

L'INTELLIGENCE ARTIFICIELLE GÉNÉRATIVE ET LE CLOUD AWS : VERS UNE NOUVELLE ÈRE TECHNOLOGIQUE

WILLIAM LENOIR

LIVRE BLANC - DÉCEMBRE 2023

 **Ippon**

SOMMAIRE

L'AUTEUR	03	
INTRODUCTION	04	
PARTIE 1	05	QUEL EST LE LIEN ENTRE L'INTELLIGENCE ARTIFICIELLE GÉNÉRATIVE ET LE CLOUD ?
		Puissance de calcul et stockage
		Flexibilité
		Services et intégrations
		Vers une démocratisation de l'IA
PARTIE 2	10	QU'EST-CE QU'UN FOUNDATION MODEL ?
		Les réseaux de neurones
		Les Transformers
		L'émergence et la diversité des Foundation Models
PARTIE 3	16	AWS BEDROCK DÉMYSTIFIÉ
		Fine-tuning avec ses propres données
		RAG : Retrieval Augmented Generation avec les Knowledge Bases
		Un agent capable de raisonner et d'agir
CONCLUSION	36	
ANNEXES	38	UML : Diagramme de classes du service AWS Bedrock et de ses intégrations
		Insurance Claims Automation API



WILLIAM LENOIR

En tant que Manager et Architecte de Solutions chez Ippon Technologies à Paris, ma passion pour l'Intelligence Artificielle et le Cloud AWS est le moteur principal de mon engagement et de ma curiosité dans le domaine technologique.

J'accompagne mes clients dans la conception et la mise en œuvre d'infrastructures Cloud AWS, ainsi que dans l'intégration de technologies clés telles que Kafka et la suite ELK.

N'hésitez pas à me contacter via [LinkedIn](#) ou [Twitter](#), et à explorer le [blog d'Ippon](#), où vous pourrez découvrir quelques-uns de mes articles.

INTRODUCTION

L'**intelligence artificielle générative**, c'est-à-dire l'utilisation de **modèles de deep learning** pour générer du texte, des images ou des vidéos, semble être au cœur de l'innovation technologique et des enjeux des systèmes d'entreprise de demain.

Dans ce livre, nous verrons comment l'intégration de l'intelligence artificielle générative avec le **Cloud AWS** ouvre de nouvelles perspectives d'innovation, et comment les entreprises peuvent tirer parti de ces applications potentielles.

Afin de répondre à cette vaste question, nous commencerons par mettre en évidence l'interdépendance entre l'IA et le Cloud computing, deux domaines bien que distincts, intimement liés.

Ensuite, nous expliquerons le fonctionnement des **Foundation Models**.

Pour terminer, nous examinerons les fonctionnalités offertes par le service **AWS Bedrock**, en mettant en lumière les perspectives d'automatisation rendues possibles par l'intelligence artificielle générative.

NB :

— *Un diagramme UML est fourni en annexe, n'hésitez pas à vous y référer pour avoir une représentation visuelle des différents concepts présentés dans la troisième partie.*

— *J'ai choisi de conserver certains termes techniques en anglais afin de permettre au lecteur de facilement comparer les informations de ce livre avec la documentation officielle d'AWS.*

QUEL EST LE LIEN
ENTRE L'INTELLIGENCE
ARTIFICIELLE GÉNÉRATIVE
ET LE CLOUD ?

01



PUISSANCE DE CALCUL ET STOCKAGE

LORSQU'IL S'AGIT D'INTELLIGENCE ARTIFICIELLE GÉNÉRATIVE, LE CLOUD EST SOUVENT ASSOCIÉ. DANS CETTE SECTION, NOUS EXPLORERONS EN DÉTAIL LEUR INTERDÉPENDANCE.

En premier lieu, il est important de souligner que l'intelligence artificielle générative est un domaine qui nécessite une **puissance de calcul accrue** (plusieurs milliers de GPUs sont nécessaires pendant des dizaines de jours pour l'entraînement des modèles selon certaines fuites d'information non vérifiées).

Qu'il s'agisse d'entraîner des modèles ou bien de les exécuter, de nombreux serveurs doivent travailler de concert afin de pouvoir obtenir des résultats exploitables.

D'autre part, l'intelligence artificielle générative ne génère rien sans données préalables. Ainsi, à la puissance de calcul, il faut ajouter une grande **capacité de stockage** pour emmagasiner les jeux de données nécessaires à l'entraînement des modèles.

Une telle combinaison constituerait un véritable challenge à mettre en œuvre au sein d'une infrastructure dédiée sur site, alors que les fournisseurs Cloud offrent des solutions spécifiquement conçues pour répondre à ces exigences.



02

FLEXIBILITÉ

UN AUTRE AVANTAGE DU CLOUD EST SA CAPACITÉ DE MISE À L'ÉCHELLE.

Que vous débutiez avec un simple POC ou bien que vous déployiez une application complexe en production, le Cloud vous accompagnera dans votre évolution et votre infrastructure pourra s'adapter à la demande.

Ainsi, il est dès lors possible de considérer un projet d'intelligence artificielle générative de manière agile avec une **flexibilité** accrue.

N'OUBLIONS PAS QU'UNE INTELLIGENCE ARTIFICIELLE GÉNÉRATIVE SEULE N'EST PAS EXPLOITABLE.

Pour créer de la valeur, vous aurez besoin de l'intégrer à vos applications, à votre workflow et à votre logique métier.

Le Cloud offre un large éventail de **services** qui facilitent les intégrations et le développement d'un tel **écosystème**. Vous pouvez, par exemple, créer votre data lake sur AWS S3 à partir de plusieurs autres sources de données (comme AWS DynamoDB ou encore AWS Kinesis). Vous pouvez ensuite exploiter ce lake avec des services ETL (comme AWS Glue ou AWS EMR) pour affiner, corrélérer et transformer vos données. Finalement, un service comme AWS QuickSight vous permettra de faire de l'analytics sur l'ensemble de vos données désormais raffinées.



VERS UNE DÉMOCRATISATION DE L'IA

BIEN QUE L'IA OCCUPE UNE PLACE CROISSANTE DANS NOS USAGES NUMÉRIQUES, SON ADOPTION ET SON IMPLÉMENTATION PAR LES ENTREPRISES DONT CE N'EST PAS LE CŒUR DE MÉTIER SEMBLE ÊTRE UN DÉFI COMPLEXE.

En effet, le secteur nécessite une expertise poussée ainsi que d'importantes ressources.

Toutefois, les fournisseurs Cloud tendent à promouvoir la **Generative AI as a Service**.

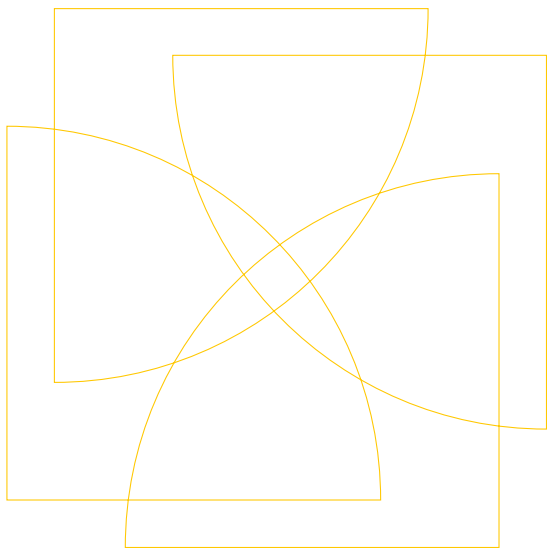
L'objectif est de pouvoir bénéficier d'une simple **API** pour interagir avec des services serverless et managés d'intelligence artificielle générative.

En appelant cette API, il est dès lors possible de converser avec une IA hébergée sur le Cloud, et ce, via quelques lignes de

code dans n'importe quel langage de programmation.

Les entreprises n'ont donc pas nécessairement besoin de développer des compétences avancées en IA, et peuvent s'appuyer sur leurs compétences en développement existantes.

En quelques mots, l'intelligence artificielle générative n'est plus réservée aux géants du web et aux laboratoires de data science. Le Cloud permet de rendre ce domaine **accessible** à tout type d'entreprise ou d'organisation, et ce, quels que soient leur taille et le niveau d'expertise de leurs équipes.



QU'EST-CE QU'UN
**FOUNDATION
MODEL ?**

02

The image features a solid blue background. In the lower half, there are several thin, gold-colored lines that form a complex, abstract geometric pattern, resembling a stylized roof or a network of connections. A large, white, sans-serif number '02' is prominently displayed in the lower center, partially overlapping the gold lines. To the right of the '02', there is a white horizontal bar that is partially cut off by the edge of the frame.



Avant de vous présenter le service Bedrock, qui est la réponse d’AWS pour accompagner les entreprises dans leurs projets d’intelligence artificielle générative, il est essentiel de clarifier certains concepts permettant de mieux comprendre le fonctionnement des Foundation Models, pierre angulaire de ce service.

POUR EXPLIQUER CE QU'EST UN FOUNDATION MODEL (FM), NOUS ALLONS PARTIR DES RÉSEAUX DE NEURONES.

Il s'agit d'une architecture développée dans le domaine du Machine Learning afin de permettre aux machines de réaliser des tâches complexes, nécessitant une intelligence, tout en conservant une capacité **d'adaptation** et **d'apprentissage**. Pour atteindre cet objectif, le **cerveau** humain a été pris comme *modèle*.

Effectivement, notre cerveau est composé d'environ *86 milliards* de neurones (source : [Wikipédia](#)) et tout au long de notre vie, des liaisons entre ces cellules (les synapses) se forment, se réduisent ou se renforcent. L'estimation du nombre de synapses dans un cerveau humain atteint des ordres de grandeur se chiffrant en *centaines de trillions* !

De même, un réseau de neurones en Machine Learning permet, via un apprentissage, de renforcer certaines liaisons au profit d'autres, émulant ainsi la *plasticité cérébrale* que l'on retrouve dans le vivant.

En Machine Learning, l'apprentissage peut être classé en plusieurs catégories, dont les plus communes sont le mode supervisé et le mode non supervisé.

Dans l'apprentissage **supervisé**, on fournit au modèle des exemples d'entrées avec les sorties correspondantes. Le réseau va donc comparer la sortie qu'il produit avec la sortie attendue. Grâce à l'analyse de son erreur, il est capable d'adapter ses paramètres internes (ses liaisons de neurones) afin de réduire peu à peu sa marge d'erreur.

Dans l'apprentissage **non supervisé**, on fournit au modèle uniquement des exemples d'entrées, sans sortie prédéfinie. À l'inverse du mode supervisé, il n'existe donc pas, pour le modèle, de "*réponse correcte*". Le modèle cherche plutôt à parcourir les données afin d'en extraire des structures, des motifs et des relations.

Les **Deep Neural Networks** sont des réseaux de neurones particuliers qui disposent de plusieurs couches intermédiaires, leur permettant de développer des modèles bien plus complexes.

À partir des Deep Neural Networks, il est possible de créer les Foundation Models.

Les **Transformers** sont une architecture spécifique pour les réseaux de neurones, spécialement conçue pour le traitement du langage naturel (NLP), et sont largement utilisés dans les modèles d'intelligence artificielle générative.

Il est important de comprendre que rendre un texte intelligible pour une machine constitue un véritable défi.

Avant les Transformers, les modèles de type **Recurrent Neural Network** (RNN) étaient utilisés.

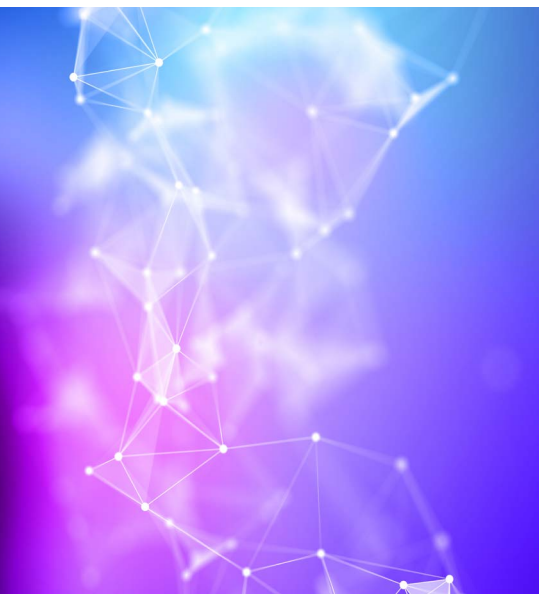
La méthodologie RNN repose sur le traitement **séquentiel** mot à mot.

Il en résulte deux grandes difficultés techniques et fonctionnelles :

- L'analyse étant séquentielle, il n'est pas possible de paralléliser les traitements lors de l'entraînement des modèles. Ce qui est fort dommage lorsque le Cloud nous permet d'avoir accès à plusieurs machines bénéficiant de **GPUs** (cartes graphiques spécialisées dans le traitement des calculs en parallèle). De plus, cette limitation nous empêche d'entraîner le modèle sur un volume important de données.
- La compréhension du texte diminue en fonction de sa longueur, puisque le modèle a du mal à conserver des références au contexte provenant du début de sa lecture lorsqu'il arrive à la fin de celle-ci.

Les Transformers, à l'opposé, permettent :

- Un traitement **parallélisé** des données. Il est dès lors possible d'entraîner son modèle sur plusieurs téraoctets de données.
- Un plus grand niveau **d'attention** et donc une meilleure **compréhension** du contexte.



L'ÉMERGENCE ET LA DIVERSITÉ DES FOUNDATION MODELS

FORT DES SOLUTIONS
TECHNIQUES APPORTÉES
PAR LES TRANSFORMERS,
LES FOUNDATION MODELS
ONT PU FAIRE LEUR
APPARITION.

Il s'agit de modèles entraînés sur une quantité colossale de données non labellisées, comprenant une grande partie de l'internet public (Wikipédia, Reddit ainsi que d'autres sources, le contenu exhaustif n'ayant pas été révélé publiquement).

Grâce à cet entraînement, les Foundation Models développent une **compréhension empirique du langage** ainsi qu'une **connaissance encyclopédique**.

Les Foundation Models, sont des modèles dit **généralistes**. C'est-à-dire, qu'à l'inverse des modèles **spécialisés**, ils sont capables d'effectuer une grande variété de tâches avec des performances correctes.

Les cas d'utilisation des Foundation Models sont très variés et se déclinent autour de la génération de textes et d'images.



Ils simplifient la production de contenus pour le web (articles de blog, sites internet et publications sur les réseaux sociaux).

Ils rendent également possible la création d'assistants virtuels, capables d'émuler une conversation avec un utilisateur, de répondre à ses questions et de demander des informations supplémentaires pour atteindre des objectifs spécifiques.

Dans le milieu professionnel, les capacités de synthèse et de recherche des Foundation Models facilitent l'accès à l'information.

En termes d'applications concrètes, nous pouvons notamment citer :

- La création d'expériences utilisateurs personnalisées et fluides, facilitant par exemple les opérations d'achat sur les sites d'e-commerce.
- La traduction de textes avec un rendu naturel.
- L'assistance dans les processus de création artistique, incluant entre autres la production de logos pour le marketing ou la génération d'images pour les studios de jeux vidéo.
- L'analyse de données complexes, telles que des textes de loi ou de la documentation technique, afin d'assister les professionnels dans leurs tâches quotidiennes.
- L'étude de données financières pour extraire des tendances,

détecter des fraudes et établir des prévisions, des outils précieux dans les secteurs de la finance, de la banque et de l'assurance.

Parmi les Foundation Models, nous pouvons citer ChatGPT, qui a gagné une importante notoriété auprès du grand public. Cependant, il ne représente pas la seule solution en matière d'intelligence artificielle générative.

Effectivement, de nombreuses entreprises ont développé en parallèle **leurs propres Foundation Models** (voir ci-après).

Il en résulte, non pas un unique modèle, mais *des* modèles. Chacun présentant des avantages et des inconvénients.

Par exemple, le nombre de paramètres que comporte un Foundation Model peut varier de quelques millions à plusieurs milliards.

Selon la taille des modèles, l'infrastructure de calcul requise varie en complexité, influençant directement la latence de génération des résultats ainsi que les coûts opérationnels.

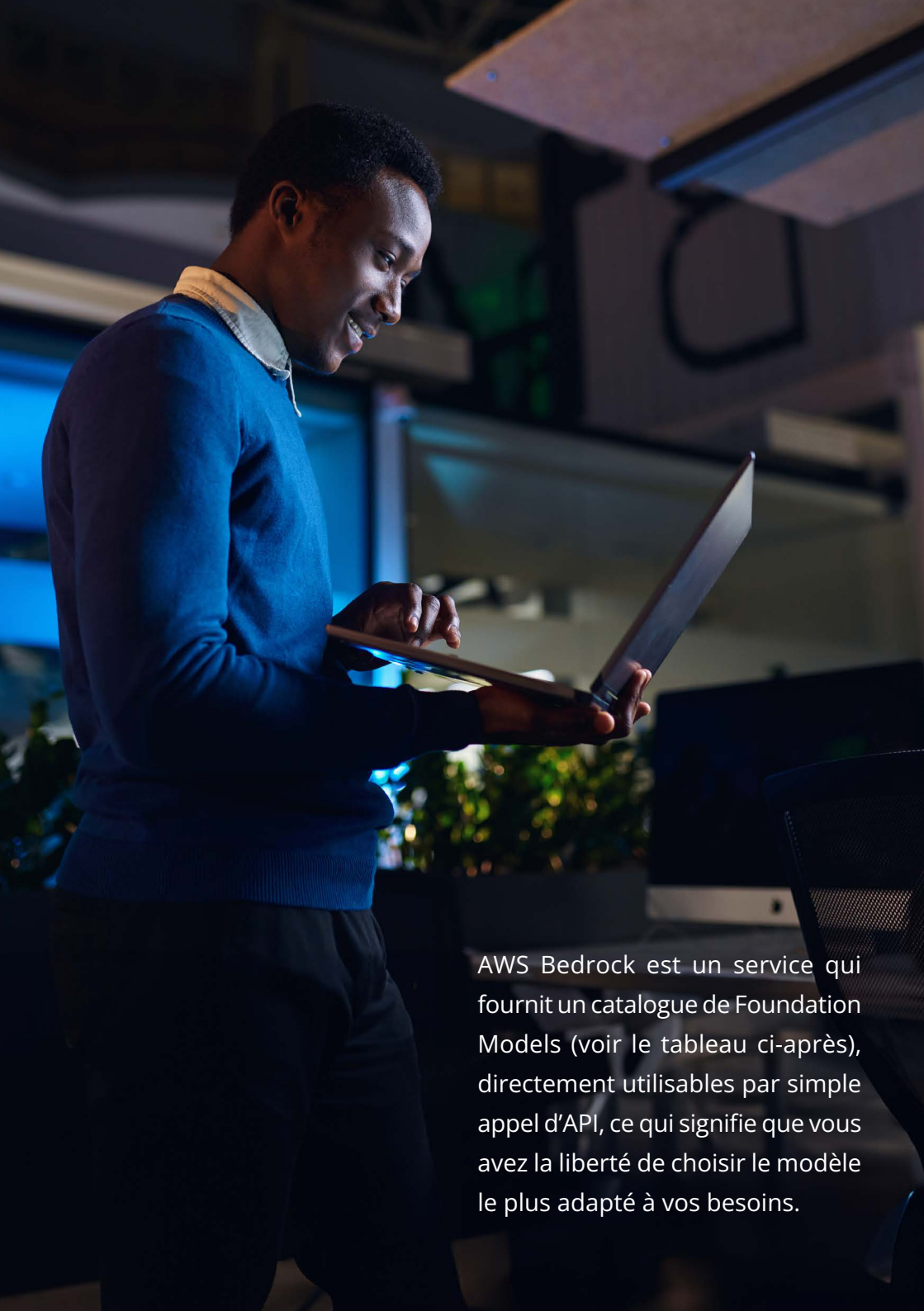
Ces considérations agissent comme des catalyseurs d'innovation, ouvrant la voie à l'émergence de nouveaux Foundation Models à tout moment dans les années à venir.

AWS BEDROCK


DÉMYSTIFIÉ

03 —

The background features several thin, gold-colored lines that intersect to form a series of overlapping triangles and polygons. A thick white horizontal line is positioned to the right of the large number '03', suggesting a page number or chapter indicator.



AWS Bedrock est un service qui fournit un catalogue de Foundation Models (voir le tableau ci-après), directement utilisables par simple appel d'API, ce qui signifie que vous avez la liberté de choisir le modèle le plus adapté à vos besoins.

Provider	Caractéristiques	 Génération de texte	 Génération d'image	 Embedding
 AI21labs	<ul style="list-style-type: none"> — Prise en charge de 7 langues — Applicable à plusieurs secteurs d'activité 	<ul style="list-style-type: none"> — Jurassic-2 Mid v1 (8 191 tokens) — Jurassic-2 Ultra v1 (8 191 tokens) 		
 amazon	<ul style="list-style-type: none"> — Large gamme d'applications — Issu de 20 ans de recherche en Machine Learning — Facilement personnalisable — Multilingue 	<ul style="list-style-type: none"> — Titan Text G1 - Lite v1 (4 000 tokens) — Titan Text G1 - Express v1 (8 000 tokens) 	<ul style="list-style-type: none"> — Titan Image Generator G1 v1 (77 tokens) 	<ul style="list-style-type: none"> — Titan Embeddings G1 - Text v1.2 (8 000 tokens) — Titan Multimodal Embeddings G1 v1 (128 tokens)
 ANTHROPIC	<ul style="list-style-type: none"> — La meilleure fenêtre de contexte du marché — Recherche sécurisée grâce à l'IA Constitutionnelle — Multilingue 	<ul style="list-style-type: none"> — Claude v1.3 (100 000 tokens) — Claude v2.1 (200 000 tokens) — Claude Instant v1.2 (100 000 tokens) 		
 cohere	<ul style="list-style-type: none"> — Optimisé pour les applications professionnelles — Uniquement en anglais 	<ul style="list-style-type: none"> — Command Light v14.7 (4 000 tokens) — Command v14.7 (4 000 tokens) 		<ul style="list-style-type: none"> — Embed English v3 (512 tokens) — Embed Multilingual v3 (512 tokens)
 Meta	<ul style="list-style-type: none"> — Optimisé pour le dialogue — Enrichi par plus d'un million d'annotations humaines — Uniquement en anglais 	<ul style="list-style-type: none"> — Llama 2 13B v1 (4 096 tokens) — Llama 2 70B v1 (4 096 tokens) — Llama 2 Chat 13B v1 (4 096 tokens) — Llama 2 Chat 70B v1 (4 096 tokens) 		
 stability.ai	<ul style="list-style-type: none"> — Capacité de résolution d'image de 1024×1024 — Génération d'images photoréalistes — Capable de créer des compositions complexes à partir d'instructions en langage naturel — Uniquement en anglais 		<ul style="list-style-type: none"> — SDXL v0.8 (77 tokens) — SDXL v1.0 (77 tokens) 	

Selon le modèle, une ou plusieurs langues sont supportées et la taille de la **fenêtre de contexte** varie.

Sa longueur maximale, exprimée en nombre de tokens, définit la quantité maximale de texte que le modèle peut garder en mémoire pour générer sa réponse.

La définition d'un token dépend du modèle ; cependant, il représente toujours la plus petite unité de texte pouvant être traitée. En général, il s'agit d'une partie de mot.

Une fenêtre de contexte plus grande permet au modèle d'analyser un contexte plus étendu et donc de fournir des réponses plus pertinentes.

Par exemple, le modèle Claude 2 d'Anthropic, avec ses 200 000 tokens,

est capable de traiter un contexte d'environ 150 000 mots, ce qui équivaut à approximativement 500 pages.

Concernant le catalogue de modèles, pour la génération de texte, nous avons **Amazon Titan**, un Foundation Model développé exclusivement par Amazon, ainsi que Claude 2 d'Anthropic, Llama 2 de Meta, Jurassic 2 d'AI21 Labs et Command de Cohere.

En ce qui concerne la génération d'images, le service propose Stable Diffusion de Stability AI ainsi qu'Amazon Titan Image Generator.

Dans cette dernière section, je vais vous présenter les fonctionnalités offertes par AWS Bedrock comme le **Fine-tuning**, l'architecture **RAG** avec les **Knowledge Bases** ainsi que l'automatisation des tâches avec les **Agents**.



FINE-TUNING AVEC SES PROPRES DONNÉES

Même si les Foundation Models mis à disposition par Bedrock permettent de répondre raisonnablement à des questions générales, ils atteindront leur limites si vous souhaitez orienter leurs réponses vers des domaines spécifiques à votre activité.

L'IDÉAL SERAIT DONC D'AVOIR
VOTRE PROPRE MODÈLE,
TAILLÉ SPÉCIFIQUEMENT
SELON VOS BESOINS.

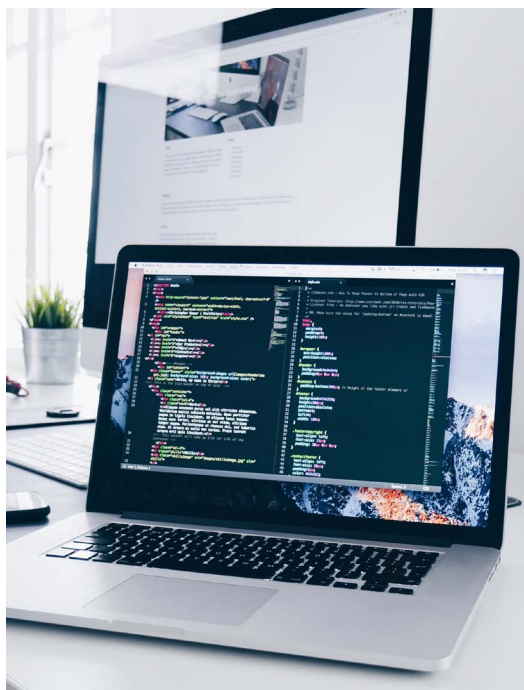
Cependant, développer son propre modèle à partir de zéro est un travail extrêmement complexe. C'est pourquoi, il est recommandé de partir d'un Foundation Model pré-entraîné et de l'utiliser comme base pour mettre en œuvre des techniques d'ajustement qui se traduisent par un nouvel entraînement du modèle, mais cette fois à partir de jeux de données spécifiques à un domaine ou à une tâche.

Le processus de développement est considérablement écourté, puisqu'avec un ensemble de données judicieusement sélectionnées, il est possible d'obtenir rapidement des performances satisfaisantes.

Bedrock permet la personnalisation des Foundation Models en utilisant vos données privées stockées dans un bucket S3 qui vous appartient.

Le **Fine-tuning**, qui utilise des données labellisées, spécialise le modèle pour des tâches spécifiques.

Tandis que le **Continued Pre-training**, actuellement en phase de preview, permet d'exploiter des données non labellisées pour que le modèle apprenne de nouveaux domaines de connaissances.



Le jeu de données est composé d'un fichier contenant des lignes au format JSON, dont la structure est détaillée ci-dessous :

Méthode	Format
Fine-tuning	<pre>{ "prompt": "<exemple de prompt>", "completion" : "<réponse attendue>" }</pre>
Continued Pre-training	<pre>{ "input": "<exemple d'input>" }</pre>

AWS indique que pour le fine-tuning 100 à 500 lignes JSON sont suffisantes pour obtenir des résultats satisfaisants. Il est important de noter qu'il est recommandé de privilégier la qualité des données plutôt que leur quantité.

Il convient également de souligner que AWS Bedrock est conçu pour assurer la **sécurité** et le **contrôle** de vos données.

En premier lieu, AWS garantit que vos données **ne sont pas utilisées pour améliorer** les modèles d'Amazon ou de tout autre fournisseur.

De plus, vos données restent dans la région que vous sélectionnez lors de l'utilisation du service.

Concernant le chiffrement au repos, les modèles sont chiffrés avec le service AWS Key Management Service (KMS), soit avec la clé gérée par le service Bedrock, soit avec la **Customer Managed Key (CMK)** de votre choix.

Cette dernière option vous donne un contrôle exclusif sur la clé, vous permettant de définir sa Key Policy et d'initier sa rotation.

Enfin, le chiffrement en transit est sécurisé avec les protocoles TLS 1.2 ou 1.3.

Pour une sécurité accrue, vous pouvez utiliser **AWS Private Link** afin d'établir une connexion privée directement depuis votre VPC au service Bedrock. Ce mécanisme garantit que le trafic ne transite pas par Internet, mais reste à l'intérieur du réseau privé d'AWS.



BEDROCK PREND EN CHARGE L'ARCHITECTURE RAG (RETRIEVAL AUGMENTED GENERATION).

RAG permet de rajouter à un Large Language Model (LLM, un sous-ensemble des Foundation Models spécifiquement conçus pour comprendre et générer du texte en langage naturel), **sans avoir besoin de l'entraîner de nouveau**, des informations **contextuelles** additionnelles.

Avant de présenter le fonctionnement d'un système RAG, nous devons parler de la méthode **d'embedding**.

Par embedding, on entend l'opération suivante :

$$F(\text{chunk}) = \mathbf{v}$$

- **chunk** est une partie d'un document.
- **v** est un vecteur dans un espace à N dimensions
- **F** est la fonction d'embedding

Les coordonnées du vecteur **v** dans cet espace peuvent être représentées par une liste de N nombres : $[a_1, a_2, a_3, \dots, a_N]$.

Les **chunks** sont issus du découpage d'un document en plusieurs parties. Cette découpe doit être faite de manière intelligente afin de préserver le **sens**.

Par exemple, une approche efficace serait de découper par phrase ou bien par paragraphe un document. De cette manière, les informations contenues dans chaque chunk pourraient être examinées individuellement.

Pour la suite, nous supposons que le découpage choisi est effectué au niveau des phrases. Ainsi, chaque phrase correspond à un chunk.

L'embedding associe à chaque chunk, donc à chaque phrase, un **vecteur** dans un espace multidimensionnel.

POURQUOI CONVERTIR DU TEXTE EN UN VECTEUR ?

Car de cette manière, nous pouvons accéder à des propriétés mathématiques intéressantes comme la **mesure de la distance entre deux vecteurs**, et utiliser cette distance à des fins de comparaison.

Or, le vecteur choisi par la fonction d'embedding pour représenter une phrase est censé retranscrire le sens de celle-ci.

Ainsi, deux vecteurs sont d'autant **plus proches** que les phrases à l'origine de leurs créations sont **sémantiquement similaires**.

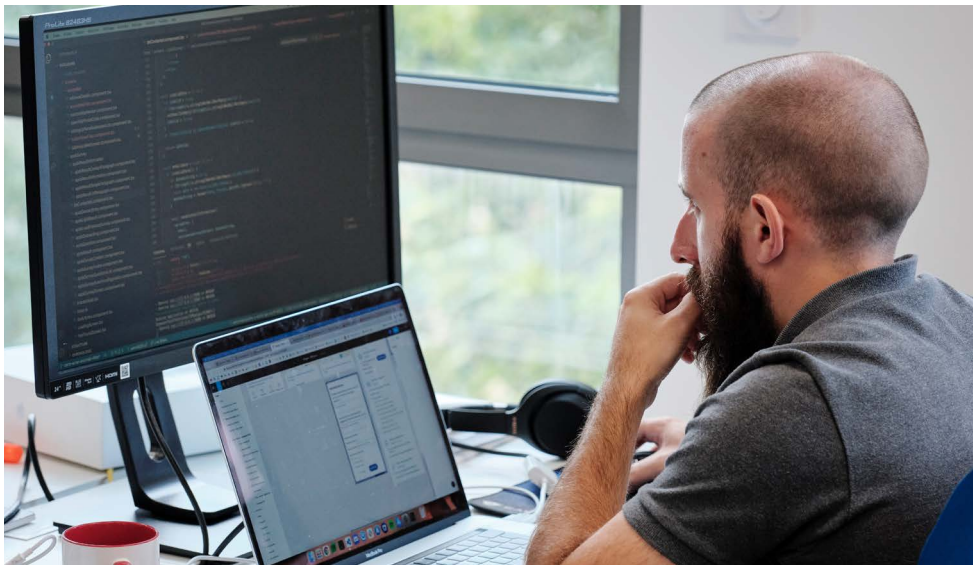
Cette fonction d'embedding capable de saisir le sens d'une donnée et de le retranscrire en un vecteur est la résultante de calculs complexes. En pratique, ces opérations sont réalisées par un **modèle**

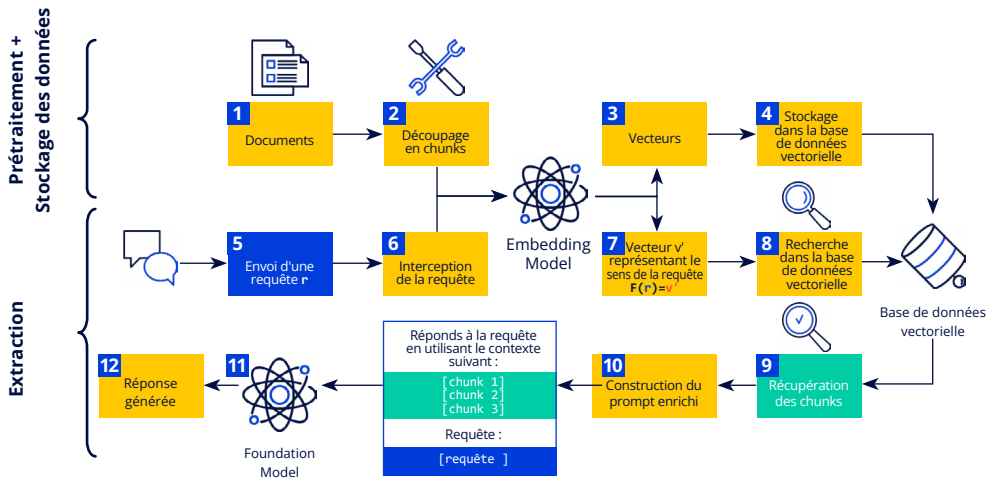
d'embedding spécifiquement entraîné pour ce type de tâche (c'est ici que la magie opère).

Ainsi, l'embedding permet de convertir l'intégralité d'un document en un ensemble de vecteurs. Ces vecteurs sont ensuite stockés dans une **base de données vectorielle** qui facilite les opérations d'indexation et de recherche sur ce type de structure (notamment la recherche des vecteurs les plus proches dans l'espace : *k-Nearest Neighbor search*).

Notons également qu'il **n'est pas possible, à partir des coordonnées d'un vecteur, de retrouver le chunk qui l'a produit**. C'est pourquoi, en plus des coordonnées, il est courant de stocker le **texte initial** ainsi que des **metadata** permettant d'identifier le document contenant ce chunk.

L'embedding n'ayant plus de secret pour vous, nous pouvons définir les différentes étapes qui constituent le système RAG et dont vous comprendrez très vite l'intérêt.





PRÉTRAITEMENT ET STOCKAGE DES DONNÉES

1. Identifier une source de données non référencée par le modèle lors de son entraînement initial (par exemple un ensemble de fichiers pdf, texte, csv appartenant à une organisation).
2. Ces documents sont découpés en chunks.
3. Grâce au modèle d'embedding, chaque chunk est transformé en un vecteur.
4. Ces vecteurs sont stockés dans une base de données vectorielle avec le chunk original et des metadata.

EXTRACTION (RETRIEVAL)

5. L'utilisateur envoie une requête r .
6. Le système RAG l'intercepte.
7. Le système utilise de nouveau le modèle d'embedding pour générer un vecteur v' représentant la valeur sémantique de la requête.
8. Les vecteurs de la base vectorielle les plus proches de v' sont recherchés.
9. Les chunks associés aux vecteurs issus des résultats de la recherche sont récupérés.
10. Le système réécrit le prompt du client en y ajoutant, avant sa requête, un bloc de contexte constitué des chunks associés aux vecteurs issus de la recherche de similarité.
11. Le Foundation Model est interrogé avec le prompt enrichi.
12. La réponse générée est envoyée à l'utilisateur.

Il n'existe pas de manière unique pour structurer un prompt, mais typiquement un **template** pourrait être :

Réponds à la requête en utilisant le contexte suivant :

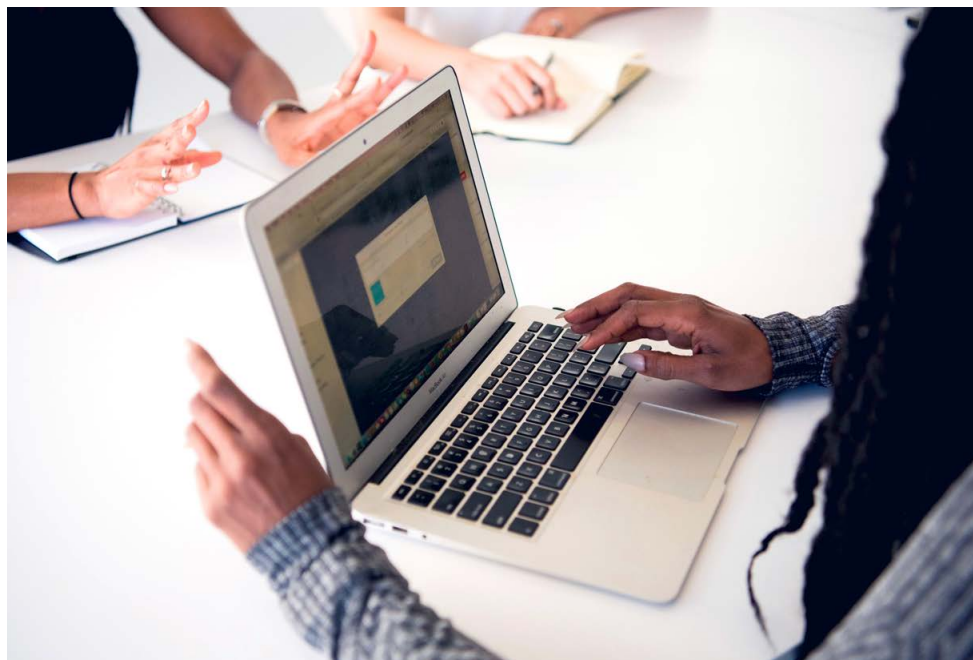
{contexte}

Requête :

{requête}

{contexte} sera remplacé par les chunks extraits de notre base et **{requête}** comporte le prompt original du client.

Ainsi, le modèle sera capable de répondre à la requête initiale en interprétant, à l'exécution, les informations contextuelles issues de notre base de données. La réponse produite sera donc beaucoup plus spécifique.



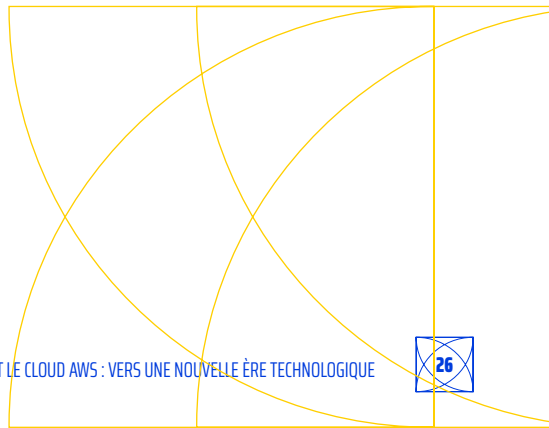
Analysons désormais les avantages et inconvénients de l'architecture RAG.

AVANTAGES

- Le modèle peut générer des réponses plus **pertinentes**, puisqu'il peut s'appuyer sur des informations privées et liées à votre contexte, réduisant ainsi les risques d'*hallucinations* (lorsque le modèle affirme des éléments erronés ou inventés).
De plus, la recherche vectorielle offre de meilleurs résultats que la recherche classique par mot-clé, car elle privilégie le sens des termes plutôt que la correspondance exacte des mots.
- Le modèle peut générer des réponses plus **maîtrisées**.
Effectivement, les Foundation Models étant développés à partir d'un vaste corpus documentaire, ils possèdent un large éventail de connaissances générales.
- Toutefois, si vous ne souhaitez pas que le modèle s'appuie sur des informations extérieures à celles que vous lui fournissez, vous pouvez utiliser une instruction telle que : "Réponds en utilisant *uniquement* le contexte fourni".
De cette manière, le modèle fera abstraction de ses connaissances préexistantes et se basera exclusivement sur votre contexte pour répondre à votre requête.
- Le modèle n'a **pas besoin d'être entraîné de nouveau** (les poids du modèle ne sont pas modifiés), ce qui vous permet de gagner du temps et de l'argent.
- **Le modèle peut être augmenté au fil de l'eau** étant donné que le workflow RAG, à condition qu'il soit automatisé, permet de bénéficier de **données récentes fréquemment mises à jour**.

INCONVÉNIENTS

- Étant donné que les chunks sont concaténés puis injectés dans le prompt, nous sommes contraints par la **limite de caractères** de ce dernier.
- La pertinence des résultats dépend fortement de la **qualité du modèle d'embedding** et de la **méthode de recherche vectorielle** utilisée.



Il est tout à fait possible de construire soi-même le workflow RAG que nous venons de présenter.

Des projets open-source tels que **LangChain** (disponible pour Python et TypeScript), permettent d'utiliser une vaste librairie de modules pour :

- Extraire le texte de différentes sources (pdf, markdown, page web, etc.) et le transformer en documents unifiés.
- Découper ces documents en chunks selon diverses méthodes.
- Effectuer des opérations sur les bases vectorielles, créer des templates de prompt et invoquer les modèles d'intelligence artificielle générative.

Toutefois, l'un des atouts majeurs d'AWS Bedrock est son haut niveau d'automatisation.

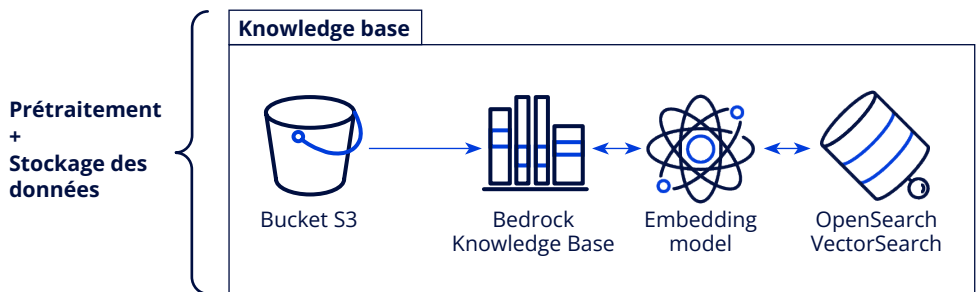
Sachez que l'intégralité du workflow que je viens de vous présenter est intégralement managé par AWS.

Vous n'avez donc ni à l'implémenter, ni à vous soucier des serveurs.

L'architecture RAG est mise en œuvre dans Bedrock via deux composants :

- La **Knowledge Base**.
- L'**Agent**, qui est capable d'utiliser une ou plusieurs Knowledge Bases.

La **Knowledge Base** permet d'automatiser la phase de Prétraitement et de stockage des données (étapes 1 à 4 du workflow).



Pour créer une Knowledge Base, vous devez spécifier :

- Un bucket S3 contenant les documents que vous souhaitez utiliser pour enrichir les connaissances du modèle ;
- Une base de données vectorielle.

Lorsque la Knowledge Base est créée, tous les fichiers préalablement déposés sur le bucket S3 seront décomposés en chunks et vectorisés. Les résultats seront stockés dans la base vectorielle que vous avez spécifiée.

AWS propose plusieurs modèles pour l'embedding, dont Titan Embeddings G1 - Text Model, ainsi que Embed English et Embed Multilingual de Cohere.

Par ailleurs, AWS Bedrock offre une intégration avec trois services de bases de données vectorielles :

- Amazon OpenSearch Service - Vector search ;
- Pinecone ;
- Redis Enterprise Cloud.

Nous verrons dans la section suivante comment l'Agent permet d'automatiser la phase d'extraction (étapes 5 à 12 du workflow).



Schématiquement, nous pouvons représenter l'Agent Bedrock comme une boîte noire qui reçoit les prompts du client et génère les réponses correspondant à ses demandes, tout en étant capable d'effectuer des opérations supplémentaires que nous allons détailler.

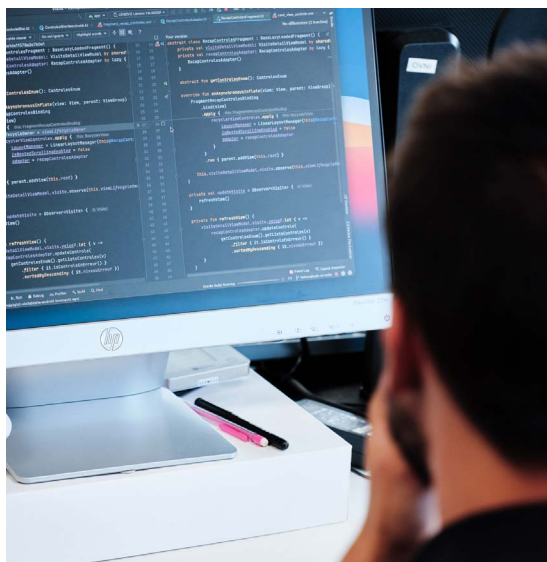
EN QUELQUES MOTS,
LES AGENTS PERMETTENT
DE DÉPASSER LES
LIMITATIONS INHÉRENTES
À L'INTELLIGENCE
ARTIFICIELLE GÉNÉRATIVE :
SON MANQUE D'EMPRISE
SUR LES SYSTÈMES
EXTÉRIEURS.

Il est possible de demander à un Foundation Model de comprendre un problème, de **raisonner** et d'établir un **plan d'actions**, mais lorsqu'il s'agit de mettre en œuvre ce plan et donc **d'agir**, le modèle se retrouve dans une impasse due à son propre isolement.

Les agents Bedrock s'affranchissent de cette limitation en bénéficiant d'une **intégration** harmonieuse entre l'agent conversationnel et **l'appel de vos propres APIs**.

L'article d'AWS intitulé "**Enable Foundation Models to Complete Tasks With Agents for Amazon Bedrock**" de Antje Barth, présente un exemple de démonstration pour mettre en avant les capacités des agents Bedrock.

Pour la suite de ce livre, nous utiliserons ce même exemple, en nous appuyant notamment sur le schéma OpenApi fourni par l'article d'AWS ("*Insurance Claims Automation API*") et que vous trouverez en annexe.



En premier lieu, je souhaite vous démontrer les capacités de raisonnement d'un Foundation Model.

Pour ce faire, j'ai construit un prompt comportant la spécification OpenAPI du blog AWS et j'ai décrit un problème simple pouvant être résolu en utilisant habilement les différentes méthodes de l'API *Insurance Claims Automation* :



Je vais te fournir une spécification OpenAPI et je vais te poser des questions :

```
{ ... openAPI ... }
```

Dis-moi comment tu ferais pour traiter le problème suivant :

"Envoie un rappel à tous les agents d'assurance qui ont des réclamations ouvertes"



La solution la plus simple serait d'appeler l'API **"/send-reminders"** une fois pour chaque réclamation ouverte, fournissant le `claimId` et les documents en attente.

Pour effectuer cette opération, il faudrait d'abord obtenir la liste des réclamations ouvertes en appelant l'API **"/claims"**. Le résultat de cette demande serait une liste de `claimIds`.

Pour chaque `claimId`, l'API **"/send-reminders"** serait ensuite appelée, fournissant le `claimId` et les documents en attente associés à cette réclamation.

Pour obtenir les documents en attente pour chaque réclamation, l'API **"/claims/{claimId}/identify-missing-documents"** serait appelée.

Je vais te fournir une spécification OpenAPI et je vais te poser des questions :

Default

```
{
  "openapi": "3.0.0",
  "info": {
    "title": "Insurance Claims Automation API",
    "version": "1.0.0",
    "description": "APIs for managing insurance claims by pulling a list of open claims, identifying outstanding paperwork for each claim, and sending reminders to policy holders."
  },
  "paths": {
    "/claims": {
      "get": {
        "summary": "Get a list of all open claims",
        "description": "Get the list of all open insurance claims. Return all the open claimIds.",
        "operationId": "getAllOpenClaims",
        "responses": {
          "200": {
            "description": "Gets the list of all open insurance claims for policy holders",
            "content": {
```

```
    }
  }
}
```

Dis-moi comment tu ferais pour traiter le problème suivant :
"Envoyer un rappel à tous les agents d'assurance qui ont des réclamations ouvertes"

La solution la plus simple serait d'appeler l'API `/send-reminders` une fois pour chaque réclamation ouverte, fournissant le `claimId` et les documents en attente.



Pour effectuer cette opération, il faudrait d'abord obtenir la liste des réclamations ouvertes en appelant l'API `/claims`. Le résultat de cette demande serait une liste de `claimIds`.

Pour chaque `claimId`, l'API `/send-reminders` serait ensuite appelée, fournissant le `claimId` et les documents en attente associés à cette réclamation.

Pour obtenir les documents en attente pour chaque réclamation, l'API `/claims/{claimId}/identify-missing-documents` serait appelée

[Add instructions](#)Default [Update inference configurations](#)

Run

On constate que le modèle a réussi à comprendre le résultat escompté : *envoyer des rappels*.

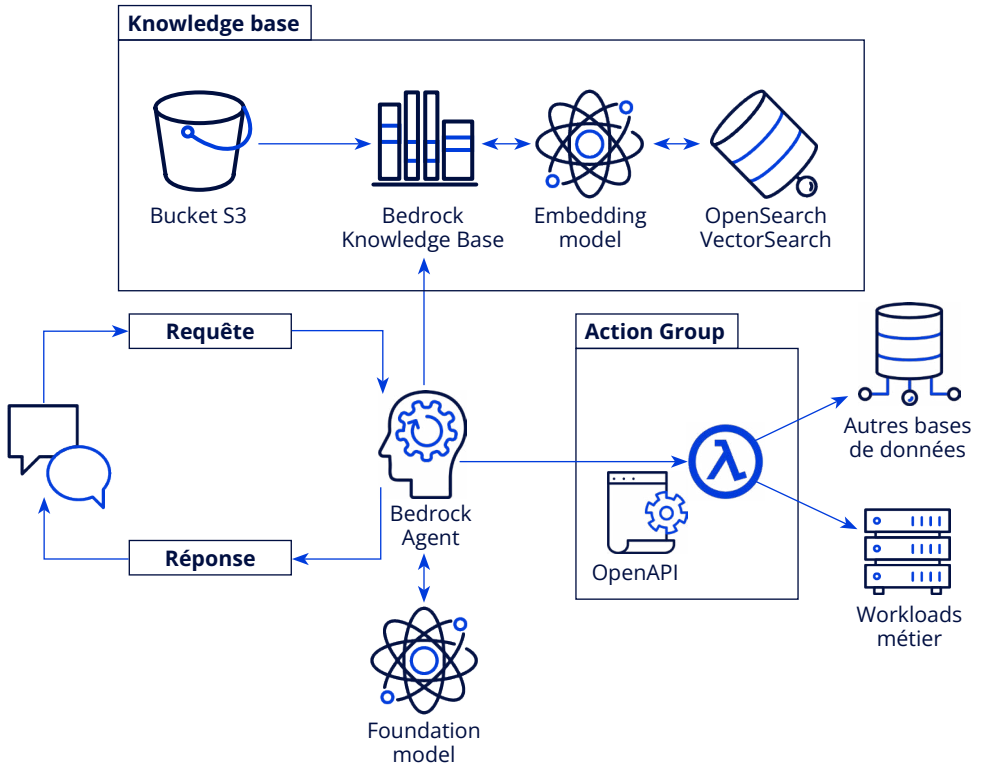
Pour atteindre cet objectif, il a su décomposer le problème en plusieurs tâches (récupérer les réclamations, puis récupérer les documents).

Il est capable de repérer les étapes qui nécessitent des boucles et a pu **comprendre** la **structure** des données et donc identifier quelles entrées et quelles sorties récupérer à partir de l'API et comment utiliser ces valeurs comme paramètres pour les appels futurs.



Notons au passage un point important : le modèle est capable d'arriver à ce niveau **sans entraînement** (fine-tuning) préalable.

Il ne manque plus qu'une chose pour rendre en pratique ce modèle opérationnel : **l'autoriser à exécuter ces appels d'API** ! Ce qui est rendu possible grâce aux **Agents**.



Pour créer un Agent dans AWS Bedrock, vous lui donnez un nom, une description et des instructions de base en langage naturel, comme son rôle et le style conversationnel à adopter. Ensuite, vous sélectionnez le Foundation Model devant être utilisé.

Vous pouvez également ajouter à votre agent une ou plusieurs **Knowledge Bases** qu'il sera susceptible d'utiliser en fonction des demandes des utilisateurs.

La connexion avec les Knowledge Bases permet d'automatiser la phase d'extraction (Retrieval) du workflow RAG.

Puisque l'agent intercepte les prompts du client, il est capable de les augmenter avec des informations contextuelles issues de la recherche dans la base vectorielle afin d'élaborer une réponse plus spécifique. L'Agent peut également inclure dans sa réponse une référence vers le document S3 qui a servi à son élaboration.




Nous pouvons ensuite ajouter à l'Agent des **Action Groups**.

Chaque Action Group possède une description et référence une fonction Lambda à laquelle on associe une spécification **OpenAPI** (standard pour la définition des APIs).







Grâce à la définition de vos APIs, l'agent est capable de comprendre quelles fonctionnalités vous avez implémentées et peut déterminer comment appeler vos APIs pour répondre aux demandes du client.

Après avoir obtenu les directives de votre client et établi un plan d'action (quelles APIs appeler, dans quel ordre et avec quels paramètres), l'agent est capable **d'agir**, en **invquant** la fonction lambda, qui fait office de point d'entrée pour votre API.

Le tableau ci-dessous résume les différentes indications que votre agent peut prendre en considération :

Type d'instruction	Exemple d'instruction	Conséquence	Exemple de prompt/réponse
Instruction de l'Agent	<i>Tu es un assistant de shopping. Tu es poli, courtois et patient. Tu conseilles les clients et les assiste pour effectuer leurs achats sur notre site internet.</i>	Ces instructions, très généralistes, permettent de donner un cadre à l'agent.	 ▶ <i>Bonjour, je suis votre assistant personnel de shopping. Comment puis-je vous aider aujourd'hui ?</i>
Instruction pour l'utilisation de la Knowledge Base	<i>Lorsqu'on te pose une question relative à notre politique de remboursement ou au service de réclamation, utilise cette base de connaissances.</i>	L'agent est capable d'interroger une base de données vectorielle construite avec la méthode RAG. La base de données peut contenir un vaste corpus documentaire vectorisé indiquant les différentes politiques de retour et de remboursement. L'agent dispose ainsi des informations les plus récentes concernant les règles établies par l'entreprise.	 ▶ <i>Le vêtement que j'ai reçu n'est pas à ma taille, comment puis-je l'échanger ?</i>  ▶ <i>Je suis désolé d'apprendre que le vêtement que vous avez reçu n'est pas à votre taille. Pour effectuer un échange, voici les étapes que vous pouvez suivre : [...]</i>

Type d'instruction	Exemple d'instruction	Conséquence	Exemple de prompt/réponse
Instruction pour l'utilisation de la Knowledge Base	<i>Lorsqu'on te pose une question relative à nos procédés de fabrication, utilise cette base de connaissances.</i>	L'agent est capable d'extraire des informations provenant d'une brochure présentant les méthodes de fabrication des textiles.	 <p>► Est-ce que le pull est fabriqué avec de la laine provenant d'un élevage français ?</p>  <p>► Bien sûr ! Le pull que vous avez choisi est fabriqué avec de la laine provenant d'un élevage français de renom.</p> <p><i>Nous sommes fiers de privilégier la qualité et la provenance de nos matériaux pour offrir à nos clients des produits de la plus haute qualité.</i></p>
Action Group Description	<i>Actions permettant de filtrer, rechercher et commander un produit.</i>	<p>L'agent est capable d'appeler les APIs du catalogue de produits et peut donc effectuer une recherche à partir des informations fournies par le client.</p> <p>Cette alternative aux Knowledge Bases permet d'exploiter une base de données non vectorielle déjà existante, telle qu'une base de données relationnelle ou orientée graphe.</p> <p>L'agent est également capable d'appeler l'API permettant d'initier et de valider une commande.</p>	 <p>► Je cherche un pull noir en taille M issu de la dernière collection pour un prix maximal de 80 euros.</p>  <p>► Bien sûr, voici plusieurs options de pulls noirs en taille M :</p> <ul style="list-style-type: none"> - Pull Noir "Élégance Urbaine" Marque : ModerneChic Prix : 75 euros Description : Ce pull noir est conçu pour un look élégant et urbain. - Pull Noir "Minimaliste Chic" Marque : PureClasse Prix : 80 euros Description : Si vous recherchez un look minimaliste et chic, ce pull est fait pour vous. <p>[...]</p>  <p>► Je souhaite commander le pull Minimaliste Chic.</p>  <p>► Excellent choix ! J'ai ajouté l'article suivant à votre panier :</p> <ul style="list-style-type: none"> Article : Pull Noir Minimaliste Chic Taille : M Prix : 80 euros Adresse : [...] <p><i>Souhaitez-vous confirmer cette commande et procéder au paiement ?</i></p>

Type d'instruction	Exemple d'instruction	Conséquence	Exemple de prompt/réponse
Action Group Description	<i>Actions permettant de créer et de gérer le compte de fidélité du client.</i>	L'agent est capable de récupérer les informations du client afin de créer son programme de fidélité en faisant appel aux APIs adéquates.	<p> ► <i>Avez-vous un programme de fidélité ?</i></p> <p> ► <i>Oui, bien sûr, nous avons une carte de fidélité qui vous permet de bénéficier de 10 % de réduction sur votre premier achat. Seriez-vous intéressée ?</i></p> <p> ► <i>Oui</i></p> <p> ► <i>Très bien, quel est votre nom ?</i></p> <p> ► <i>Jane Doe</i></p> <p> ► <i>Vous êtes bien enregistrée dans notre base de clients. Souhaitez-vous rechercher un premier article pour bénéficier de votre réduction ?</i></p>



CONCLUSION

L'INTELLIGENCE
ARTIFICIELLE GÉNÉRATIVE
TRANSFORMERA-T-ELLE
NOS HABITUDES ?

SANS AUCUN DOUTE,
ET IL NE VOUS A PAS ÉCHAPPÉ
QUE C'EST DÉJÀ LE CAS.

CHATGPT A ATTIRÉ UN MILLION D'UTILISATEURS EN SEULEMENT CINQ JOURS APRÈS SON LANCEMENT ET EN COMPTE AUJOURD'HUI PLUS DE 100 MILLIONS.

Une telle adoption par le grand public témoigne de l'intérêt croissant pour les services interactifs fondés sur la compréhension du langage naturel.

J'espère que ce livre vous a offert un aperçu de l'intelligence artificielle générative et des différentes méthodes (fine-tuning, RAG et utilisation des agents) pouvant être mises en œuvre pour étendre ses capacités.

Je recommande vivement aux entreprises de commencer dès maintenant à explorer les possibilités offertes par les nouveaux services AWS d'intelligence artificielle générative.

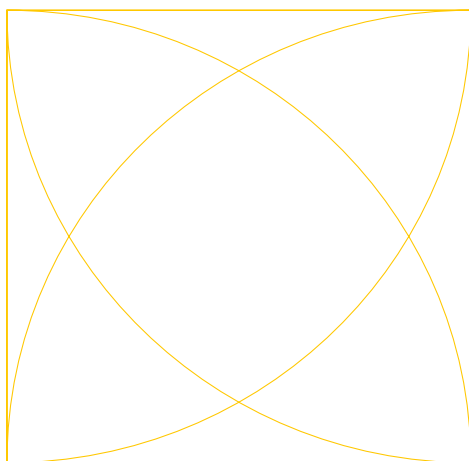
Si vos ressources sont limitées, je vous conseille de ne pas vous lancer dans le développement from scratch de solutions RAG (à moins bien sûr que des spécificités propres à votre contexte ne le justifient). Effectivement, la tendance actuelle des fournisseurs Cloud tend vers l'automatisation et la promotion d'offres managées sans serveur pour l'intelligence artificielle générative.

Opter pour le Cloud et cette typologie de service vous permettra de profiter à moindre coût des dernières avancées technologiques en matière d'intelligence artificielle générative, tout en réduisant les risques liés à des développements complexes et onéreux susceptibles de ne pas aboutir.

Il est donc essentiel de concevoir dès à présent les situations pour lesquelles l'intelligence artificielle générative pourrait être utilisée afin de développer votre activité.

Ippon Technologies dispose d'une expertise approfondie en matière de Cloud et de Data. N'hésitez donc pas à nous contacter si vous souhaitez être accompagné dans vos projets de transformation.

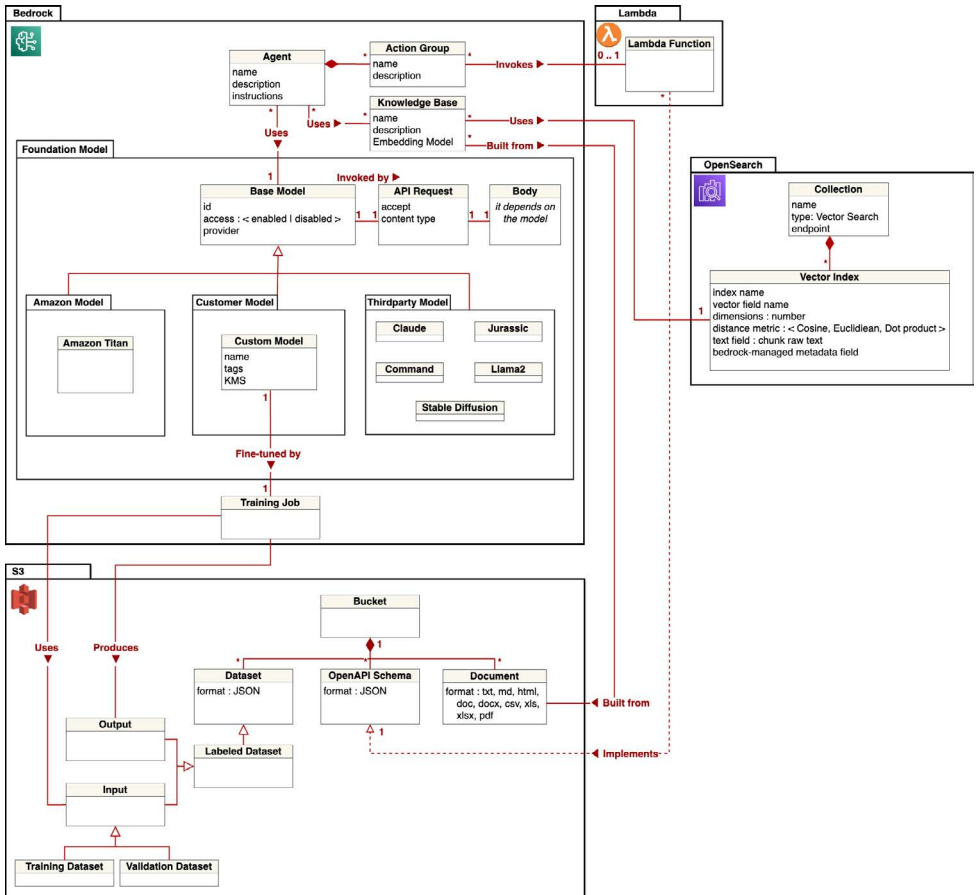
Le futur est à votre portée — il ne tient qu'à vous de saisir les outils disponibles pour façonner les usages de demain.



ANNEXES



UML : DIAGRAMME DE CLASSES DU SERVICE AWS BEDROCK ET DE SES INTÉGRATIONS



INSURANCE CLAIMS AUTOMATION API

openapi: 3.0.0

info:

title: Insurance Claims Automation API

version: 1.0.0

description: >

APIs for managing insurance claims by pulling a list of open claims, identifying outstanding paperwork for each claim, and sending reminders to policy holders.

paths:

/claims:

get:

summary: Get a list of all open claims

description: >

Get the list of all open insurance claims.

Return all the open claimIds.

operationId: getAllOpenClaims

responses:

"200":

description: >

Gets the list of all open insurance claims for policy holders.

content:

application/json:

schema:

type: array

items:

type: object

properties:

claimId:

type: string

description: Unique ID of the claim.

policyHolderId:

type: string

description: >

Unique ID of the policy holder who has filed the claim.

claimStatus:

type: string

description: >

The status of the claim.

Claim can be in Open or Closed state.




```
/claims/{claimId}/identify-missing-documents:
get:
  summary: Identify missing documents for a specific claim
  description: >
    Get the list of pending documents that need to be uploaded
    by policy holder before the claim can be processed.
    The API takes in only one claim id and returns the list of documents
    that are pending to be uploaded by policy holder for that claim.
    This API should be called for each claim id.
  operationId: identifyMissingDocuments
  parameters:
    - name: claimId
      in: path
      description: Unique ID of the open insurance claim.
      required: true
      schema:
        type: string
  responses:
    "200":
      description: >
        List of documents that are pending to be uploaded by policy holder
        for insurance claim.
      content:
        application/json:
          schema:
            type: object
            properties:
              pendingDocuments:
                type: string
                description: The list of pending documents for the claim.
```

```
/send-reminders:
post:
  summary: >
    API to send reminder to the customer about pending documents
    for open claim
  description: >
    Send reminder to the customer about pending documents for open claim.
    The API takes in only one claim id and its pending documents at a time,
    sends the reminder and returns the tracking details for the reminder.
    This API should be called
    for each claim id you want to send reminders for.
  operationId: sendReminders
  requestBody:
    required: true
    content:
      application/json:
        schema:
```



```
type: object
properties:
  claimId:
    type: string
    description: Unique ID of open claims to send reminders for.
  pendingDocuments:
    type: string
    description: The list of pending documents for the claim.
required:
  - claimId
  - pendingDocuments
responses:
  "200":
    description: Reminders sent successfully.
    content:
      application/json:
        schema:
          type: object
          properties:
            sendReminderTrackingId:
              type: string
              description: >
                Unique Id to track the status of the send reminder Call.
            sendReminderStatus:
              type: string
              description: Status of send reminder notifications.
  "400":
    description: >
      Bad request. One or more required fields are missing or invalid.
```



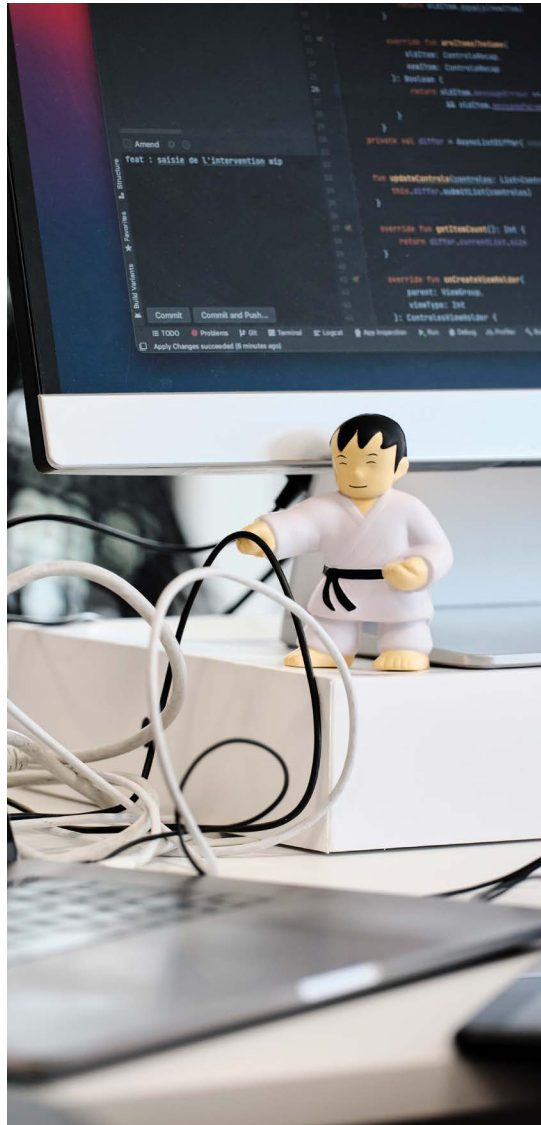
CRÉDITS PHOTOS

François Delauney
Christina Wocintechchat
Lucas Bravo
Junior Teixeira
freepik.com
rawpixel.com
unsplash.com

QUELQUES MOTS SUR IPPON TECHNOLOGIES

Cabinet de conseil et d'expertises en technologie, international et indépendant, Ippon accompagne la transformation numérique des entreprises, en les aidant à concevoir leur stratégie et à déployer leur roadmap à l'échelle, afin de délivrer rapidement la valeur attendue.

Partenaire Advanced Tier Services d'AWS, Ippon accélère les programmes de transformation, grâce à une maîtrise des méthodologies cloud et la compréhension des besoins de ses clients. Les compétences d'Ippon en matière de Data, DevOps et Migration sont reconnues par AWS à travers l'obtention de ces trois compétences.



- AWS Glue Delivery
- Amazon EMR Delivery
- AWS Lambda Delivery
- DevOps Services Competency
- Amazon API Gateway Delivery
- Migration Services Competency
- Well-Architected Partner Program
- Data & Analytics Services Competency





CONTACTEZ-NOUS !

fr.ippon.tech
blog.ippon.fr
twitch.tv/ippontech

contact@ippon.fr
+33 1 46 12 48 48