsiste en des autorisations de ligne et des masques de colonnes.

RPO

Voir l'objectif du point de récupération.

RTO

Voir l'objectif en matière de temps de rétablissement.

runbook

Ensemble de procédures manuelles ou automatisées nécessaires à l'exécution d'une tâche spécifique. Elles visent généralement à rationaliser les opérations ou les procédures répétitives présentant des taux d'erreur élevés.

S

SAML2.0

Norme ouverte utilisée par de nombreux fournisseurs d'identité (IdPs). Cette fonctionnalité active l'authentification unique fédérée (SSO), permettant aux utilisateurs de se connecter à AWS Management Console ou d'appeler les AWS API opérations sans qu'il soit nécessaire de créer un identifiant utilisateur IAM pour chaque membre de l'organisation. Pour plus d'informations sur la fédération SAML 2.0, veuillez consulter À propos de la fédération SAML 2.0 dans la documentation. IAM

SCADA

Voir Contrôle de supervision et acquisition de données.

SCP

Voir la politique de contrôle des services.

secret

Dans AWS Secrets Manager des informations confidentielles ou restreintes, telles qu'un mot de passe ou des informations d'identification utilisateur, que vous stockez sous forme cryptée. Il comprend la valeur secrète et ses métadonnées. La valeur secrète peut être binaire, une chaîne unique ou plusieurs chaînes. Pour plus d'informations, consultez Que contient le secret d'un Secrets Manager ? dans la documentation Secrets Manager.

contrôle de sécurité

Barrière de protection technique ou administrative qui empêche, détecte ou réduit la capacité d'un assaillant d'exploiter une vulnérabilité de sécurité. Il existe quatre principaux types de contrôles de sécurité : préventifs, détectifs, réactifs et proactifs.

renforcement de la sécurité

Processus qui consiste à réduire la surface d'attaque pour la rendre plus résistante aux attaques. Cela peut inclure des actions telles que la suppression de ressources qui ne sont plus requises, la mise en œuvre des bonnes pratiques de sécurité consistant à accorder le moindre privilège ou la désactivation de fonctionnalités inutiles dans les fichiers de configuration.

système de gestion des informations et des événements de sécurité (SIEM)

Outils et services qui associent les systèmes de gestion des informations de sécurité (SIM) et de gestion des événements de sécurité (SEM). Un SIEM système collecte, surveille et analyse les données provenant de serveurs, de réseaux, d'appareils et d'autres sources afin de détecter les menaces et les failles de sécurité, mais aussi de générer des alertes.

automatisation des réponses de sécurité

Action prédéfinie et programmée conçue pour répondre automatiquement à un événement de sécurité ou y remédier. Ces automatisations servent de contrôles de sécurité détectifs ou réactifs qui vous aident à mettre en œuvre les meilleures pratiques en matière AWS de sécurité. Parmi les actions de réponse automatique, citons la modification d'un groupe VPC de sécurité, l'application de correctifs à une EC2 instance Amazon ou la rotation des informations d'identification.

chiffrement côté serveur

Chiffrement des données à destination, par le Service AWS qui les reçoit.

Politique de contrôle des services (SCP)

Politique qui propose un contrôle centralisé des autorisations pour tous les comptes d'une organisation dans AWS Organizations. SCPsdéfinissez des barrières de protection ou des limites

aux actions qu'un administrateur peut déléguer à des utilisateurs ou à des rôles. Vous pouvez utiliser SCPs comme listes d'autorisation ou de refus, pour indiquer les services ou les actions autorisés ou interdits. Pour plus d'informations, veuillez consulter la rubrique Stratégies de contrôle de service dans la AWS Organizations documentation.

point de terminaison du service

Le URL point d'entrée d'un Service AWS. Pour vous connecter par programmation au service cible, vous pouvez utiliser un point de terminaison. Pour plus d'informations, veuillez consulter la rubrique Service AWS endpoints dans Références générales AWS.

contrat de niveau de service () SLA

Accord qui précise ce qu'une équipe informatique promet de fournir à ses clients, comme le temps de disponibilité et les performances des services.

indicateur de niveau de service () SLI

Mesure d'un aspect des performances d'un service, tel que son taux d'erreur, sa disponibilité ou son débit.

objectif de niveau de service () SLO

Mesure cible qui représente l'état d'un service, tel que mesuré par un indicateur de niveau de service.

modèle de responsabilité partagée

Un modèle décrivant la responsabilité que vous partagez en matière AWS de sécurité et de conformité dans le cloud. AWS est responsable de la sécurité du cloud, alors que vous êtes responsable de la sécurité dans le cloud. Pour de plus amples informations, veuillez consulter Modèle de responsabilité partagée.

SIEM

Voir les informations de sécurité et le système de gestion des événements.

point de défaillance unique (SPOF)

Défaillance d'un seul composant critique d'une application susceptible de perturber le système.

SLA

Voir contrat de niveau de service.

SLI

Voir l'indicateur de niveau de service.

SLO

Voir l'objectif de niveau de service.

split-and-seed modèle

Modèle permettant de mettre à l'échelle et d'accélérer les projets de modernisation. Au fur et à mesure que les nouvelles fonctionnalités et les nouvelles versions de produits sont définies, l'équipe principale se divise pour créer des équipes de produit. Cela permet de mettre à l'échelle les capacités et les services de votre organisation, d'améliorer la productivité des développeurs et de favoriser une innovation rapide. Pour plus d'informations, veuillez consulter Phased approach to modernizing applications dans le. AWS Cloud

SPOF

Voir point de défaillance unique.

schéma d'étoiles

Structure organisationnelle de base de données qui utilise une grande table de faits pour stocker les données transactionnelles ou mesurées et utilise une ou plusieurs tables dimensionnelles plus petites pour stocker les attributs des données. Cette structure est conçue pour être utilisée dans un entrepôt de données ou à des fins de business intelligence.

modèle de figuier étrangleur

Approche de modernisation des systèmes monolithiques en réécrivant et en remplaçant progressivement les fonctionnalités du système jusqu'à ce que le système hérité puisse être mis hors service. Ce modèle utilise l'analogie d'un figuier de vigne qui se développe dans un arbre existant et qui finit par supplanter son hôte. Le schéma a été présenté par Martin Fowler comme un moyen de gérer les risques lors de la réécriture de systèmes monolithiques. Pour obtenir un exemple d'application de ce modèle, veuillez consulter Modernizing legacy MicrosoftASP.NET(ASMX) des services Web de manière incrémentielle à l'aide de conteneurs et d'Amazon API Gateway.

sous-réseau

Plage d'adresses IP dans votreVPC. Un sous-réseau doit se trouver dans une seule zone de disponibilité.

contrôle de supervision et acquisition de données (SCADA)

Dans le secteur manufacturier, un système qui utilise du matériel et des logiciels pour surveiller les actifs physiques et les opérations de production.

chiffrement symétrique

Algorithme de chiffrement qui utilise la même clé pour chiffrer et déchiffrer les données.

tests synthétiques

Tester un système de manière à simuler les interactions des utilisateurs afin de détecter les problèmes potentiels ou de surveiller les performances. Vous pouvez utiliser Amazon CloudWatch Synthetics pour créer ces tests.

T

balises

Des paires clé-valeur qui jouent le rôle de métadonnées pour organiser vos AWS ressources. Les balises peuvent vous aider à gérer, identifier, organiser, rechercher et filtrer des ressources. Pour plus d'informations, veuillez consulter la rubrique Balisage de vos AWS ressources.

variable cible

La valeur que vous essayez de prédire dans le cadre du ML supervisé. Elle est également qualifiée de variable de résultat. Par exemple, dans un environnement de fabrication, la variable cible peut être un défaut du produit.

liste de tâches

Outil utilisé pour suivre les progrès dans un runbook. Liste de tâches qui contient une vue d'ensemble du runbook et une liste des tâches générales à effectuer. Pour chaque tâche générale, elle inclut le temps estimé nécessaire, le propriétaire et l'avancement.

environnement de test

Voir environnement.

entraînement

Pour fournir des données à partir desquelles votre modèle de ML peut apprendre. Les données d'entraînement doivent contenir la bonne réponse. L'algorithme d'apprentissage identifie des modèles dans les données d'entraînement, qui mettent en correspondance les attributs des données d'entrée avec la cible (la réponse que vous souhaitez prédire). Il fournit un modèle de ML qui capture ces modèles. Vous pouvez alors utiliser le modèle de ML pour obtenir des prédictions sur de nouvelles données pour lesquelles vous ne connaissez pas la cible.

passerelle de transit

Hub de transit de réseau que vous pouvez utiliser pour relier votre réseau VPCs et vos réseaux sur site. Pour plus d'informations, veuillez consulter la rubrique Qu'est-ce qu'une passerelle de transit ? dans la AWS Transit Gateway documentation.

flux de travail basé sur jonction

Approche selon laquelle les développeurs génèrent et testent des fonctionnalités localement dans une branche de fonctionnalités, puis fusionnent ces modifications dans la branche principale. La branche principale est ensuite intégrée aux environnements de développement, de préproduction et de production, de manière séquentielle.

accès sécurisé

Octroi d'autorisations à un service que vous spécifiez pour effectuer des tâches au sein de votre organisation dans AWS Organizations et dans ses comptes en votre nom. Le service de confiance crée un rôle lié au service dans chaque compte, lorsque ce rôle est nécessaire, pour effectuer des tâches de gestion à votre place. Pour plus d'informations, consultez la section Utilisation AWS Organizations avec d'autres AWS services dans la AWS Organizations documentation.

réglage

Pour modifier certains aspects de votre processus d'entraînement afin d'améliorer la précision du modèle de ML. Par exemple, vous pouvez entraîner le modèle de ML en générant un ensemble d'étiquetage, en ajoutant des étiquettes, puis en répétant ces étapes plusieurs fois avec différents paramètres pour optimiser le modèle.

équipe de deux pizzas

Une petite DevOps équipe que vous pouvez nourrir avec deux pizzas. Une équipe de deux pizzas garantit les meilleures opportunités de collaboration possible dans le développement de logiciels.

U

incertitude

Un concept qui fait référence à des informations imprécises, incomplètes ou inconnues susceptibles de compromettre la fiabilité des modèles de ML prédictifs. Il existe deux types d'incertitude : l'incertitude épistémique est causée par des données limitées et incomplètes, alors que l'incertitude aléatoire est causée par le bruit et le caractère aléatoire inhérents aux données.

Pour plus d'informations, veuillez consulter le guide Quantifying uncertainty in deep learning systems.

tasks indifférenciés

Également connu sous le nom de « levage de charges lourdes », ce travail est nécessaire pour créer et exploiter une application, mais qui n'apporte pas de valeur directe à l'utilisateur final ni d'avantage concurrentiel. Les exemples de tâches indifférenciées incluent l'approvisionnement, la maintenance et la planification des capacités.

environnements supérieurs

Voir environnement.

V

mise à vide

Opération de maintenance de base de données qui implique un nettoyage après des mises à jour incrémentielles afin de récupérer de l'espace de stockage et d'améliorer les performances.

contrôle de version

Processus et outils permettant de suivre les modifications, telles que les modifications apportées au code source dans un référentiel.

VPCpeering

Connexion entre deux VPCs qui vous permet d'acheminer le trafic à l'aide d'adresses IP privées. Pour plus d'informations, veuillez consulter la rubrique Qu'est-ce que l'VPCappairage ? dans la VPC documentation Amazon.

vulnérabilités

Défaut logiciel ou matériel qui compromet la sécurité du système.

W

cache actif

Cache tampon qui contient les données actuelles et pertinentes fréquemment consultées. L'instance de base de données peut lire à partir du cache tampon, ce qui est plus rapide que la lecture à partir de la mémoire principale ou du disque.

données chaudes

Données rarement consultées. Lorsque vous interrogez ce type de données, des requêtes modérément lentes sont généralement acceptables.

fonction de fenêtrage

SQLFonction qui effectue un calcul sur un groupe de lignes liées d'une manière ou d'une autre à l'enregistrement en cours. Les fonctions de fenêtre sont utiles pour traiter des tâches, telles que le calcul d'une moyenne mobile ou l'accès à la valeur des lignes en fonction de la position relative de la ligne en cours.

charge de travail

Ensemble de ressources et de code qui fournit une valeur métier, par exemple une application destinée au client ou un processus de backend.

flux de travail

Groupes fonctionnels d'un projet de migration chargés d'un ensemble de tâches spécifique. Chaque flux de travail est indépendant, mais prend en charge les autres flux de travail du projet. Par exemple, le flux de travail du portefeuille est chargé de prioriser les applications, de planifier les vagues et de collecter les métadonnées de migration. Le flux de travail du portefeuille fournit ces actifs au flux de travail de migration, qui migre ensuite les serveurs et les applications.

WORM

Voir écrire une fois, lire plusieurs.

WQF

Voir le cadre AWS de qualification de la charge de travail.

écrire une fois, lire plusieurs (WORM)

Modèle de stockage qui écrit les données une seule fois et empêche leur suppression ou leur modification. Les utilisateurs autorisés peuvent lire les données autant de fois que nécessaire, mais ils ne peuvent pas les modifier. Cette infrastructure de stockage de données est considérée comme immuable.

Z

exploit zéro

Une attaque, généralement un logiciel malveillant, qui tire parti d'une vulnérabilité de type « jour zéro ».

vulnérabilité de type « jour zéro »

Une faille ou une vulnérabilité non atténuée dans un système de production. Les acteurs malveillants peuvent utiliser ce type de vulnérabilité pour attaquer le système. Les développeurs prennent souvent conscience de la vulnérabilité à la suite de l'attaque.

application zombie

Application dont l'utilisation moyenne de CPU la mémoire est inférieure à 5 %. Dans un projet de migration, il est courant de retirer ces applications.

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.