PROGRAMMEZ!
Le magazine des développeurs

SPÉCIAL AMAZON WEB SERVICES

Bien démarrer avec Amazon Web Services

DÉVELOPPER ET INTÉGRER DE L'IA DANS VOS APPLICATIONS

MACHINE LEARNING, DEEP LEARNING, IOT : DU CODE, DU CODE ET DU CODE !

SPÉCIAL HIVER 20/21

Le seul magazine écrit par et pour les développeurs

M 01642 - 3H - F: 6,50 € - RD

Printed in EU - Imprimé en UE - BELGIQUE 7 € - Canada 9,80 $ CAN - SUISSE 13,10 FS - DOM Surf 7,50 € - TOM 1020 XPF - MAROC 55 DH

PROGRAMMEZ!

LE MAGAZINE DES DÉVELOPPEURS

SAVOIR C'EST POUVOIR !

Abonnez-vous dès aujourd'hui sur www.programmez.com

Kiosque / Abonnement
Version papier / Version PDF

Contenus

Divers

Programmez! est une publication bimestrielle de Nefer-IT.
Adresse : 57, rue de Gisors 95300 Pontoise – France. Pour nous contacter : redaction@programmez.com

#SPÉCIALFROID20202021 ALIAS #HORSSÉRIE3*

"By your command"

On parle beaucoup de cloud computing, d'intelligence artificielle, de machine learning, d'analyse des images et des textes, etc. Mais finalement de quoi parle-t-on réellement ? Le terme « IA » est tellement utilisé pour tout et n'importe quoi que l'on s'y perd sur ce que l'on entend aujourd'hui par IA.

IA, au sens strict du terme, n'existe pas. Du moins pas dans des outils facilement disponibles. Oui, certains robots possèdent une certaine capacité, parfois parle-t-on d'intelligence. Mais nous sommes loin des Cylons ou d'un ordinateur comme HAL ou Terminator. Par contre, ce qui est disponible sous le terme IA, c'est tout ce qui est Machine Learning, Deep Learning, réseaux neuronaux, etc. Le Machine Learning est sans doute la partie la plus connue. Car l'écrasante majorité des IA du marché est en réalité du Machine Leaning, avec du Deep Learning.

On va parler apprentissage, algorithme, données d'entraînement, analyse d'images, de sons, de vidéos pour prévenir, agir préventivement. Cette dernière partie est particulièrement intéressante. C'est tout ce qui l'on appelle la vision, la reconnaissance d'objets. Pour des usages très précis, les mécanismes de vision et de reconnaissance d'objets sont particulièrement utiles, par exemple dans le monde médical pour aider les médecins à « voir » les anomalies ou encore dans l'industrie ou dans la prévention de risques.

Dans ce numéro spécial hiver 2020-2021, Programmez! et Amazon Web Services vous proposent une plongée dans le monde de l'IA.

Comment créer et entraîner des modèles ?

Comment utiliser TensorFlow ?

Combiner IA et IoT.

Comment extraire intelligemment des textes d'une vidéo ou d'un fichier audio ?

Comment manipuler un texte et le traduire à la volée ?

Comment détecter une fraude ? Comment reconnaître rapidement des objets ?

Voilà en quelques mots tout ce que nous vous proposons en 84 pages ! Le menu est dense et velu. Et surtout, pour vous montrer que les services AWS supportent tous les langages actuels, nous manipulerons les SDK, les API et les services en Java, Python, JavaScript !
Je sens que certaines/certains ont failli s'évanouir. Oui, oui, on peut faire de l'IA sans utiliser C++ ou Python.

En readme, nous ferons une rapide overview/présentation du cloud AWS : comment créer un compte, ce que contient AWS, les différents SDK, etc. Mais ne vous inquiétez pas, passée la phase de démarrage pour padawan débutant, nous atteindrons le niveau supérieur.

Installez-vous bien. Préparez votre café en perfusion (il faut ce qu'il faut). Faites chauffer votre 4G, 5G, box internet.

François Tonic
Bio = false //pour les curieuses/curieux voir en **

Rédacteur en chef
ftonic@programmez.com

**En réalité le #247*

LES PROCHAINS NUMÉROS

Programmez! n°245

PHP 8.0 : overview & 1er bilan

Retour sur Drupal 9

Disponible dès le 5 mars

*** Historien, historien de l'informatique, journaliste informatique depuis 25 ans, maker (quand je trouve du temps la nuit)*

Rédacteur en chef depuis 20 ans (bref : vous me supportez depuis 20 ans)

Geek depuis 1983

Développeur-testeur à l'ancienne (avant StackOverFlow)

Adepte du RTFM

Auteur de « Histoire de la micro-informatique » volumes 1/2/3 + 6 livres (égyptologie)

Conférencier et animateur de conférences IT et en égyptologie

Fan de Dark Vador, des Cylons, du n°6, de Caprica 6 et de l'ordre 66

Rédacteur en chef de Technosaures, le magazine sur les anciennes machines et technologies (1970-2000)

A connu Windows 3.1

Victime de Doom

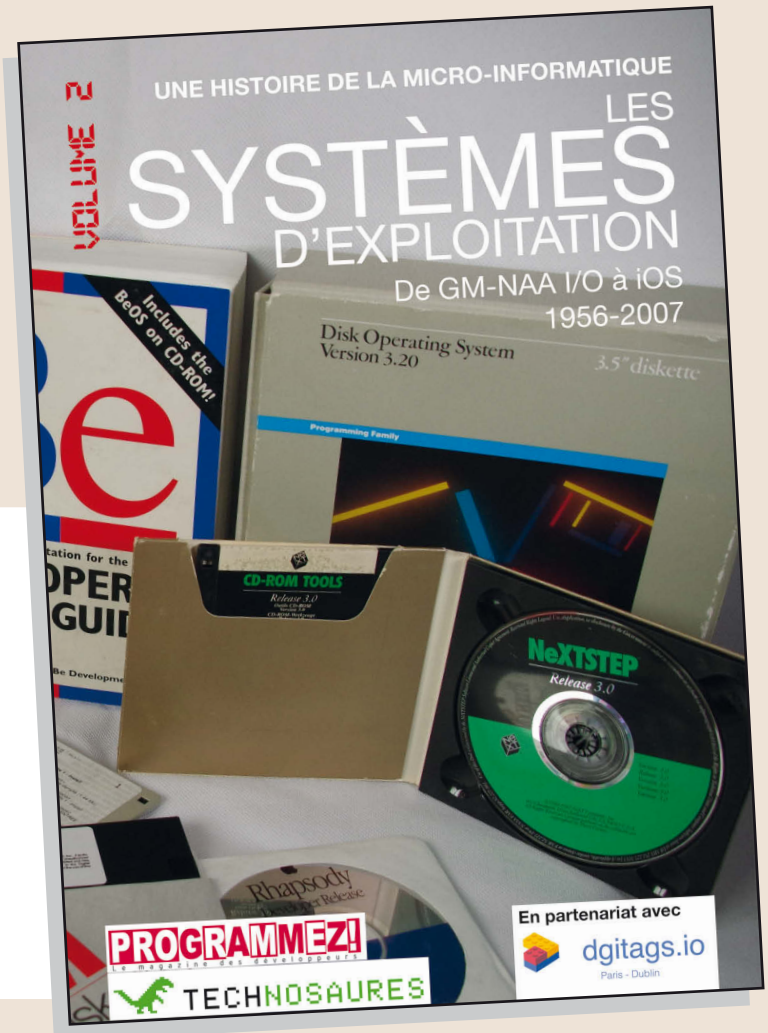
Peut exposer de superbes ordinateurs pour anniversaire, fête d'entreprise, soirée développeur, salon IT.

Le CADEAU idéal pour toutes/tous les geeks !

UNE HISTOIRE DE LA MICRO-INFORMATIQUE

Volume 3 : 90 nouvelles machines, + d'ordinateurs français !

116 pages. Format mook.



UNE HISTOIRE DE LA MICRO-INFORMATIQUE

Volume 2 : les systèmes d'exploitation de 1956 à 2007

100 pages. Format mook.

Commandez directement sur
www.programmez.com/catalogue/livres

Avertissement : cet agenda est prévisionnel.
Tout dépendra de l'évolution de la situation Covid et des restrictions imposées par les Etats.

Les événements Programmez!

Nos prochains meetups :

23 février : les dernières nouveautés Java
23 mars : l'art du refactoring et le code legacy
27 avril : Flutter
Mai : pause = true
29 juin : sujet à venir

Nos prochaines DevCon :

25 mars : conférence spéciale informatique quantique. À partir de 13h30.

Informations & inscription : programmez.com

février

Lun.	Mar.	Mer.	jeu.	Ven.	Sam.	Dim.
1	2	3	4	**5** Web Stories Conf.	**6** FOSDEM (Bruxelles) Uniquement virtuel	**7**
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28

mars

Lun.	Mar.	Mer.	jeu.	Ven.	Sam.	Dim.
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	**16**	**17**	**18**	**19**	20	21
IoT Week 2021						
22	23	24	25	26	27	28
29	30	31				

avril

Lun.	Mar.	Mer.	jeu.	Ven.	Sam.	Dim.

mai

Lun.	Mar.	Mer.	jeu.	Ven.	Sam.	Dim.
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	**18** Devopsdays Paris	19	20	21	22	23
24	25	26	27	28	29	30
31	NCrafts Paris : pas de date connue					

juin

Lun.	Mar.	Mer.	jeu.	Ven.	Sam.	Dim.
	1	2	3	4	5	6
7	8	9	10	**11** Devfest Lille 2021	12	13
14	**15**	16	17	18	19	20
BlendWebMix (Lyon)						
21	22	23	24	25	26	27
28	**29**	**30**	1 juil.	2 juil.		
Hack in Paris						
		Devoxx France				

juillet

Lun.	Mar.	Mer.	jeu.	Ven.	Sam.	Dim.
			1 Hack in Paris	**2** Devoxx France	**3**	**4**
				Nantes Maker Campus		
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	37	28	29	30	31	

août

Lun.	Mar.	Mer.	jeu.	Ven.	Sam.	Dim.

septembre

Lun.	Mar.	Mer.	jeu.	Ven.	Sam.	Dim.

octobre

Lun.	Mar.	Mer.	jeu.	Ven.	Sam.	Dim.
				1	2	3
4	5	6	7	8	9	10
11	12	13	**14** VolCamp	**15**	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

novembre

Lun.	Mar.	Mer.	jeu.	Ven.	Sam.	Dim.

décembre

Lun.	Mar.	Mer.	jeu.	Ven.	Sam.	Dim.

SANS DATE

- MixIT
- BreizhCamp
- DevFest du bout du monde
- RivieraDev
- DevFest Nantes
- Best of Web
- Flutter Con Paris
- Sunny Tech
- DevFest Toulouse

Revoir les vidéos de nos dernières conférences sur YouTube
https://tinyurl.com/ygmzlh9e

Page meetup.com
https://www.meetup.com/fr-FR/Meetup-Programmez/

N°5
Disponible !

Technosaures n°5

Atari ST - Commodre SX-64 - Amiga 2000
Apple IIGS - Alan Kay - GOTEK

Commandez directement sur **programmez.com**

6,66 € *(+frais de port*)* **36 pages**

*Revue trimestrielle. Editée par Nefer-IT. *Avec frais de port : 7,66 €*

Abonnement 1 an : 30 €

François Tonic

Amazon Web Services, kézako ?

Amazon Web Services (AWS) est l'entité dédiée au Cloud avec l'ensemble des services (SaaS, PaaS, IaaS, etc.), l'infrastructure et les datacenters. AWS devance ses principaux concurrents que sont Microsoft, Google, IBM, Oracle. Si S3 et EC2 sont deux services connus et reconnus, ce ne sont que deux services parmi des centaines d'offres ! Aujourd'hui, AWS couvre un champ fonctionnel considérable qu'un développeur ne peut tout maîtriser. Tous les ans, de nouveaux services apparaissent et de multiples évolutions des services existants sortent.

Les grands domaines fonctionnels

Comme les autres grands fournisseurs de cloud, les services d'AWS se divisent en plusieurs grandes « unités » fonctionnelles qui correspondent à des fonctionnalités, des usages précis. Par exemple, nous trouvons tous les services liés à l'infrastructure et aux ressources machines. Les services compute proposent par exemple l'ensemble des instances EC2 et les services liés aux containers Docker. Dans la partie base de données, on retrouvera tous services de données et les bases de données : Aurora, DynamoDB, Neptune, etc.

On retrouve les services liés à la sécurité, aux analyses, aux architectures hybrides, aux IoT, au Machine Learning, au stockage, etc.

Désormais, on ne parle plus forcément de SaaS, PaaS, IaaS. Ces notions se mélangent de plus en plus, notamment avec les architectures / infrastructures par conteneurs et l'hybridation. Cependant, la partie infrastructure (IaaS) reste très visible notamment pour les environnements historiques (ce que l'on appelle le patrimoine IT, les applications legacy) que l'on virtualise. **Figure 1**

1 service = 1 SLA

Un des éléments clés des services cloud est la qualité de service, sous-entendu, la disponibilité dudit service exprimé avec plusieurs 9 après « 99, ». La notion de SLA varie selon les fournisseurs et les services.

Par exemple :

- EC2 / EBS / ECS / EKS: minimum 99,99 % de disponibilité ;
- S3 : minimum 99,9 % de disponibilité, et 99,999999999 % de durabilité ;
- RDS : minimum 99,95 % de disponibilité ;
- DynamoDB : minimum 99,99 % de disponibilité, 99,999 % avec Global Tables ;
- ElastiCache (Memcache, Redis) : minimum 99,9 % de disponibilité.

Si nous regardons plus spécifiquement les services IA et ML (Machine Learning), nous trouvons :

- Minimum 99,9 % de disponibilité pour Lex, Polly, Translate, Textract, Kendra, etc.
- 99,95 % de disponibilité pour SageMaker.

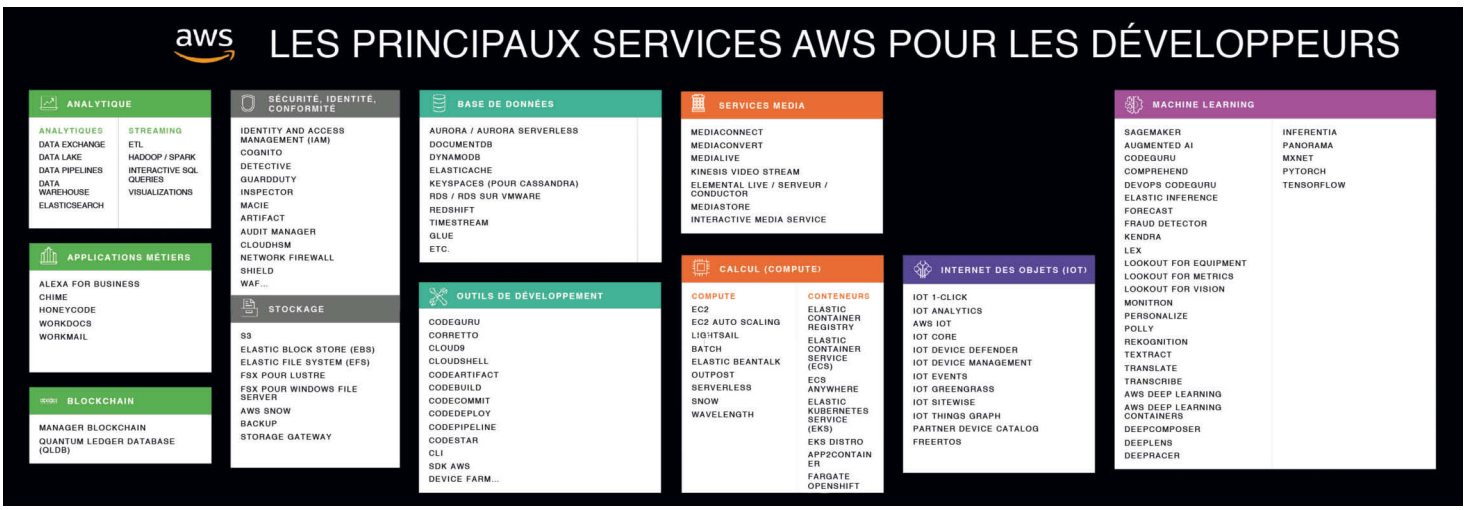
Tous les services AWS ne possèdent pas d'un niveau de service contractuel. C'est notamment le cas pour les services préversions. Au % de disponibilité, le SLA du service définit aussi la couverture contractuelle (ce qui est couvert et non couvert en cas de panne).

Pour en savoir plus :
https://aws.amazon.com/fr/legal/service-level-agreements/

Combien coûte AWS ?

Nous ne le répéterons jamais assez : les services cloud reposent sur plusieurs modèles économiques : le paiement à l'usage (typiquement, chaque minute est due), souscription.

Figure 1



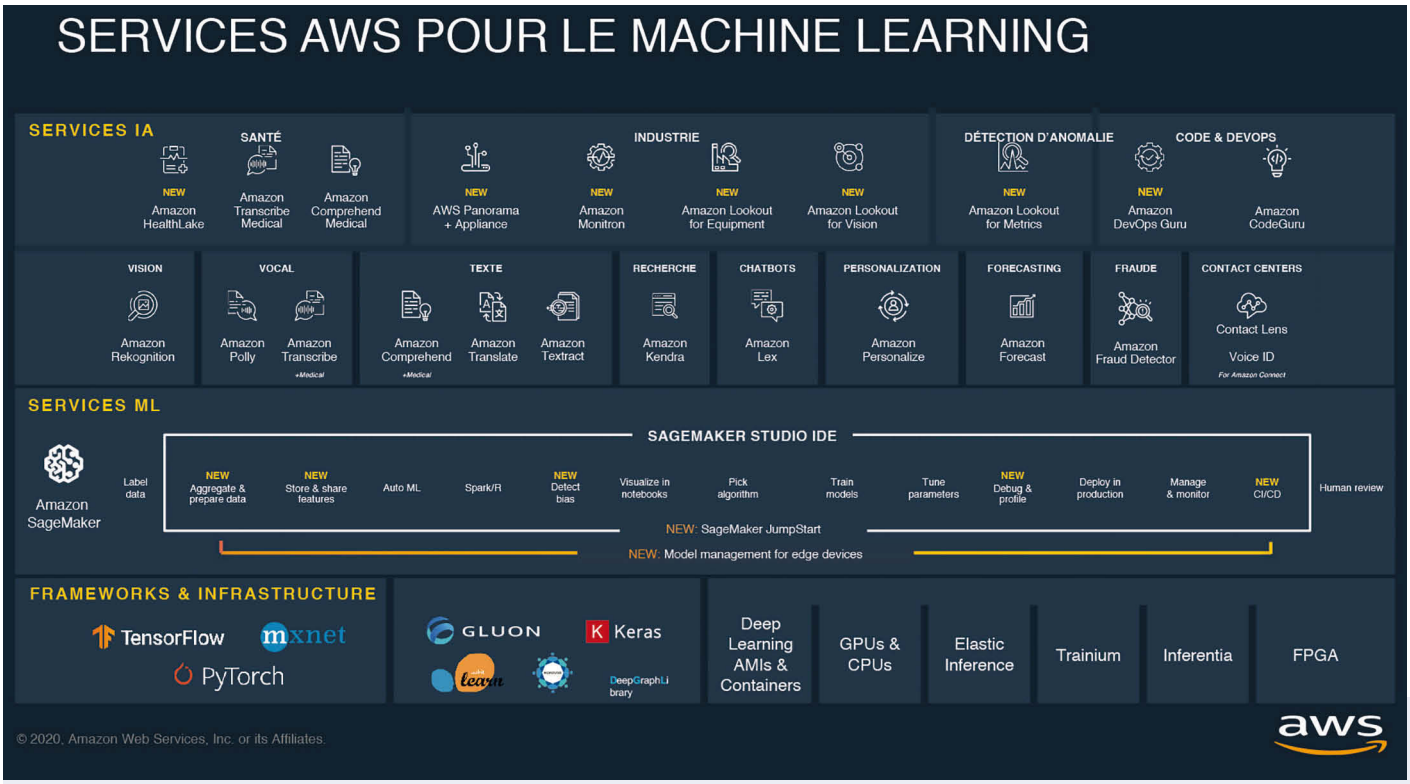


Figure 2

QU'ALLONS-NOUS UTILISER DANS CE HORS-SÉRIE ?

Vous l'aurez compris, dans ce numéro spécial AWS, nous n'allons pas utiliser tous les services disponibles. Pourquoi ?

1 cela ne servira à rien
2 on utilise les services dédiés aux besoins de nos projets
3 on utilise les services tiers pour exécuter les services d'IA, de machine learning, etc.

Souvent, le cloud ressemble à un Tetris ou plutôt à un Lego que l'on assemble. En effet, Pour utiliser des services de bases de données, d'analyses ou de machine learning, il faut des ressources, c'est-à-dire du compute, pour déployer lesdits services. Ainsi, nous allons user et abuser d'instances EC2 et du stockage S3.
Parfois, le plus difficile est de comprendre l'empilement de services : base de données, stockage, compute, etc. **Figure 2**

Dans ce numéro, nous allons donc utiliser en priorité les services liés au machine learning, à la vision. Comme nous le verrons, ces services ont besoin de ressources machines et de stockages notamment des instances EC2 et du stockage S3. Il ne faut jamais oublier que la plupart des applications cloud ou utilisant des services cloud s'appuient sur l'allocation de ressources dites compute (calculs).

La souscription peut avoir plusieurs aspects : des formules mensuelles ou annuelles. Certains fournisseurs proposent des souscriptions de plusieurs années. On retrouve, grosso modo, les mêmes modèles économiques, quel que soit le fournisseur. AWS propose 3 modèles : paiement à l'usage, paiement selon des capacités réservées et la dégressivité des tarifs selon les volumes consommés.

Les développeurs peuvent profiter d'une offre gratuite. Elle couvre plus de 85 services. Dans cette formule, les services peuvent rester gratuits, gratuits durant 12 mois et gratuit durant une période d'essai indiquée à la souscription.

Quelques exemples **(voir tableau)**

La facture peut rapidement monter si vous ne faites pas attention. Vérifiez la tarification des services utilisés. Arrêtez les services que vous n'utilisez pas : mettre en pause un service ne suffit pas pour ne pas payer le service.
Supprimer les services, instances que vous n'utilisez plus. Purgez systématiquement les données et les ensembles de données que vous n'utilisez pas. En machine learning, vous pouvez utiliser un volume important d'images, de fichiers.

	Gratuit ad vitam aeternam	12 mois gratuits	Période d'essai
EC2	-	750 heures / mois (sur une gamme d'instances)	-
S3	-	5 Go de stockage standard. + quotas de requêtes GET et PUT	-
RDS (base de données)	-	750 heures / mois	-
DynamoDB	25 Go de stockage		
Chime (version basic)	Illimité	-	-
SageMaker	-	250 heures / mois	
Rekognition	-	5 000 images / mois	
Device Farm	-	-	250 minutes

Ne faites pas des traitements ou des requêtes inutiles : l'optimisation des flux permet d'économiser la ressource.

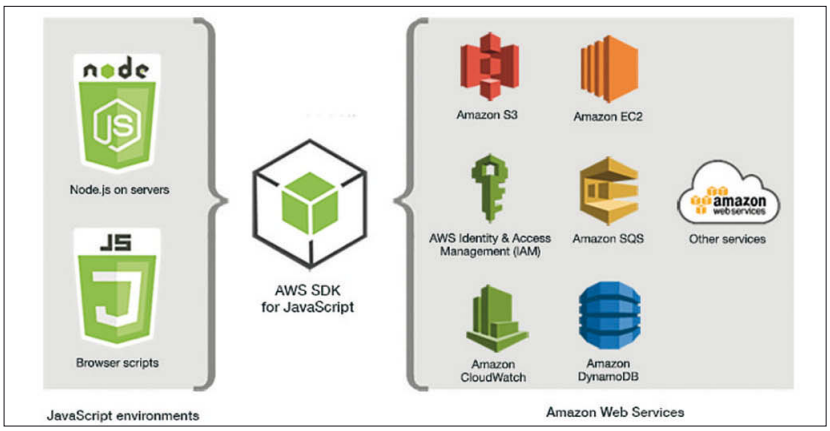
Estimez les volumes et les besoins en ressource pour calculer votre budget cloud.

François Tonic

Oui, AWS aime beaucoup le développeur !

Pour pouvoir utiliser, implémenter, développer des applications, le développeur a besoin de SDK, d'API. Les services peuvent être utilisés à travers plusieurs langages : C++, Go, Java, JavaScript, C# / .Net, Node.JS, PHP, Python et Ruby. Vous n'aurez pas besoin de changer de langages pour utiliser un service AWS. Comme nous le verrons dans ce hors-série, selon les sujets, nous utiliserons Python, Java ou JavaScript. Hormis cas particulier, les usages fournis peuvent être réalisés avec d'autres langages que ceux donnés en exemple.

Tout naturellement, AWS propose tout ce qu'il faut pour pouvoir utiliser et implémenter les services cloud dans les applications. Pour ce faire, les développeurs disposent d'un SDK global : SDK AWS. D'autres SDK spécifiques existent mais pour des usages très précis.



Par définition, le SDK AWS permet d'accéder aux services AWS depuis le langage utilisé (C++, Go, Java, JS, etc.). Si on regarde la version JS, côté utilisateur, le code s'exécute côté navigateur ou serveur (Node). Le SDK pour JS permet d'accéder aux services S3, EC2, identité, CloudWatch, etc.

L'installation est rapide via un simple npm install. Puis, on charge le SDK dans son code : AWS = require('aws-sdk'). Ensuite, il faut configurer les différents éléments (région, identifiants, les endpoints…).

Même si une différence d'implémentation peut exister selon le langage, les SDK AWS incluent l'ensemble des API.

Pour en savoir plus : lire l'article « bien démarrer le développement avec AWS ».

Documentation, section SDK & Toolkit :
https://docs.aws.amazon.com/index.html

Les principaux SDK

C++	Go	Java	JavaScript	.Net	Node.JS	PHP	Python	Ruby
SDK AWS	SDK AWS	SDK AWS	SDK AWS	SDK AWS	SDK AWS	SDK AWS	SDK AWS	SDK AWS
		SDK IoT	SDK mobile	SDK Unity Mobile			SDK IoT	
			SDK IoT	SDK Xamarin				
				Toolkit for VS* Team Services				

VS = Visual Studio

Les outils supportés

C++	Go	Java	JavaScript	.Net	Node.JS	PHP	Python	Ruby
Cloud9	Cloud9	Toolkit pour Eclipse	Toolkit pour VS	Toolkit pour VS	Toolkit pour VS	Cloud9	Toolkit pour PyCharm	Cloud9
		Toolkit pour IntelliJ	Cloud9		Cloud9		Toolkit pour IntelliJ	
							Toolkit pour VS	
							Cloud9	

Bien démarrer le développement sur AWS

Pour pouvoir utiliser les services qui vous sont présentés dans ce numéro spécial, vous devez disposer d'un compte AWS. Dans cet article, je vous montre comment créer et sécuriser votre compte AWS, si vous n'en avez pas encore.

Sébastien Stormacq

Principal Developer Advocate

Seb écrit du code depuis qu'il a touché pour la première fois à un Commodore 64 au milieu des années quatre-vingt. Il inspire les développeurs pour utiliser le cloud AWS, en utilisant son mélange secret de passion, d'enthousiasme, d'attention aux clients, de curiosité et de créativité. Ses intérêts portent sur les architectures logicielles, les outils de développement et l'informatique mobile, iOS en particulier.

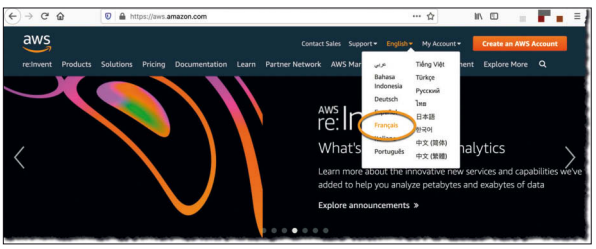
Si vous disposez déjà d'un compte AWS, jetez tout de même un œil à la section « Sécuriser mon compte » avant de passer aux autres articles.

> Pour suivre ce tuto, vous avez besoin d'une connexion internet, d'un ordinateur ou une tablette avec un navigateur internet moderne (Firefox, Edge, Safari, Chrome par exemple), un numéro de téléphone où vous pouvez recevoir un SMS, et une carte de crédit avec $1.

ÉTAPE 1 : créer mon compte

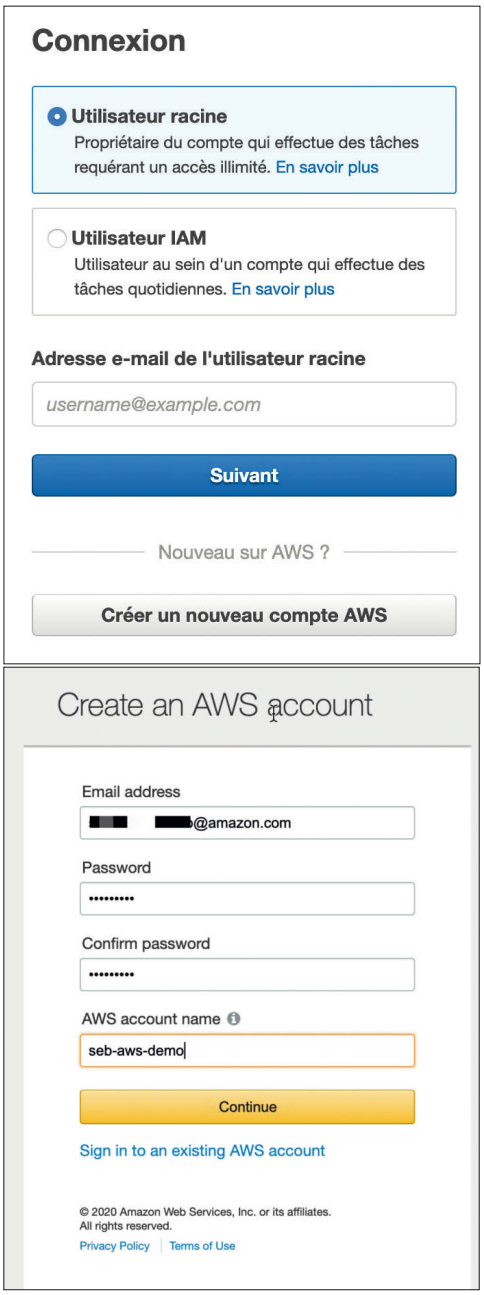
Pour créer un compte AWS, vous devez disposer d'une adresse email, d'un numéro de téléphone et d'une carte de crédit. Certains services AWS vous proposent une offre gratuite pendant 12 mois après la création de votre compte, certains pas. C'est pourquoi une carte de crédit valide est requise dès la création du compte. Pour en savoir plus sur les offres gratuites, regardez https://aws.amazon.com/fr/free. Dans cet article je vous montrerai comment configurer une alerte par email sur la base de vos limites de budget.

Pour créer un compte, j'ouvre mon navigateur et je vais sur https://aws.amazon.com. Si la page n'est pas en français, je change de langue dans le menu situé en haut à droite.



Ensuite je clique sur **Créer un compte AWS**.

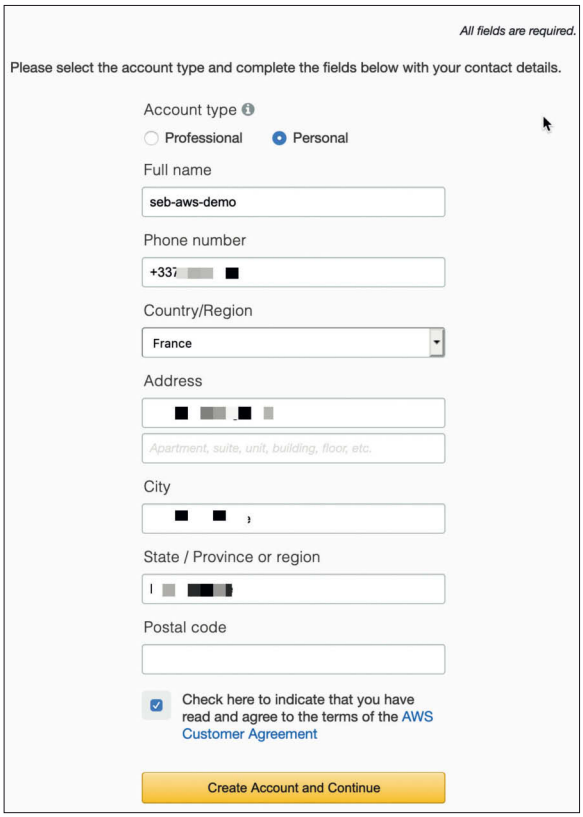
Sur l'écran suivant, je clique sur **Créer un nouveau compte AWS**.



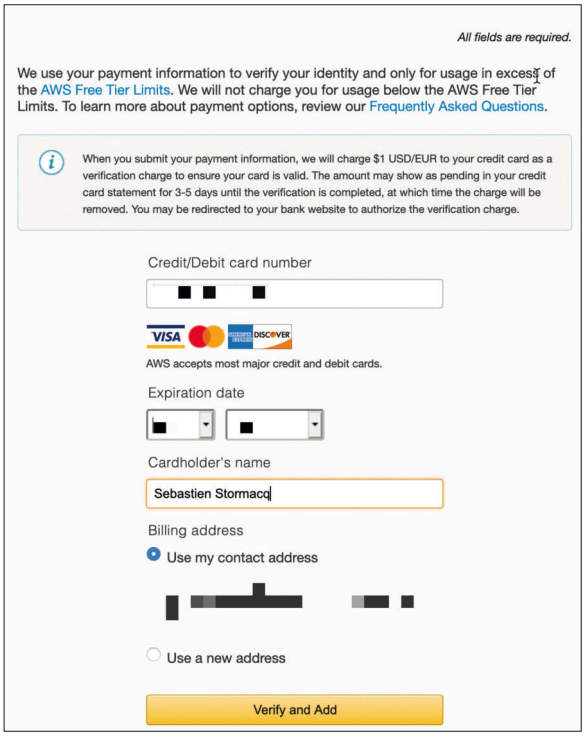
Sur le deuxième écran, je définis le type de compte (professionnel ou personnel) et je rentre quelques données complémentaires.

Si la console rebascule en anglais, vous pouvez changer la langue dans le menu en haut à droite.

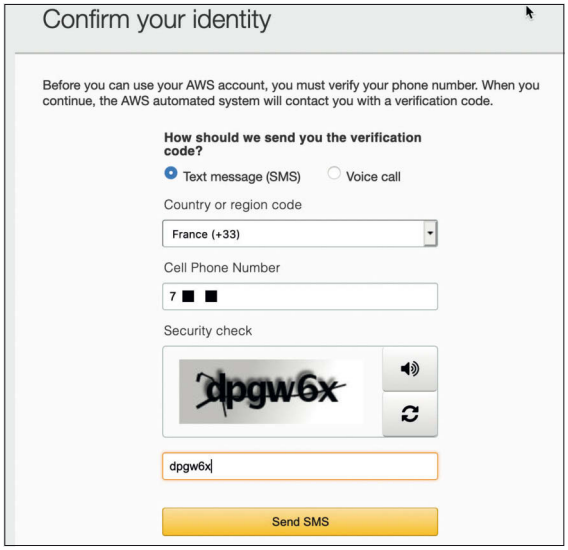
Je rentre une adresse email valide, un mot de passe unique et complexe et je donne un nom à mon compte. Le nom permettra plus tard d'indiquer sur quel compte AWS je veux me connecter, si j'en ai plusieurs. Le nom doit être mondialement unique et peut être changé plus tard.



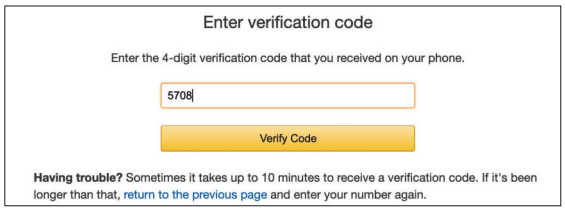
Sur le troisième écran, je rentre les détails de ma carte de crédit. AWS prélève $1 (environ 0.83 €) sur la carte pour vérifier qu'elle est valide. Ce montant vous sera remboursé sur votre première facture. Les cartes prépayées sont acceptées, du moment que vous avez au moins $1 dessus.

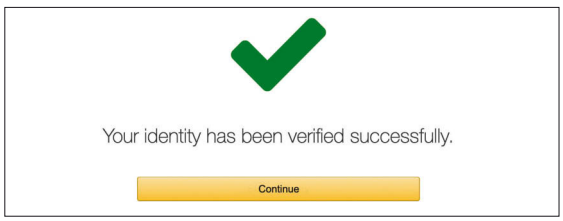


Finalement, je confirme mon numéro de téléphone. AWS vous envoie un code par SMS ou vous appelle pour vérifier le numéro.



Le SMS, dans cet exemple, arrive quasiment instantanément.

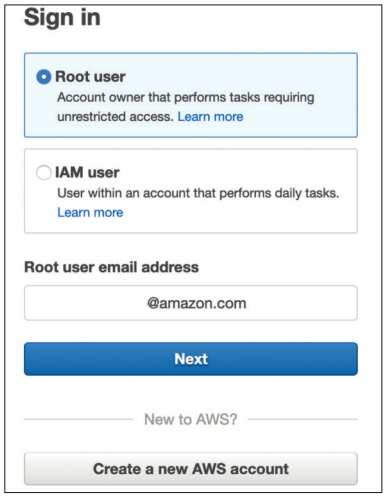




La dernière étape est de choisir mon contrat de support, entre basic (gratuit), développeur (à partir de $29 par mois) et business (à partir de $100 par mois).

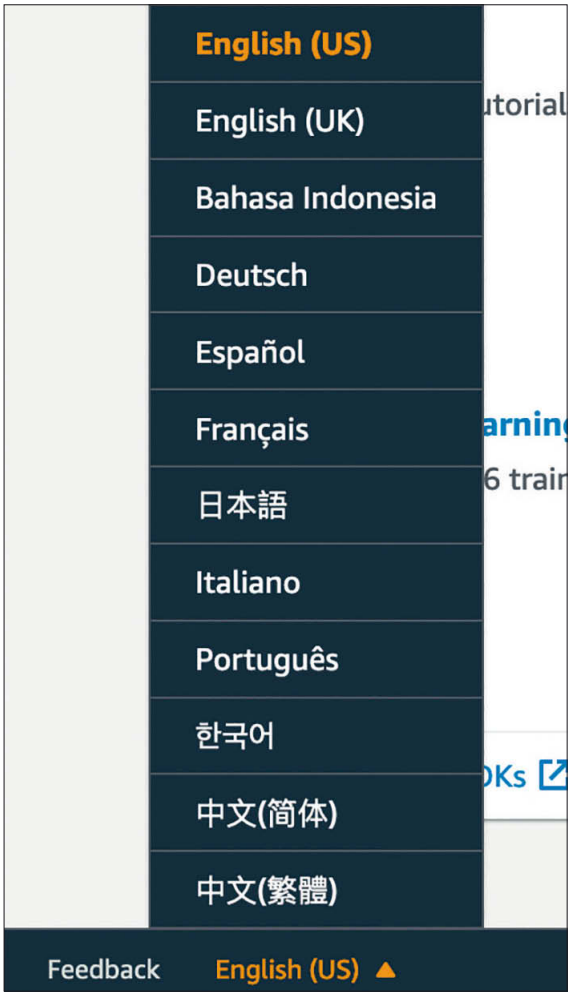
Et de se laisser rediriger vers la page d'authentification.

Maintenant que j'ai un compte, je m'y connecte avec l'adresse email et le mot de passe que j'ai choisi dans la première étape.



ÉTAPE 2 :
sécuriser son compte

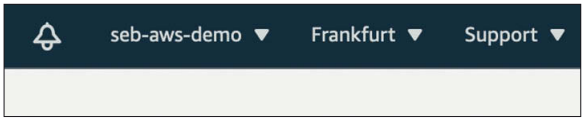
Une fois que je suis connecté, si ma console est en anglais, je change la langue en utilisant le menu en bas à gauche.



Ensuite, je choisis dans laquelle des 24 régions AWS je veux créer mon infrastructure. Le choix de la région dépend de plusieurs paramètres :

- L'endroit géographique où vous souhaitez conserver vos données. Ce choix est régi par des impératifs de réglementation ou de préférences.
- La proximité avec mes clients. Pour minimiser la latence, il est préférable de créer une infrastructure à proximité de l'endroit où se trouvent vos clients.
- La disponibilité des services. Tous les services ne sont pas disponibles dans toutes les régions. La matrice de déploiement des services est disponible ici https://aws.amazon.com/fr/about-aws/global-infrastructure/regional-product-services/
- Le prix. Comme les coûts d'opération de centres de données ne sont pas les mêmes dans tous les pays, les prix d'utilisation des services AWS varient d'une région à l'autre. De cette manière, Amazon peut proposer les prix les plus bas en fonction de ses coûts d'opération.

Le choix de la région par défaut se fait dans le menu en haut à droite.

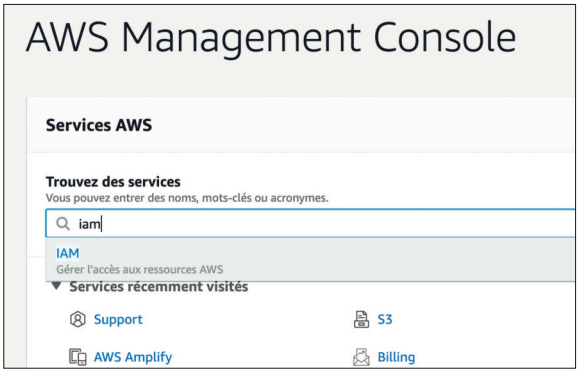


Maintenant que mon compte AWS est créé, voici cinq étapes fortement conseillées pour sécuriser mon compte :

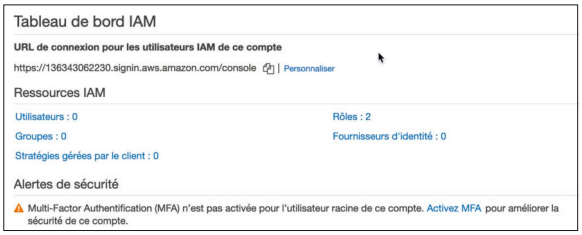
1. Configurer l'authentification à deux facteurs ;
2. Vérifier mon numéro de téléphone ;
3. Vérifier mes paramètres de sécurité ;
4. Créer un utilisateur IAM ;
5. Paramétrer une alerte de budget.

Configurer l'authentification à deux facteurs

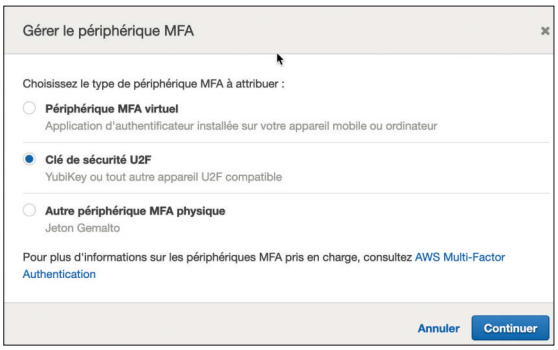
Comme pour chacun de vos comptes internet, il est fortement recommandé d'utiliser plusieurs facteurs d'authentification.
Dans la console AWS, je cherche le service Identity & Access Management (IAM).



Dans le tableau de bord IAM, sous la section **Alertes de sécurité**, je clique **Activez MFA**.



Je clique sur le bouton bleu **Activer MFA**.
La console AWS supporte trois types d'authentification par facteurs multiples : les applications qui génèrent des codes (1Password, Google Authenticator, HDE OTP, ou n'importe quelle application qui supporte le système OTP), les appareils de type U2F (les clés YubiKey par exemple) et les jetons avec écran qui génèrent des codes uniques (comme les jetons Gemalto).
Nous recommandons vivement d'utiliser un appareil physique pour le MFA de l'utilisateur principal du compte et d'utiliser les applications OTP pour les utilisateurs IAM (voir la section suivante).



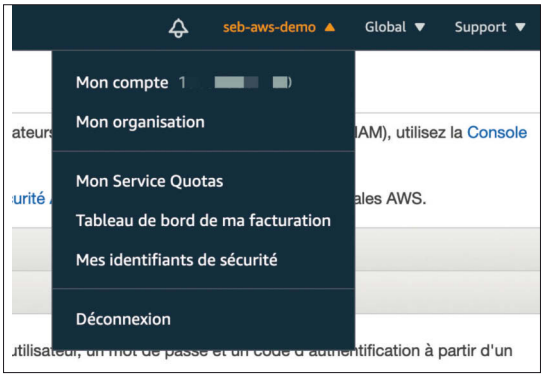
Dans cet exemple, je choisis d'utiliser ma clé Yubikey. La procédure varie légèrement en fonction de l'option choisie. Il suffit de se laisser guider par les instructions de la console.

Une fois l'appareil enregistré, il apparaît dans la console IAM et, à tout moment, je peux le supprimer et en enregistrer un autre.

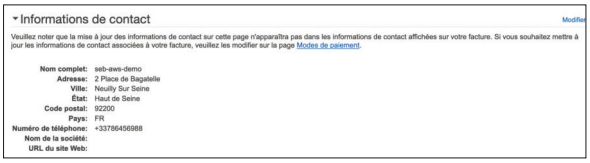
Vérifier votre numéro de téléphone

Au cas peu probable où j'oublierais mon mot de passe, je peux contacter le support AWS pour le réinitialiser. Pour des raisons de sécurité, les équipes de support AWS me rappelleront sur mon téléphone pour vérifier mon identité. Le support ne réinitialisera pas le mot de passe s'ils ne peuvent pas me joindre sur le numéro de téléphone renseigné sur mon compte (croyez-moi, ça m'est déjà arrivé). Il est donc très important que le numéro de téléphone soit correct. Pensez à le modifier si vous changez de numéro.

Dans la console AWS, je clique sur le nom de mon compte en haut à droite et je choisis l'option **Mon compte**.

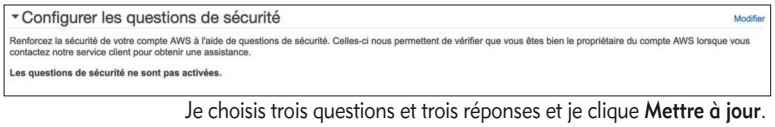


Sur l'écran suivant, je vérifie les informations renseignées sous **Information de contact**. Le cas échéant, je clique sur **Modifier** pour mettre mes informations à jour.



Vérifier mes paramètres de sécurité

Sur le même écran, je décide de choisir des questions et réponses de sécurité qui me seront demandées en cas d'oubli de mon mot de passe. Dans la section **Configurer les questions de sécurité**, je clique **Modifier**.

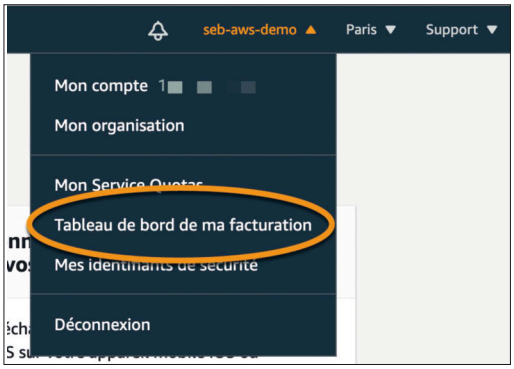


Je choisis trois questions et trois réponses et je clique **Mettre à jour**.

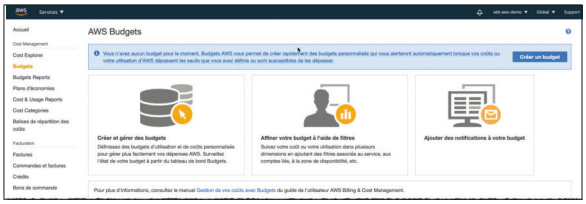
Paramétrer une alerte de budget

La prochaine étape consiste à paramétrer une alerte au cas où la facturation de mon compte approcherait la limite de mon budget. Imaginez que je démarre une grosse machine virtuelle, avec beaucoup de vCPU et de mémoire, juste pour un test pendant une heure ou deux, mais que j'oublie de la terminer une fois le travail fini. AWS va continuer à compter les heures et ma facture augmentera en conséquence. Pour éviter de recevoir une mauvaise surprise à la fin du mois, nous vous encourageons à définir des seuils d'alertes en fonction de votre budget. AWS vous enverra un message par email à chaque fois que le système de facturation atteindra un de vos seuils préconfigurés.

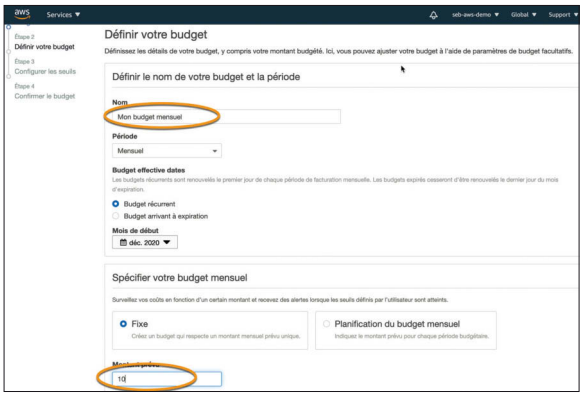
Dans la console AWS, je clique sur le nom de mon compte en haut à droite et ensuite sur **Tableau de bord de ma facturation**.



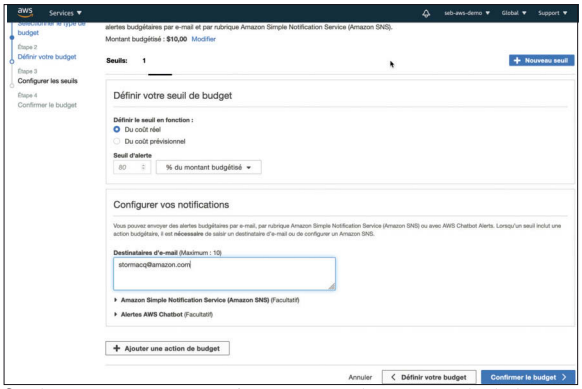
Sur l'écran suivant, je clique sur **Budgets** dans le menu à gauche et **Créer un budget**.



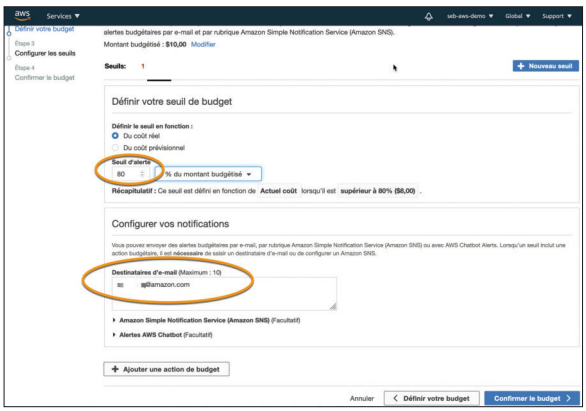
Je choisis **Budget de Coûts** et clique sur **Définir votre budget**.



Je donne un nom et un plafond à mon budget, ici $10 par mois. Je clique ensuite sur le bouton **Configurer les seuils** en bas de page.



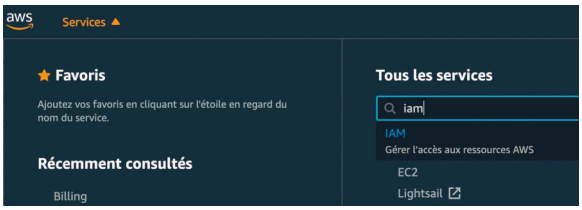
Sur l'écran suivant, je configure un ou plusieurs seuils d'alertes. Je peux par exemple créer une alerte quand le prévisionnel arrive à 50 % de mon budget et une autre quand le montant réel arrive à 80 %.



J'encode mon seuil d'alerte et je rentre la ou les adresses email où je veux que les notifications soient envoyées. Enfin, je clique **Confirmer le budget**. Sur l'écran suivant, je valide mes choix et clique **Créer**.
À tout moment, je peux réviser et ajouter des budgets et des alertes.

ÉTAPE 4 :
créer un utilisateur IAM

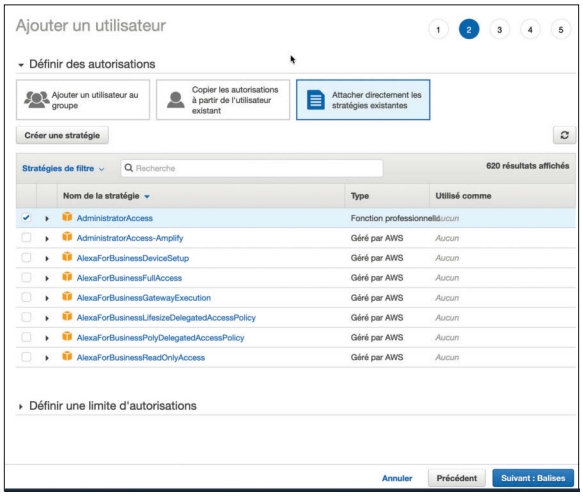
L'utilisateur que j'ai utilisé jusqu'à présent pour me connecter à mon compte AWS est appelé « utilisateur principal » (root account). C'est celui que j'ai utilisé pour créer le compte et dont le nom d'utilisateur est une adresse email. Tout comme l'utilisateur root sous Unix et Linux, ou l'utilisateur Administrator sous Windows, cet utilisateur a tous les droits, y compris celui de clôturer le compte AWS. Il n'est pas recommandé d'utiliser cet utilisateur pour vos tâches quotidiennes dans la console. Je vous conseille donc d'écrire son mot de passe sur une feuille, de la mettre sous enveloppe, de sceller le tout et de l'enfermer dans un coffre-fort. Ensuite vous stockez en sécurité, dans un autre coffre-fort, le jeton MFA utilisé comme second facteur d'authentification. Ce compte n'est à utiliser que dans les cas d'urgence.
Au quotidien, je peux utiliser différents utilisateurs, groupes et rôles que je définis dans la console IAM.
Je vais maintenant me créer un utilisateur avec les permissions d'administration pour mon usage quotidien. Cet utilisateur ne pourra pas changer le mot de passe de l'utilisateur principal ni les données de facturation.
Dans la console, je clique sur le menu **Services** en haut à gauche et je cherche et sélectionne **IAM**.



Dans la console IAM, je clique sur **Utilisateurs** dans le menu de gauche.

Sur l'écran suivant, je clique **Ajouter un utilisateur**.

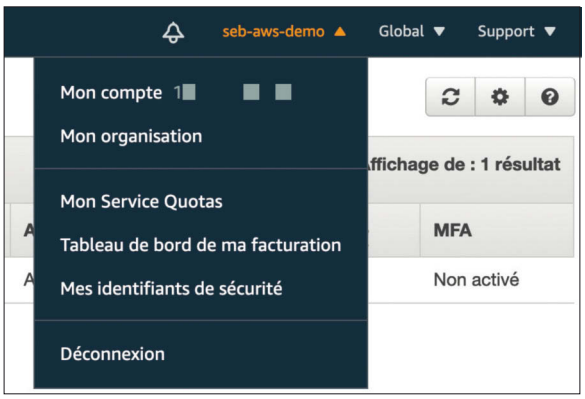
J'indique un nom d'utilisateur (« seb » dans cet exemple) et je coche les cases **Accès par programmation** et **Accès à AWS Management Console**. Ensuite je clique le bouton **Suivant : Autorisations**.



Sur l'écran d'autorisations, je définis les permissions que je donne à cet utilisateur. Ici, il s'agit de créer un utilisateur qui a les permissions d'utiliser tous les services. Je clique sur **Attacher directement les stratégies existantes** et je sélectionne la stratégie **AdministratorAccess**. Ensuite, je clique **Suivant : Balises**.
Pour cet exemple, je choisis de ne pas ajouter de balises à cet utilisateur et je clique directement sur **Suivant**. Le dernier écran me propose de relire mes choix. Si je suis d'accord, je clique **Créer cet utilisateur**.

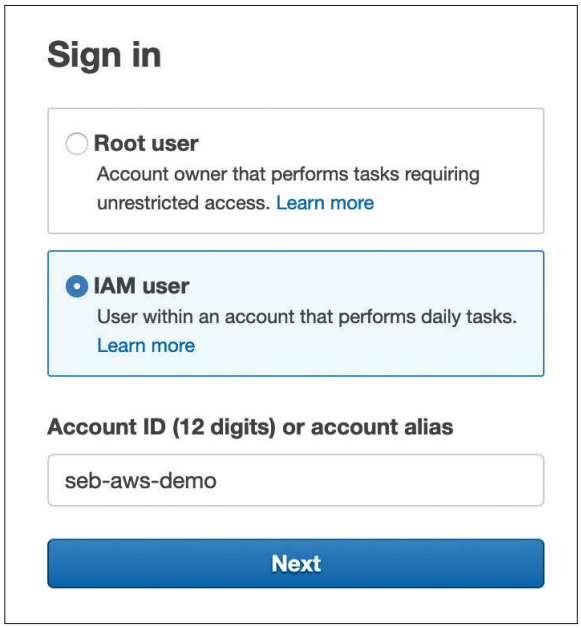
Une fois l'utilisateur créé, je reçois trois informations importantes :
- La clé d'accès ;
- La clé d'accès secrète ;
- Le mot de passe initial, pour se connecter en console.

La clé d'accès et la clé secrète permettent de s'identifier, via la ligne de commande ou vos propres scripts, comme étant cet utilisateur.
C'EST LE SEUL ENDROIT OÙ VOUS TROUVEREZ LA CLÉ SECRÈTE. Prenez-en note et conservez-la avec toutes les précautions que vous prenez pour un mot de passe. La combinaison de la clé d'accès et la clé secrète permet d'accéder à votre compte avec toutes les permissions ! Au cas où je perds la clé secrète, je peux retourner dans la console IAM pour l'invalider et en générer une nouvelle.
Je conseille fortement d'activer le MFA pour cet utilisateur. Là où il est recommandé d'utiliser un jeton physique pour le compte principal, vous pouvez utiliser une application OTP pour les utilisateurs IAM.
Une fois ces informations notées et enregistrées (par exemple dans un gestionnaire de mots de passe), je me déconnecte de la console en cliquant le menu en haut à droite, puis en choisissant **Déconnexion**.



Enfin, je vérifie que je peux me connecter avec l'utilisateur IAM que je viens de créer (« seb » dans mon exemple) et je change son mot de passe.

Pour ce faire, je clique **Se reconnecter** au milieu de la page, ou **Mon compte** en haut à droite.



Cette fois-ci je sélectionne IAM User et je rentre le numéro de mon compte AWS que je viens de créer. Je peux aussi utiliser l'alias plutôt que le numéro de compte. Je clique sur **Next**.

Je rentre le nom d'utilisateur que j'ai défini dans IAM et le mot de passe généré par la console AWS.
SI VOUS RECEVEZ UN MESSAGE INDIQUANT UN MOT DE PASSE INCORRECT, ESSAYEZ D'UTILISER LE NUMERO DE COMPTE PLUTÔT QUE L'ALIAS. Il peut y avoir parfois quelques heures avant qu'un alias ne soit actif.
Ensuite je me retrouve dans la console et je suis prêt à démarrer mon utilisation des services AWS.

ÉTAPE 5 : installer la ligne de commande

Tous les services AWS sont disponibles via la console web, mais peuvent aussi être utilisés depuis des applications, des scripts ou depuis la ligne de commande AWS.
Indépendamment de tout langage de programmation, la ligne de commande permet d'invoquer la totalité des APIs AWS, ainsi que d'automatiser des actions, en les assemblant en scripts shell qui vont vous servir à exécuter les séquences de tâches répétitives.
La ligne de commande est disponible pour Windows, macOS, Linux et Docker. Dans cette section, vous trouverez les instructions pour configurer la ligne de commande.
Le détail des instructions d'installation est disponible en anglais sur https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html
Sur macOS, le plus simple est d'utiliser `brew` et de taper `brew install awscli` dans mon terminal. Après l'installation, je teste la ligne de commande de la manière suivante :

```
→  ~ aws --version
aws-cli/2.1.7 Python/3.9.0 Darwin/19.6.0 source/x86_64 prompt/off
→  ~
```

Les numéros de versions que vous aurez seront peut-être plus récents. La dernière étape consiste à entrer ma clé d'accès et clé secrète générée par IAM ci-devant. Je tape `aws configure` et me laisse guider par les indications.

```
→  ~ aws configure
AWS Access Key ID [None]: AKIAR7PVOA3LLLINOEV2
AWS Secret Access Key [None]: Z[redacted]AR
Default region name [None]: eu-west-3
Default output format [None]:
→  ~
```

Dans cet exemple, je choisis d'utiliser la région AWS Europe (Paris) comme région par défaut (*eu-west-3*). Cette valeur ne sera utilisée que lorsque je ne donne pas de région en paramètre à la ligne de commande, avec le paramètre *--region*.
La ligne de commande stocke vos paramètres et vos secrets dans deux fichiers sous *~/.aws*. Pensez à sécuriser et conserver une copie de ces fichiers.

Installer un SDK AWS

Vous pouvez utiliser une variété de SDKs pour intégrer les API AWS dans vos applications : C++, Go, Java, .NET, JavaScript, Node.js, PHP, Python et Ruby. Vous trouverez plus d'informations sur https://aws.amazon.com/fr/tools/. La documentation est dans la catégorie « SDKs & Toolkits » sur https://docs.aws.amazon.com.
Par exemple, voyons comment installer le SDK Python, aussi appelé boto3 (https://github.com/boto/boto3).
En supposant que vous avez déjà installé Python et pip, il suffit d'une commande pour installer boto3 :

```
$ pip install boto3
```

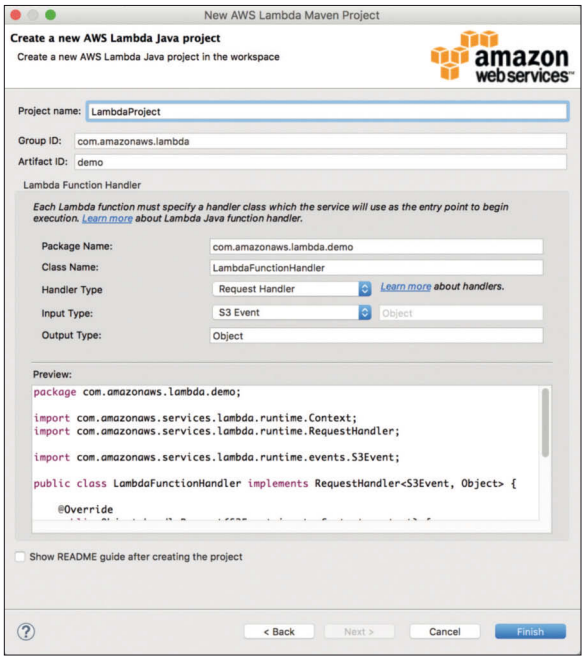
Si vos secrets AWS sont bien configurés dans *~/.aws*, vous pouvez directement utiliser les API AWS de votre choix. Par exemple, vous pouvez afficher tous vos compartiments S3 ainsi :

```
import boto3
>>> s3 = boto3.resource('s3')
>>> for bucket in s3.buckets.all():
    print(bucket.name)
```

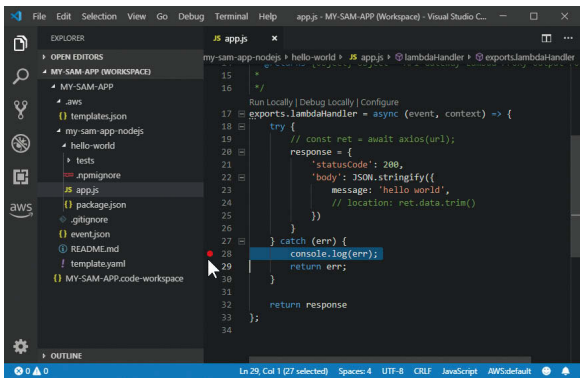
C'est tout !

Installer une extension pour votre environnement de développement

AWS propose des extensions pour les environnements les plus populaires : Eclipse, IntelliJ, PyCharm, Visual Studio, Visual Studio Code, Azure DevOps et Rider. Vous trouverez les liens et les instructions d'installation sur https://aws.amazon.com/fr/tools/.
Ces extensions vous permettent de créer, coder, debugger et déployer vos applications sur AWS, directement depuis votre environnement de développement. Selon l'environnement, vous pouvez également visualiser les ressources AWS en cours d'exécution dans votre compte AWS, les arrêter, etc.
Par exemple, vous pouvez facilement créer une nouvelle fonction Lambda dans Eclipse.



Vous pouvez aussi développer et déployer des applications serverless depuis Visual Studio Code, grâce à l'intégration du Serverless Application Model (https://aws.amazon.com/serverless/sam/).



Exécuter vos premiers exemples

Afin de démarrer rapidement, vous trouverez des exemples dans la documentation des différents SDK, par exemple :
Python : https://boto3.amazonaws.com/v1/documentation/api/latest/guide/examples.html
Node.js : https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/s3-node-examples.html
Java : https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/examples.html
PHP : https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/examples_index.html
Vous êtes maintenant prêt(e) à suivre les articles et tutoriels de ce hors-série.

Ressources complémentaires

Comment créer son compte AWS ? : https://www.youtube.com/watch?v=TjKu5iwr3x8
3 étapes pour sécuriser son compte AWS. https://www.youtube.com/watch?v=Jk3bJODIVf8
Créer un utilisateur IAM. : https://www.youtube.com/watch?v=ZMHIBza1I1A
Installer la ligne de commande AWS. : https://www.youtube.com/watch?v=V63ZzExqQIw
Comment utiliser la ligne de commande AWS ? : https://www.youtube.com/watch?v=c7BnfiMSB4Y
50 vidéos pour bien démarrer sur AWS. : https://stormacq.com/2020/08/31/bien-demarrer.html
Le blog AWS News : https://aws.amazon.com/blogs/aws/
Le Podcast AWS en français : https://aws.amazon.com/fr/blogs/france/podcasts/
Le channel YouTube en français : https://www.youtube.com/sebsto
Le podcast Les Technos : https://lestechnos.be

Julien Simon

En tant qu'évangéliste mondial de l'IA et de l'apprentissage automatique, Julien s'attache à aider les développeurs et les entreprises à donner vie à leurs idées. Il prend souvent la parole lors de conférences, et écrit sur le blog AWS. Avant de rejoindre AWS, Julien a occupé pendant 10 ans des postes de CTO/VP Engineering dans des startups Web de haut niveau.

Un panorama des instances GPU disponibles sur AWS

De la reconnaissance de la parole à l'entraînement d'assistants virtuels et de voitures autonomes, les spécialistes des données font face à des défis de plus en plus complexes avec l'IA. La résolution de ce genre de problèmes nécessite des modèles d'apprentissage profond qui prennent beaucoup de temps à entraîner.

Les instances Amazon EC2 basées sur un processeur graphique permettent d'accéder aux GPU NVIDIA avec des milliers de cœurs de calcul. Vous pouvez utiliser ces instances pour accélérer les applications scientifiques, d'ingénierie et de rendu en tirant parti des infrastructures de calcul parallèle CUDA ou OpenCL (Open Computing Language). Vous pouvez également les utiliser pour des applications graphiques, y compris le streaming de jeux, le streaming d'applications 3D et d'autres charges de travail graphiques. Et bien sûr, vous pouvez les utiliser pour entraîner vos modèles de machine learning. Examinons les différentes familles d'instances disponibles sur AWS, et leurs cas d'utilisation privilégiés.

Instances G4

Les instances EC2 G4 sont les instances GPU les plus rentables et les plus polyvalentes du marché pour le déploiement de modèles d'apprentissage automatique tels que la classification d'images, la détection d'objets et la reconnaissance vocale, et pour les applications graphiques gourmandes telles que les stations de travail graphiques distantes, la diffusion en continu de jeux et le rendu graphique. Les instances G4 sont disponibles avec un choix de GPU NVIDIA (G4dn) ou de GPU AMD (G4ad).

Instances G4dn

Les instances G4dn sont dotées de GPU NVIDIA T4 et de processeurs Intel Cascade Lake personnalisés. Elles sont optimisées pour l'inférence et l'entraînement à petite échelle.

Ces instances G4dn sont équipées de GPU NVIDIA T4 qui offrent un débit à faible latence jusqu'à 40 fois supérieur à celui des processeurs, ce qui permet de traiter plus de demandes en temps réel, et d'optimiser le coût de votre infrastructure d'inférence.

De plus, les instances G4dn offrent jusqu'à 65 Tflops de performances FP16 et constituent une solution convaincante pour les petits travaux d'entraînement.

Instances G4ad

Les instances G4ad, optimisées par les GPU AMD Radeon Pro V520, offrent les meilleures performances pour les applications graphiques gourmandes. Ces instances améliorent jusqu'à 45 % le rapport prix/performance par rapport aux instances G4dn, qui étaient déjà les instances les moins coûteuses dans le cloud. Elles sont idéales pour les applications graphiques telles que les stations de travail graphiques distantes, le streaming de jeux et le rendu qui exploitent des API standard comme OpenGL, DirectX et Vulkan. Ces instances fournissent jusqu'à 4 processeurs graphiques AMD Radeon Pro V520, 64 vCPU, réseau 25 Gbit/s et 2,4 To de stockage SSD local basé sur NVME.

Les instances G4ad permettent aux clients de configurer des stations de travail virtuelles avec des capacités de simulation, de rendu et de conception hautes performances en quelques minutes, ce qui permet aux clients d'évoluer rapidement. Les clients peuvent utiliser le logiciel AMD Radeon Pro pour entreprise et le protocole d'affichage à distance haute performance, NICE DCV, avec des instances G4ad sans frais supplémentaires pour gérer leurs environnements de postes de travail virtuels avec la prise en charge d'un maximum de deux moniteurs 4k par GPU.

Instances P3

Les instances P3 EC2 offrent des fonctionnalités de calcul haute performance dans le cloud avec jusqu'à 8 GPU NVIDIA V100 et un débit réseau pouvant atteindre 100 Gb/s pour les applications d'apprentissage automatique et HPC (calcul haute performance). Ces instances offrent jusqu'à 1 pétaflop de performances de précision mixte par instance pour accélérer significativement les applications d'apprentissage automatique et de calcul haute performance. Il a été démontré que les instances P3 Amazon EC2 réduisent les temps d'entraînement de jours en minutes, et multiplient par 3 ou 4 le nombre de simulations effectuées pour le calcul haute performance.

Avec jusqu'à 4 fois plus de bande passante réseau que les instances P3.16xlarge, les instances P3dn.24xlarge Amazon EC2 sont les tout derniers membres de la famille P3, et sont optimisées pour les applications de machine learning distribué et de HPC. Ces instances offrent un débit réseau pouvant atteindre 100 Gbit/s, 96 vCPU personnalisés Intel Xeon Scalable (Skylake), 8 GPU NVIDIA V100 dotés chacun de 32 Go de mémoire et 1,8 To de stockage local SSD basé sur NVMe.

Les instances P3dn.24xlarge sont également compatibles avec Elastic Fabric Adapter (EFA) qui accélère les applications de machine learning distribuées utilisant la NVIDIA Collective Communications Library (NCCL). L'EFA peut évoluer vers des milliers de GPU, améliorant considérablement le débit et la scalabilité des modèles de deep learning, ce qui accélère les résultats.

Vous pouvez utiliser plusieurs instances P3 Amazon EC2 avec un débit réseau pouvant atteindre 100 Gbit/s afin de former rapidement des modèles de machine learning. Un débit réseau plus élevé permet aux développeurs d'éliminer les goulots d'étranglement du transfert de données et de faire efficacement monter en charge leurs tâches d'entraînement sur plusieurs instances P3. Des clients ont pu entraîner ResNet-50, un modèle de classification d'image courant en seulement 18 minutes à l'aide de 16 instances P3. Ce niveau de performance était auparavant inaccessible pour la plupart des clients ML, car de gros investissements étaient nécessaires pour déployer des clusters GPU sur site. Avec les instances P3 et leur disponibilité à la demande, ce niveau de performance est désormais accessible à tous les développeurs et ingénieurs en machine learning.

Instances P4d

Les instances EC2 P4d offrent les performances les plus élevées pour la formation machine learning et les applications de calcul hautes performances dans le cloud. Les instances P4d sont alimentées par les derniers GPU NVIDIA A100 et offrent une mise en réseau à haut débit et à faible latence. Ces instances sont les premières dans le cloud à prendre en charge la mise en réseau d'instances de 400 Gbit/s. Les instances P4d offrent une performance d'entraînement 2,5 fois supérieure par rapport aux instances P3 et P3dn des générations précédentes.

Les instances Amazon EC2 P4d sont déployées dans des clusters hyperscale appelés EC2 UltraClusters. Chaque UltraCluster EC2 est l'un des superordinateurs les plus puissants au monde, permettant aux clients d'entraîner des modèles distribués complexes. Les clients peuvent facilement évoluer de quelques GPU NVIDIA A100 à des milliers en fonction de leurs besoins.

Les chercheurs, les spécialistes des données et les développeurs peuvent tirer parti des instances P4d pour des cas d'utilisation tels que le traitement du langage naturel, la détection et la classification d'objets et les moteurs de recommandation, ainsi que pour exécuter des applications HPC telles que la découverte pharmaceutique, l'analyse sismique et la modélisation financière. Contrairement aux systèmes locaux, les clients peuvent accéder à une capacité de calcul et de stockage quasi illimitée, mettre à l'échelle leur infrastructure en fonction des besoins de l'entreprise et effectuer en quelques minutes une formation ML multi-nœuds ou une application HPC distribuée étroitement couplée, sans coûts de configuration ou de maintenance.

Pour résumer :

Famille	Type de GPU	Principaux cas d'usage
G4dn	NVIDIA T4	Inférence ML, training ML à petite échelle
G4ad	AMD Radeon Pro V520	Applications graphiques, jeux
P3	NVIDIA V100	Machine Learning et Deep Learning
P3dn	NVIDIA V100	Deep Learning et HPC distribué
P4d	NVIDIA A100	Deep Learning et HPC distribué à grande échelle

Pour aller plus loin :

https://aws.amazon.com/ec2/instance-types/g4/
https://aws.amazon.com/ec2/instance-types/p3/
https://aws.amazon.com/ec2/instance-types/p4/
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/accelerated-computing-instances.html
https://aws.amazon.com/machine-learning/accelerate-machine-learning-P3/

L'OM : du foot et du cloud !

L'Olympique de Marseille est un des clubs de football les plus connus. Au-delà du sport, un club sportif est la combinaison de nombreux éléments : supporters, administration, budget, les boutiques, les objets dérivés. C'est une machine à plusieurs centaines de personnes pour faire tourner le club au quotidien : répondre aux attentes des supporters, vendre la marque OM. Ce sont des éléments fondamentaux pour les finances d'un club. Et l'OM n'a pas hésité à utiliser le cloud pour y parvenir.

Frédéric Cozic (Chief Technology & Innovation Officer) et Benjamin Prato (Head of cloud & IT à l'Olympique de Marseille) sur l'utilisation de services AWS dans l'informatique du club.

En novembre 2019, le club annonçait une collaboration avec AWS pour utiliser la puissance du cloud computing. Pourquoi le cloud alors qu'il semble bien loin du sport ?
L'Olympique de Marseille a lancé en janvier 2019 un projet de refonte complète de sa plateforme technologique. Alors que les briques métiers étaient jusqu'alors fragmentées par solutions, cette nouvelle architecture devait permettre de s'articuler autour d'une plateforme cloud solide qui allait centraliser la donnée, quel que soit son type : business, connaissance client, sportive, médicale, etc. Près de 2 ans après, la plateforme OM cloud est là, construite à 100% sur un environnement AWS, et agrégeant l'ensemble des données du Club dans un Datalake. En capitalisant sur cette plateforme, l'entreprise peut plus facilement innover, tester de nouvelles idées et gagner ainsi en agilité.

En quoi le machine learning et/ou le deep learning vous aide au quotidien ?
L'utilisation de la donnée est désormais au cœur de l'ensemble des prises de décisions du club. Elle est employée pour soutenir un choix apportant un complément d'information ou une recommandation. Après avoir intégré plusieurs années de données historiques, nous utilisons le machine Learning au travers d'algorithmes prédictifs, par exemple pour prédire les blessures des joueurs, ou encore optimiser le remplissage du stade. Ceci, en réimaginant et en adaptant le concept du Yield Management, communément utilisé dans l'aérien ou l'hôtellerie. Là encore, ces nouveaux outils viennent en appui des équipes afin de gagner en efficacité dans la prise de décision finale.

Pourquoi votre choix s'est porté sur AWS ? Quels services utilisez-vous ?
Dans le cadre de ce projet de refonte technologique globale et avec la volonté d'en construire les fondations dans le cloud, nous avons beaucoup échangé avec les équipes AWS. De premiers essais concrets à une roadmap complète, nous nous sommes très vite alignés sur une collaboration étroite au travers de laquelle l'OM accède à un ensemble de briques techniques les plus avancées dans de nombreux domaines. De plus, AWS participe de ce fait à un projet ambitieux et innovant dans l'industrie du sport européen. Cet alignement nous a permis d'élever mutuellement l'ambition du projet dans son ensemble.

Plus concrètement, nous utilisons un panel assez large de services AWS. Pour le stockage de nos données, nous utilisons principalement du S3 et plus récemment du Redshift en complément, pour les données plus structurées, ainsi que du RDS pour nos besoins OLTP. Nous traitons ensuite ces données en fonction du besoin avec du Glue / Spark, du Athena directement depuis notre stockage S3, ou avec Redshift en SQL ou de la Lambda. Sur la partie compute nous utilisons au maximum des technologies Serverless comme Lambda et du ECS / Fargate quand nous atteignons les limites, et que nous voulons faire tourner des applications plus classiques. Nous utilisons également Step Functions pour orchestrer certains pipelines. Sur la partie messaging et queues, nous utilisons SNS, SQS et Kinesis en fonction du besoin. Nous utilisons également les briques de compute plus classiques comme EC2 et ses services connexes. Pour finir, nous utilisons des services managés comme AWS Transfert, API Gateway ainsi que Sagemaker sur la partie IA/ML.

Comment s'architecture votre approche ML : les données en entrée, les traitements, le stockage et le résultat ?
Nous avons différents pipelines de données, certains en mode batch et d'autres en streaming. Nous ingérons les données principalement via de l'API Gateway ou des dépôts de fichier plus classiques en fonction de nos options. Nos pipelines Data sont majoritairement « datalake-centric ». Pour les jobs en streaming, les fichiers transitent dans différentes zones de notre datalake, déclenchant des événements traités par différentes Lambda pour ce qui concerne le streaming, et qui sont en partie orchestrés par Step Functions. Sur la partie Batch, nous préparons les données via des jobs Glue et nous réalisons les traitements via du Fargate ou différents services SageMaker en fonction du besoin. La plupart des données en sortie de nos algorithmes sont réécrites sur S3 et rechargées dans des bases de données en fonction de notre use case.

Quelles sont les principales difficultés de votre projet ?
La multitude de services et leur richesse donnent en quelque sorte des super-pouvoirs à un développeur. Cet avantage peut également être à double tranchant, car il est parfois difficile d'arbitrer entre plusieurs services ou entre plusieurs façons de faire. Nous essayons aujourd'hui de suivre la mouvance « Lake-House » dans notre architecture Data. Ilest d'ailleurs parfois difficile ou très challengeant de trouver le bon flux de données. On se pose souvent la question, ou mettre telle donnée ? Quelle est la solution la plus efficiente en termes de coût et de performance ? La solution que nous mettons en place est-elle pérenne dans le temps ?
Au-delà des technologies AWS, ce monde évolue tellement vite et nous offre aujourd'hui tellement d'options et d'outils qu'il faut réfléchir plus que jamais à sa stratégie technique, et toujours penser long terme afin de garantir la pérennité des projets

Quels sont les langages / SDK / outils utilisés ? Avez-vous fait monter en compétence vos développeurs ? Comment faites-vous l'interfaçage entre les services AWS et vos propres apps / back office ?
Nous utilisons principalement 2/3 langages dans le développement applicatif : Python, Java et Scala. En fonction du besoin en performance, du tooling pour répondre à la problématique et de la complexité applicative, nous faisons le choix de l'un ou l'autre. Nous faisons également du R sur la partie analyse de données et certains traitements data science. Nous utilisons les SDK AWS Java et Boto3, Sagemaker côté Python. Également des frameworks applicatifs plus classiques comme Django, Flask ou encore Spring. Côté Data science ce sont des outils classiques, du Pytorch, scikit-learn ou encore du TensorFlow, la liste est non exhaustive. L'interface avec nos applications externes se fait principalement via de l'API ou de l'export de fichiers par lots en utilisant API Gateway ou AWS Transfer. Nos ingénieurs et Data Scientists suivent régulièrement les Immersion days et d'autres ressources mises à dispositions par AWS. Nous essayons de suivre les certifications en proposant les formations de la plateforme CloudGuru. La formation passant énormément par la pratique, surtout à la vitesse à laquelle les services évoluent nous encourageons les utilisateurs et développeurs de la plateforme à tester et expérimenter sur des comptes dédiés.
Et la quasi-totalité des services ML est hébergée sur notre plateforme AWS sur des services comme SageMaker, ECS ou encore Lambda. L'ensemble de nos nouveaux projets sont hébergés sur AWS.

Passer la barrière de la langue

Quoi de mieux que d'accueillir un client dans la langue de son choix ? De le/la laisser s'exprimer dans sa langue, de comprendre son message et de lui répondre dans la même langue. Imaginez un chatbot d'aide, le premier tri des questions dans un centre d'appel, des quizz en tous genres et bien d'autres usages. Que ça soit en audio ou par écrit, vos applications peuvent tirer parti des services de compréhension, traduction et génération de texte en langages naturels proposés par AWS.

Sébastien Stormacq

Principal Developer Advocate

Seb écrit du code depuis qu'il a touché pour la première fois à un Commodore 64 au milieu des années 80. Il inspire les développeurs pour utiliser le cloud AWS, en utilisant son mélange secret de passion, d'enthousiasme, d'attention aux clients, de curiosité et de créativité. Ses intérêts portent sur les architectures logicielles, les outils de développement et l'informatique mobile, iOS en particulier.

Dans cet article, je vais vous montrer comment reconnaître la langue des messages envoyés par les clients, comprendre les principaux éléments, traduire dans une autre langue et générer une réponse vocale.

Pour simplifier notre article, je pars de textes, mais vous pouvez utiliser des fichiers audio. La capture et l'envoi de fichier audio, en temps réel ou non, vous est laissée en guise d'exercice. Pour ne pas favoriser un langage de programmation plutôt qu'un autre, je vais user et abuser de la ligne de commande AWS. Tout ce qui y est démontré dans cet article peut se faire dans le langage de votre choix, via l'utilisation d'un de nos SDK (https://aws.amazon.com/tools/) ou depuis la console.

Détecter la langue du texte

Quand mon application reçoit du texte (ou de l'audio), la première chose à faire est de déterminer la langue. Si je reçois de l'audio, je lance une tâche de transcription pour convertir le flux audio en texte. Amazon Transcribe s'occupe de la transcription. https://docs.aws.amazon.com/cli/latest/reference/transcribe/start-transcription-job.html

Une fois que j'ai du texte, Amazon Comprehend me permet de l'analyser. Je détecte la langue dans laquelle le message est rédigé avec :

```
—> ~ aws comprehend detect-dominant-language --text "Programmez! est le meilleur magazine en français pour les développeurs"

{
  "Languages": [
    {
      "LanguageCode": "fr",
      "Score": 0.9964491724967957
    }
  ]
}
```

Le JSON retourné par l'API et la ligne de commande me dit que le texte est rédigé en français, avec une probabilité de 99.6%. Dans la console, ça donne ceci : **figure 1**

Ensuite, j'essaye de comprendre de quoi mon client parle :

```
—> ~ aws comprehend detect-entities --text "'Programmez' est le meilleur magazine en français pour les développeurs" --language-code fr
{
  "Entities": [
    {
      "Score": 0.9720069766044617,
      "Type": "TITLE",
      "Text": "Programmez",
      "BeginOffset": 1,
      "EndOffset": 11
    },
    {
      "Score": 0.9653366804122925,
      "Type": "OTHER",
      "Text": "français",
      "BeginOffset": 41,
      "EndOffset": 49
    }
  ]
}
```

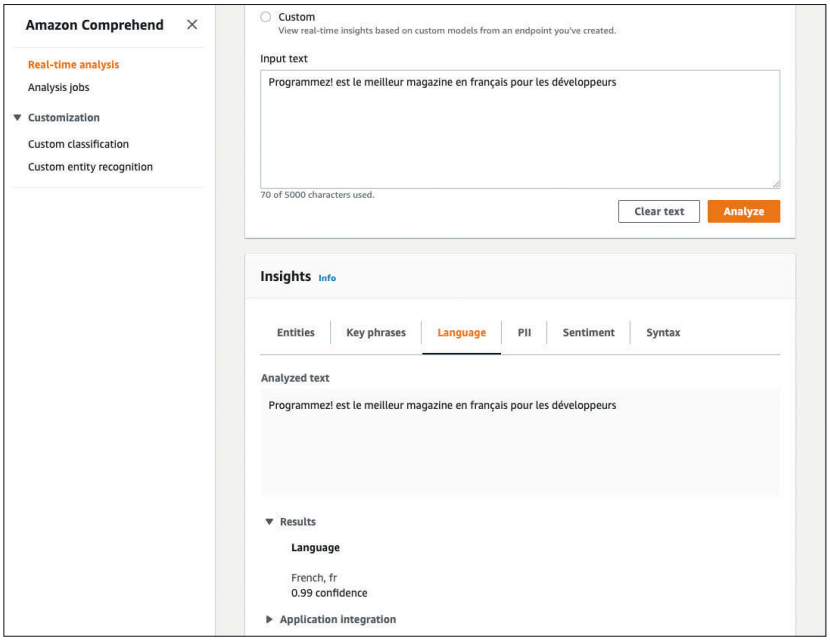
Ici, le JSON nous dit que ce message parle de quelque chose qui s'appelle « Programmez » et « français »

Analyser le sentiment du texte

Je peux aussi demander à Amazon Comprehend si le message est plutôt positif ou négatif :

```
—> ~ aws comprehend detect-sentiment --text "Programmez est le meilleur magazine en français pour les développeurs" --language-code fr
{
```

Figure 1



```
  "Sentiment": "POSITIVE",
  "SentimentScore": {
    "Positive": 0.9956327080726624,
    "Negative": 0.0009476951090618968,
    "Neutral": 0.0034117382019758224,
    "Mixed": 7.804364031471778e-06
  }
}
```

Figure 2

Figure 3

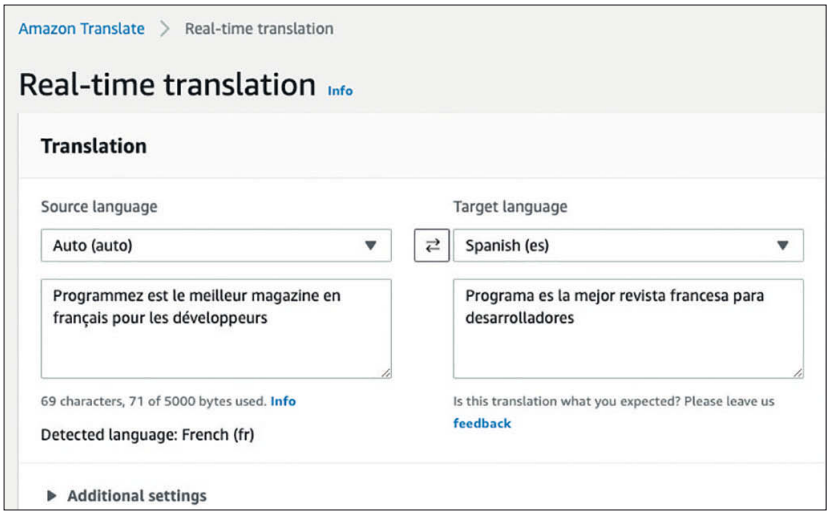
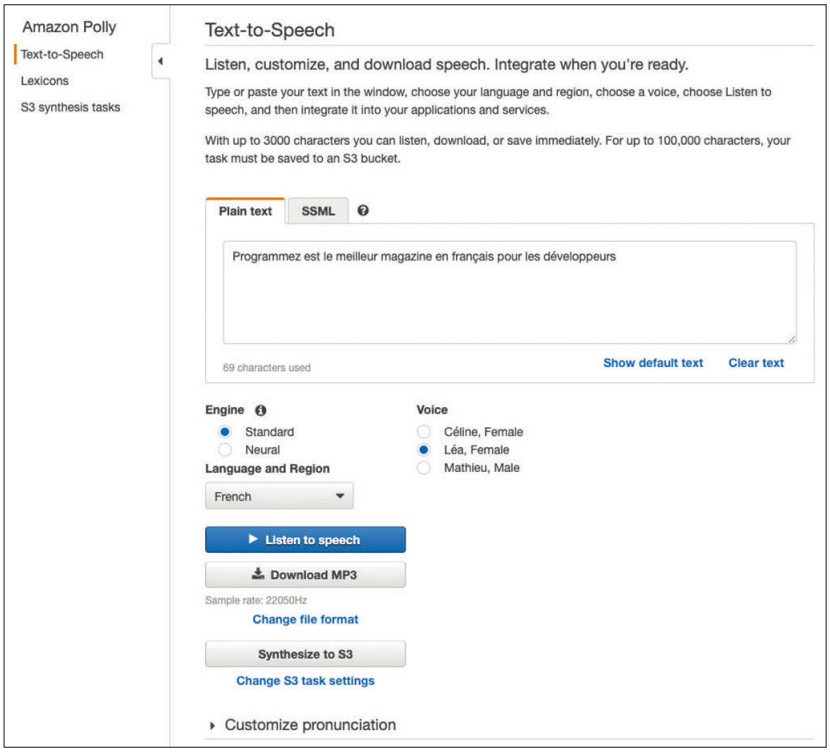


Figure 4



Dans ce cas, Comprehend pense que le message est positif (il en est sûr 99.56%).
Dans la console, chaque onglet vous donne une des informations ci-dessus. **Figure 2**

Traduire le texte

Je peux traduire le message en utilisant Amazon Translate :

```
—> ~ aws translate translate-text --text "Programmez est le meilleur magazine en français pour les développeurs" --source-language-code fr --target-language-code es
{
  "TranslatedText": "Programa es la mejor revista francesa para desarrolladores",
  "SourceLanguageCode": "fr",
  "TargetLanguageCode": "es"
}
```

Amazon Translate peut traduire des textes dans 71 paires de langues. Il est possible de lui donner un dictionnaire personnalisé pour qu'il reconnaisse des marques, des modèles ou tout autre vocabulaire spécifique à votre métier ou celui de vos clients.
Je peux faire la même chose avec la console. **Figure 3**

Générer un message sonore à partir d'un texte

Finalement, je synthétise la réponse en utilisant une des nombreuses voix de Amazon Polly, dans l'une des 29 langues supportées. Pour le français, j'ai le choix entre les voix de Céline, Léa ou Mathieu.

```
—> ~ aws polly synthesize-speech --output-format mp3 --text "Programmez est le meilleur magazine en français pour les développeurs" --voice-id Lea demo.mp3 && afplay demo.mp3
{
  "ContentType": "audio/mpeg",
  "RequestCharacters": "69"
}
```

Ou avec la console : **figure 4**

Voilà pour ce tour rapide de Amazon Transcribe, Amazon Comprehend, Amazon Translate et Amazon Polly.

Pour terminer

Je rappelle que j'ai utilisé la ligne de commande pour cet article, mais vous pouvez évidemment faire de même dans vos applications web ou mobiles. Si vous développez une application web ou mobile, le moyen le plus simple pour intégrer ces services d'analyse de langage naturel est de passer par la ligne de commande et les librairies open-source proposées par AWS Amplify (https://aws.amazon.com/amplify/). En particulier, essayez la commande *amplify add prediction*. Les détails sont sur https://docs.amplify.aws/lib/predictions/getting-started. Amplify supporte les applications web écrites en Javascript (comme celles développées avec React , Angulmar, Vue ou Ionic), iOS, Android et Flutter.

Simplifier l'analyse d'images et vidéos dans des applications JavaScript avec Amazon Rekognition

Davide Gallitelli
Solutions Architect au sein des équipes AWS France et assistant les clients français dans l'adoption des solutions basées sur l'Intelligence Artificielle & le Machine Learning.

Au cours des décennies, les informaticiens ont adopté de nombreuses approches différentes pour résoudre le défi de reproduire la capacité de vision dans des ordinateurs et appareils mobiles.

Aujourd'hui, un large consensus s'est dégagé sur le fait que la meilleure façon de s'attaquer à ce problème est de passer par le Deep Learning. Essentiellement, vous présentez le réseau d'apprentissage avec un large éventail d'exemples étiquetés (« ceci est un chien », « c'est un oiseau », etc.) afin qu'il puisse corréler les caractéristiques de l'image avec les étiquettes. Cette phase est coûteuse sur le plan informatique en raison de la taille et de la nature multicouche des réseaux neuronaux. Une fois la phase de formation terminée, l'évaluation des nouvelles images par rapport au réseau est beaucoup plus facile. Les résultats sont traditionnellement exprimés en niveaux de confiance (0 à 100 %) plutôt que sous forme de faits durs et froids. Cela vous permet de décider à quel point la précision est appropriée pour vos applications.

Une introduction à Amazon Rekognition

Amazon Rekognition est un service simplifiant l'ajout de l'analyse des images à vos applications.
Propulsé par le Deep Learning et construit par notre équipe Computer Vision au fil des années, Amazon Rekognition comprend des fonctionnalités permettant de détecter objets et scènes, reconnaissance, analyse et comparaison faciale, modération de contenu en détectant images inappropriées, reconnaissance de célébrités, reconnaissance de texte dans l'image et détection des équipements de protection individuelle (EPI). Ce service analyse désormais des milliards d'images par jour, et il est maintenant disponible et prêt à être intégré dans vos applications.
Pour commencer, connectez-vous simplement à la console Amazon Rekognition pour essayer le service avec des exemples d'images ou avec vos propres images. Si vous envoyez par exemple une image d'un plat de pâtes, vous obtiendrez : **figure 1**

NOTE : *pour les sections suivantes, on suppose que vous avez créé un compte AWS [1] et que vous avez configuré la CLI AWS [2]. Pour plus d'information, vous pouvez aussi suivre la chaîne YouTube Amazon Web Services France, qui propose une série qui s'appelle "Bien démarrer sur AWS" [3].*

Comme tout service et fonctionnalité de la plateforme AWS, Rekognition met à disposition des développeur : une CLI et des API dans les langages de programmation de votre préférence, notamment C++, Go, Java, JavaScript, .NET, Node.js, PHP, Python et Ruby. Pour obtenir le même résultat montré avant en utilisant la CLI, vous n'avez besoin que d'un seul call :

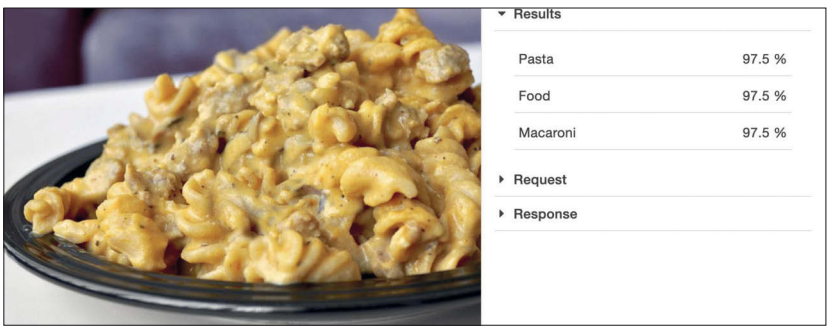
```
aws rekognition detect-labels --image-bytes fileb://pasta.jpg
```

Rekognition est très pratique pour rajouter de l'intelligence aux applications - mobile et web. Supposons, par exemple, que nous avons une API basée sur du Node.js. Nous pouvons intégrer le SDK JavaScript au code existant pour faire le même appel au service Rekognition :

```
var params = {
 Image: { /* nécessaire */
  Bytes: Buffer.from('...') || 'STRING_VALUE' /* Strings sont encodés en
 Base-64 encodé pour vous*/,
 },
 MaxLabels: 'NUMBER_VALUE',
 MinConfidence: 'NUMBER_VALUE'
};
rekognition.detectLabels(params, function(err, data) {
 if (err) console.log(err, err.stack); // erreur
 else     console.log(data);          // ok!
});
```

Dans l'exemple proposé, on utilise une image locale lue avec la méthode Buffer; néanmoins, c'est aussi possible d'utiliser une image stockée sur S3 et de changer la valeur Bytes avec un dictionnaire S30bject. Pour voir comment faire, regardez la documentation du SDK JavaScript pour Rekognition [4]. Le résultat est en format JSON :

Figure 1



```
{
  "Labels": [
    {
      "Name": "Pasta",
      "Confidence": 97.57051086425781,
      "Instances": [
        {
          "BoundingBox": {...},
          "Confidence": 97.57051086425781
        }
      ],
      "Parents": [
        {
          "Name": "Food"
        }
      ]
    }, [...]]
}
```


Comment customiser Rekognition

Les clients ont souvent besoin d'analyser leurs images pour trouver des objets qui sont propres à leurs besoins métiers. Dans la plupart des cas, il est difficile de le faire car ces modèles nécessitent des milliers d'images étiquetées et une expertise en apprentissage profond. Dans d'autres cas, il peut s'agir d'un seul objet, comme l'identification du logo de l'entreprise, la découverte d'un défaut industriel ou agricole particulier ou la localisation d'un événement spécifique comme un ouragan dans des images satellites.

Avec Rekognition Custom Labels, qui s'appuie sur les fonctionnalités existantes d'Amazon Rekognition, vous pouvez identifier les objets et les scènes dans des images qui sont spécifiques à vos besoins commerciaux. Par exemple, vous pouvez trouver votre logo dans les publications sur les réseaux sociaux, identifier vos produits sur les étagères des magasins, classer les pièces de machines dans une chaîne d'assemblage, distinguer les plantes saines et infectées, ou détecter des personnages animés dans les vidéos.

Pour tester les fonctionnalités de Rekognition Custom Labels, on va se mettre dans la peau d'un producteur de pâtes ou d'un food blogger. Il souhaite mettre en place un système qui lui permet automatiquement de rajouter un tag dans sa page web concernant la typologie de pâtes qui est présente dans sa photo. Pour cette démo, on n'a pas un dataset disponible publiquement, mais on a récupéré des photos depuis Internet pour pouvoir entraîner notre modèle de Computer Vision. Dans notre cas, on a pu récupérer 126 images, pour 3 types de pasta : fusilli, penne et spaghetti. Chacun de ces types de pâtes est un "label" ou une "class", que notre système de CV doit détecter et rajouter sous forme de tag à la page web. Étant notre objectif de détecter une seule class entre plusieurs, cette tâche prend le nom de "classification en classes multiples" ou "multi-class classification". Vous pouvez en apprendre plus sur la page Wikipédia (en anglais) [5] .

Préparation du dataset

Pour démarrer, lancez Rekognition depuis la AWS Management Console, choisissez "Custom Labels" dans le menu à gauche, et appuyez sur "Get Started"/"Commencer". La première étape sera de nommer notre projet, "pasta-classifier". Une fois la création du projet terminée, vous pouvez noter dans la page deux sections : "Models"/"Modèles" et "Datasets"/"Ensembles de données". Le cœur de notre système de classification est le modèle : il s'agit d'une version de l'algorithme de Computer Vision qui est "entraîné" à reconnaître nos objets dans le dataset. Du coup, pour pouvoir réussir notre challenge, nous allons créer un dataset, et ensuite l'utiliser pour entraîner le modèle. **Figure 2**

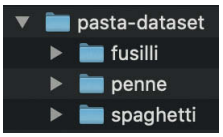
Pour ce projet, nous allons utiliser une fonctionnalité très puissante de Amazon Rekognition Custom Labels, c'est à dire d'attacher automatiquement les labels à nos images selon l'arborescence de Amazon S3. Cette étape va simplifier et accélérer énormément le temps de développement ; néanmoins, sachez que c'est aussi possible de charger une par une (ou en lot) les images sur Amazon Rekognition Custom Labels et d'attacher des labels dans la console, ou de passer par des services de classification et labellisation automatique comme SageMaker Ground Truth. Il y aura une édition spécifique de cette rubrique qui rentre dans les détails de SageMaker Ground Truth, du coup ne la perdez pas!

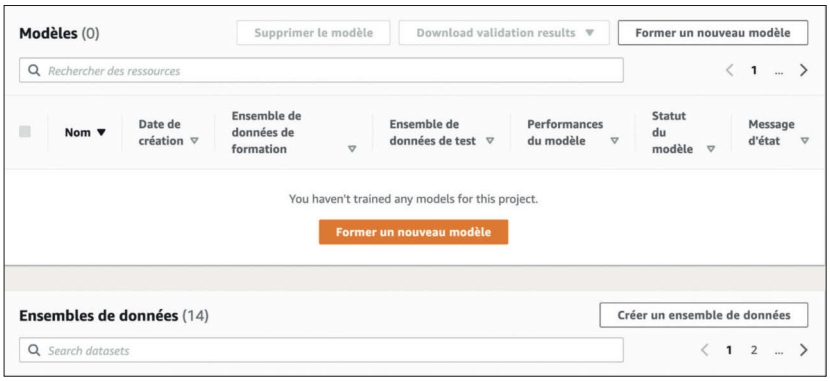
Pour pouvoir profiter de cette fonctionnalité, assurez-vous d'avoir créé l'arborescence dans votre file explorer comme dans l'image à gauche. Les noms des fichiers dans chaque répertoire ne sont pas importants, mais les noms doivent être précis car ils vont être utilisés comme "labels". Vous pouvez vous simplifier la tâche en utilisant la commande suivante :

```
mkdir pasta-dataset && cd pasta-dataset && mkdir fusilli penne spaghetti
```

Une fois les images prêtes dans les bons répertoires, c'est le moment de les charger sur Amazon S3. Vous avez plusieurs options pour faire ça, ici on va le faire depuis la console AWS. Ouvrez un autre onglet dans votre navigateur de préférence, et allez sur Amazon S3 depuis la AWS Management Console. Créez un bucket avec un nom qui soit unique, par exemple "datasets-programmez-VOTRE_NOM", et créez un répertoire "pasta-dataset". Cliquez sur le bouton "Upload", et choisissez les 3 répertoires "fusilli", "penne" et "spaghetti". Si vous ne choisissez pas le répertoire "pasta-dataset" directement, il faudra mettre à jour le chemin plus tard dans Amazon Rekognition Custom Labels.

Figure 2





Si vous avez tout bien fait, vous pouvez vérifier la configuration de votre bucket S3 dans la console ou grâce à la commande CLI suivante :

```
aws s3 ls datasets-programmez-VOTRE_NOM/pasta-dataset/
```

Avec le résultat suivant :

```
PRE fusilli/
PRE penne/
PRE spaghetti/
```

Pour permettre à Rekognition Custom Labels d'avoir accès à ce bucket, il faudra associer une *bucket policy*. Une bucket policy est la façon avec laquelle on permet aux services AWS d'avoir accès aux ressources dans S3 [6]. Revenez sur la page d'accueil du bucket créé, et cliquez sur "Autorisations"/"Permissions", et sur le bouton "Modifier"/"Edit" dans la section "Stratégie de compartiment"/"Bucket Policy". Rajoutez la policy suivante :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSRekognitionS3AclBucketRead20191011",
      "Effect": "Allow",
      "Principal": {
        "Service": "rekognition.amazonaws.com"
      },
      "Action": [
        "s3:GetBucketAcl",
        "s3:GetBucketLocation"
      ],
      "Resource": "arn:aws:s3:::datasets-programmez-VOTRE_NOM/*"
    },
    {
      "Sid": "AWSRekognitionS3GetBucket20191011",
      "Effect": "Allow",
      "Principal": {
        "Service": "rekognition.amazonaws.com"
      },
      "Action": [
        "s3:GetObject",
        "s3:GetObjectAcl",
        "s3:GetObjectVersion",
        "s3:GetObjectTagging"
      ],
      "Resource": "arn:aws:s3:::datasets-programmez-VOTRE_NOM/*"
    }
  ]
}
```

On enregistre les modifications.

Entraînement du modèle

On est enfin prêt pour passer à l'entraînement du modèle. Revenez sur la console Rekognition Custom Labels, dans le projet "pasta-classifier". Cliquez sur "Create dataset" ou "Créer un ensemble de données" et donnez un nom à votre dataset - par exemple "pasta-dataset". Choisissez "Importer des images depuis un compartiment Amazon S3", remplissez plus en bas le path vers notre dataset "datasets-programmez-VOTRE_NOM/pasta-dataset/" et activez la fonction"Étiquetage automatique". Une fois enregistrées les configurations, et sauf erreur, vous devriez avoir une console qui ressemble à ça: **figure 3**

Comme on peut le voir dans l'image, on a 126 images classées en 3 groupes : fusilli (48 images), penne (44 images) et spaghetti (34 images). C'est le moment de lancer l'entraînement du modèle. Il suffit de cliquer sur le bouton orange "Former un modèle"/"Train a model", et choisir le dataset "pasta-dataset". Vu qu'on a 126 images, plus que suffisantes pour notre projet, choisissez "Fractionner l'ensemble de données de formation"/"Split training dataset" - ça permettra de mettre de côté 20% des images dans le dataset afin de les utiliser pour vérifier les performances de notre modèle. Maintenant, vous pouvez lancer le processus d'entraînement ! **Figure 4**

Après une attente de 0,974 heures (soit environ 58 minutes), notre modèle est prêt à être utilisé. Première chose, on vérifie les résultats obtenus. Notre modèle présente de superbes résultats : 100% de précision, un F-Score et Recall de 1 (0 faux positif et faux négatif). Notre problème n'étant pas très

Figure 3

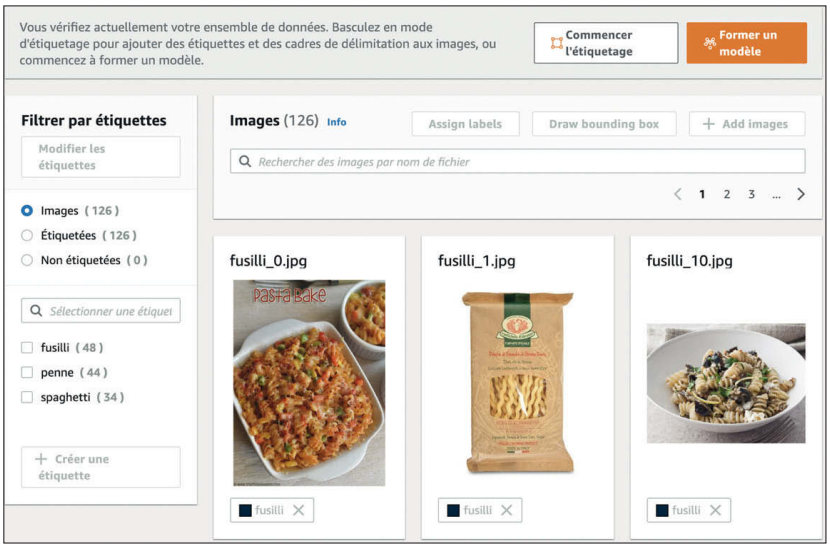
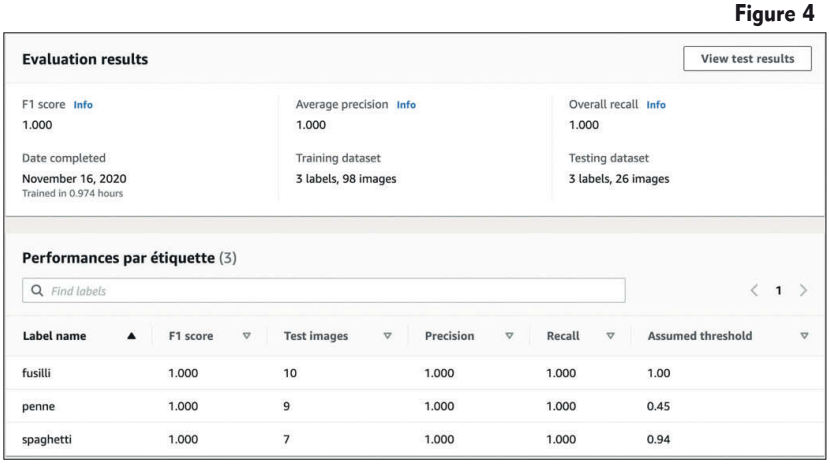


Figure 4



PRÉPARER LE DATASET À L'AIDE DE AMAZON SAGEMAKER GROUND TRUTH

Pour cet article, on a utilisé un dataset déjà labellisé et prêt à être utilisé. Néanmoins, ce n'est pas toujours le cas dans des situations de vie réelle - créer une source de vérité de base (ou *ground truth*) pour les données de production est une tâche coûteuse, longue et fastidieuse. Avec Amazon SageMaker Ground Truth, vous pouvez créer facilement et à peu de frais des jeux de données d'apprentissage automatique étiquetés de manière plus précise. Pour réduire les coûts d'étiquetage, utilisez l'apprentissage automatique Ground Truth pour choisir des images « difficiles » qui nécessitent des annotations humaines et des images « faciles » qui peuvent être étiquetées automatiquement avec l'apprentissage automatique. Vous pouvez choisir les humaines qui travailleront sur vos données pour vous aider à les labelliser : soit votre workforce privée à vous, par exemple des collègues, soit un prestataire entre une liste exhaustive d'agences 3P, soit une workforce publique via Amazon Mechanical Turk. En plus, le fichier manifest généré comme output par Amazon SageMaker Ground Truth est compatible avec les services AI, notamment Amazon Comprehend (Custom Classification et Custom Entity Recognition), et bien sûr Amazon Rekognition Custom Labels. Fun fact : pour produire le dataset utilisé dans la démonstration précédente, une équipe de 2 Solutions Architect a fait partie d'une équipe de travail privée de SageMaker Ground Truth afin d'obtenir le dataset entièrement labellisé.

Figure 1

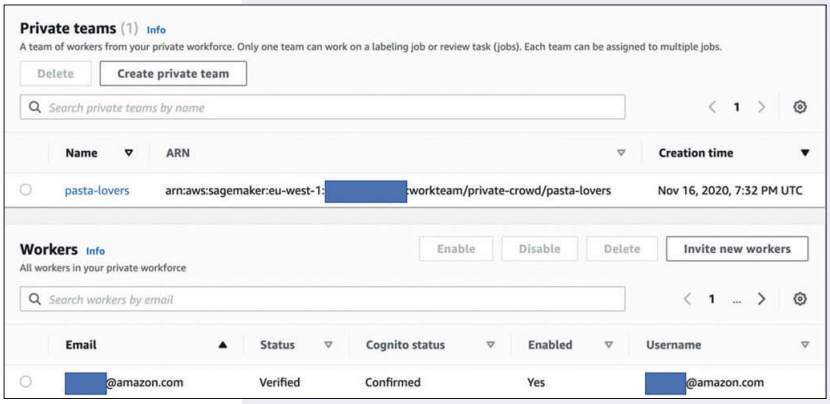
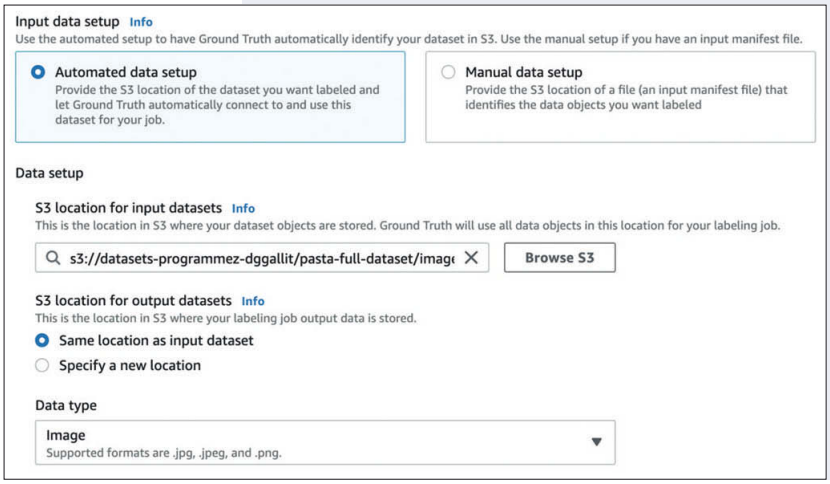


Figure 2



Quelles sont les étapes par lesquelles il faudrait passer pour mettre en place l'étiquetage automatique par SageMaker Ground Truth? Si on souhaitait les schématiser :

1. Charger les données sur S3 ;
2. Créer une workforce privée et y assigner un (ou plus) utilisateurs ;
3. Créer le workload de labellisation ;
4. Aller sur la page web pour commencer la tâche de classification.

La première étape est toujours de stocker les données sur S3. Comme montré tout à l'heure, vous pouvez charger les données via la console AWS, ou par CLI. Dans ce cas, on va utiliser la CLI pour charger toutes les images précédemment chargées, sans sous-répertoires. Positionnez-vous dans le répertoire avec vos images et utilisez la commande :

```
aws s3 cp . s3://datasets-programmez-VOTRE_NOM/pasta-full-dataset
/images/ --recursive
```

Une fois le chargement terminé, cherchez sur la console AWS le service SageMaker. Sans rentrer dans les détails - il y a un article dédié à SageMaker écrit par un de mes collègues - SageMaker est une plateforme permettant de résoudre tous les défis dans un pipeline de Machine Learning, dès la conception d'un algorithme jusqu'à la mise en production, dans une seule plateforme et avec le seul IDE spécifique pour le Machine Learning sur le marché. Aujourd'hui, on regardera uniquement Amazon SageMaker Ground Truth.

Avant de créer notre tâche d'étiquetage, choisissons notre workforce, qui s'occupera de labelliser nos données. Pour cette démo, nous allons être notre workforce - n'hésitez pas non plus à demander de l'aide à un de vos collègues! Dans la liste à gauche, choisissez "Labeling workforces", choisissez l'onglet "Private" et cliquez sur "Create a private team". Bien sûr, le nom de notre équipe sera "pasta-lovers"; laissons le reste des configurations par défaut. Vous allez être redirigés sur la page initiale des private workforces. Cliquez tout en bas sur "Invite New Workers" et rentrez votre adresse email dans le textbox qui apparaîtra (et celle de votre collègue qui est d'accord pour travailler avec vous sur ce projet, séparées par des virgules). La page initiale des workforces privées devrait ressembler à la suivante : **figure 1**

Vous pouvez désormais assigner le worker à la workforce : cliquez sur le team "pasta-lovers", et dans l'onglet "Workers" appuyez sur le bouton "Add workers to team". Rajoutez-le(s) worker(s) créé(s) précédemment. La workforce est enfin configurée.

C'est le moment de configurer la tâche d'étiquetage. Cliquez à gauche sur "Labeling Jobs"/"Étiquetage des tâches" et cliquez sur le bouton "Create a labelling job"/"Créer une tâche d'étiquetage". Rentrez "pasta-labeller" comme nom de la tâche,

et spécifiez le path vers S3 utilisé avant comme dans l'image suivante : **figure 2**

Pour avoir accès à vos ressources, Amazon SageMaker Ground Truth nécessite un rôle d'exécution : dans le drop down de la section IAM Role, choisissez "Create a new role"/"Créer un nouveau rôle", spécifiez le nom du bucket sous "Specific S3 Buckets" et cliquez sur "Create". Pour valider que vous avez tout bien configuré, cliquez sur "Complete data setup" et attendez le résultat positif. Comme mentionné auparavant, la tâche qu'on souhaite résoudre c'est une classification d'image avec un seule label entre plusieurs. Vérifiez donc que le type de tâche choisi soit bien "Image Classification (Single Label)" et laissez active la configuration pour le CORS. Appuyez sur "Next".

Configurez les workers. Choisissez la private team créée auparavant, et configurez le task timeout et task expiration time selon vos préférences (par exemple, 30 minutes par worker par image et une expiration à 10 jours). Si on avait beaucoup plus de données, on aurait pu choisir d'activer le ADL (Automated Data Labeling) : si active, SageMaker essayera d'apprendre la tâche en background, en utilisant les données labellisées par les humains dans notre workforce. Avec de larges ensembles de données, activer le automated data labelling permet jusqu'à 70% de réduction de coût - un exemple et une analyse est disponible sur le blog AWS Machine Learning [7] . **figure 3**

Enfin, vous pouvez fournir des instructions supplémentaires à vos collaborateurs, en détaillant la tâche spécifique à accomplir et en leur donnant quelques exemples. **figure 4**

Une fois personnalisée comme vous le souhaitez, cliquez sur "Create". En utilisant le lien reçu par mail, vous pouvez vous connecter à la plateforme et commencer à étiqueter le dataset! Si vous avez perdu le lien, vous pouvez toujours le récupérer en passant par la page "Labeling workforces", onglet "Private". **figure 5**

Pour vérifier l'avancement de votre tâche d'étiquetage, vous pouvez cliquer sur "Labeling Jobs" dans la console SageMaker, et choisir "pasta-labeller". Dès que vos collègues ont terminé d'étiqueter votre dataset, vous pouvez accéder au manifest.json de sortie disponible sur S3. Pour le récupérer, suivez le lien "Output Dataset Location" dans la page de la tâche d'étiquetage, et naviguez dans "manifests/output". Dans mon cas, le path complet est :

```
s3://datasets-programmez-dggallit/pasta-full-dataset/images/pasta-labeller
/manifests/output/output.manifest
```

Vous pouvez utiliser ce fichier manifest comme entrée pour votre tâche d'apprentissage avec Rekognition Custom Labels, en choisissant l'option "Import images labeled by Amazon SageMaker Ground Truth" pour "Image Location".

Figure 3

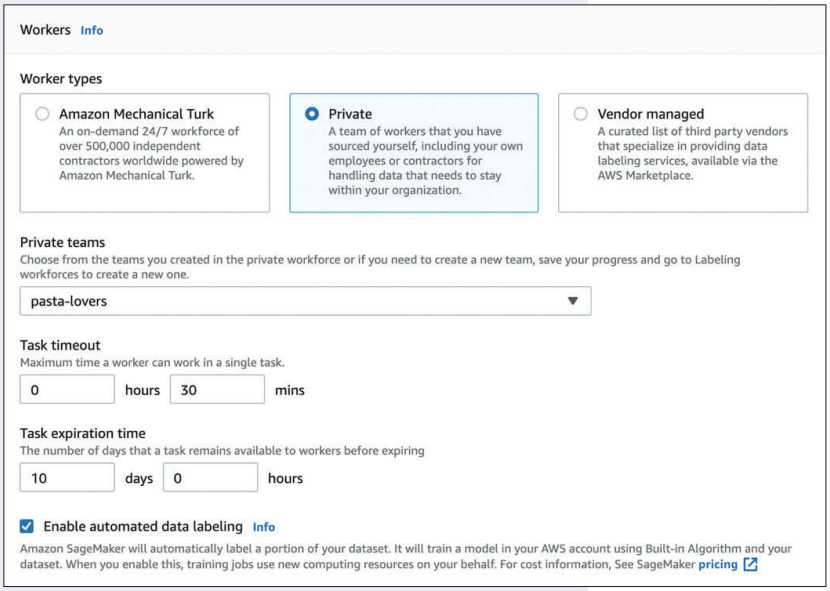


Figure 4

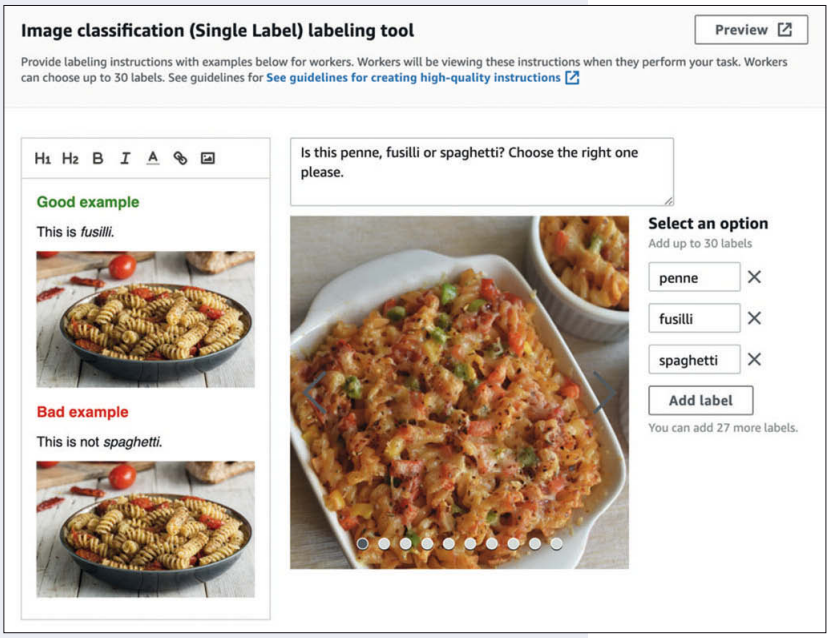
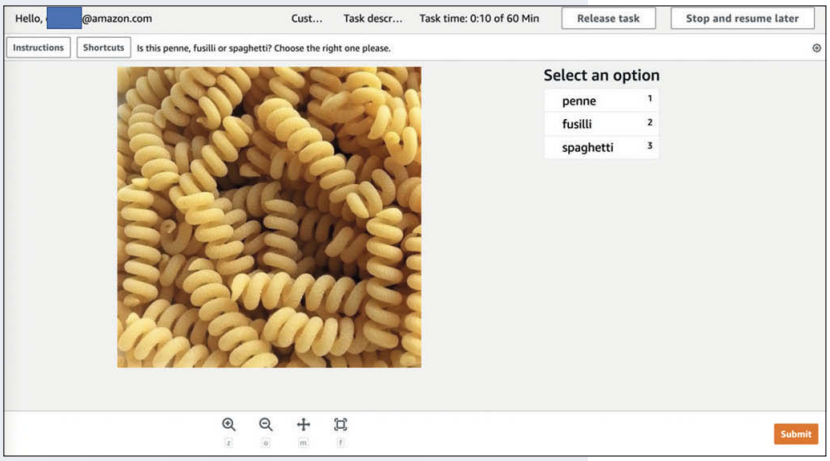


Figure 5



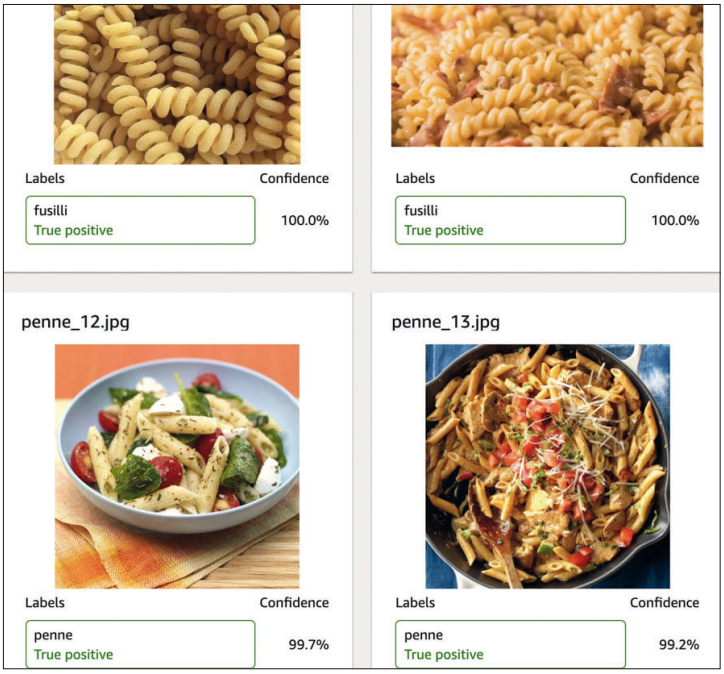


Figure 5

AUGMENTER LES PERFORMANCES DE PRÉDICTION AVEC A2I

Mais que se passe-t-il si le modèle d'étiquette personnalisée que vous avez formé ne peut pas reconnaître les images avec un niveau élevé de confiance, ou si vous avez besoin de votre équipe d'experts pour valider les résultats de votre modèle pendant la phase de test ou vérifier les résultats en production ? Vous pouvez facilement envoyer des prédictions depuis Amazon Rekognition Custom Labels à Amazon Augmented AI (Amazon A2I). Amazon A2I facilite l'intégration d'une évaluation humaine dans votre flux de travail ML. Cela vous permet d'obliger les humains à entrer automatiquement dans votre pipeline ML pour examiner les résultats inférieurs à un seuil de confiance, configurer les workflows de révision et d'audit, et augmenter les résultats de prédiction pour améliorer la précision du modèle.

complexe et les images bien distinctes, il s'agit des résultats attendus. On peut aussi voir des exemples des prédictions en cliquant sur "View Test Results" : **figure 5**

Déploiement du modèle

Pour utiliser le modèle que vous venez d'entraîner, il faut le démarrer en le déployant. Pour faire cela, un onglet "Code de l'API" plus en bas dans la même page montre la commande pour lancer le projet avec la CLI :

```
aws rekognition start-project-version \
 --project-version-arn "arn:aws:rekognition:eu-west-1:<AWS_Account
_ID>:project/pasta-classifier/version/<CLASSIFIER>>" \
 --min-inference-units 1 \
 --region eu-west-1
```

Pour vérifier l'état du endpoint, il vous suffit d'utiliser l'appel describe-project-version :

```
aws rekognition describe-project-versions —project-arn "<project-arn>"
```

Tester le modèle en JavaScript

Vous pouvez désormais intégrer votre pasta-classifier dans votre application JavaScript. La méthode dans le SDK JavaScript est très similaire à celle présentée précédemment pour la prédiction avec Amazon Rekognition standard, avec la seule différence étant que vous utilisez detectCustomLabels à la place de detectLabels:

```
var params = {
 Image: { /* nécessaire */
  Bytes: Buffer.from('...') || 'STRING_VALUE' /* Strings sont encodés en
Base-64 pour vous */,
 },
 ProjectVersionArn: 'STRING_VALUE', /* nécessaire */
 MaxResults: 'NUMBER_VALUE',
 MinConfidence: 'NUMBER_VALUE'
};
rekognition.detectCustomLabels(params, function(err, data) {
 if (err) console.log(err, err.stack); // erreur
 else    console.log(data);          // ok!
});
```

Comme vous le voyez, Rekognition Custom Labels simplifie grandement la création et le déploiement de modèles de classification d'images et de détection d'objets. Par ailleurs, la construction du jeu de données est elle aussi simplifiée grâce à SageMaker Ground Truth.
A vous de jouer.

Références

[1] Créer un compte AWS - https://amzn.to/3ntUtki
[2] Configurer la CLI AWS - https://amzn.to/3ntUtki
[3] Chaîne YouTube "Bien démarrer sur AWS" - https://bit.ly/38037q2
[4] Documentation du SDK JavaScript pour Amazon Rekognition - https://amzn.to/3f5eESP
[5] Multi-class classification - en.wikipedia.org/wiki/Multiclass_classification
[6] Set Up Amazon S3 Bucket Permissions for SDK Use - https://amzn.to/2UxRxqE
[7] Annotate data for less with Amazon SageMaker Ground Truth and automated data labeling - https://amzn.to/3f3txF8

Améliorez votre code et vos performances avec Amazon CodeGuru

Il peut être difficile de détecter certains problèmes de code et d'identifier les lignes de code les plus coûteuses sans une expertise en ingénierie des performances, même pour les plus expérimentés. Amazon CodeGuru [1] est un service qui vous permet de découvrir rapidement les problèmes de code et d'améliorer les performances des applications.

Steve Houël

Solutions Architect au sein des équipes AWS France. Basé à Nantes, son travail consiste à assister et accompagner les clients Français dans l'adoption des solutions basées sur les services AWS avec un focus tout particulier sur le développement d'applications Serverless.

Florent Brosse

Solutions Architect au sein des équipes AWS France, il aide les clients français en leur faisant découvrir et en leur prodiguant les bonnes pratiques de l'ensemble des services proposés par AWS.

C'est un service basé sur l'apprentissage machine, lancé lors du re:Invent 2019, pour la revue automatisée de code et les recommandations de performance des applications. CodeGuru fournit aux équipes de développement les outils nécessaires pour maintenir une qualité de code élevée tout au long du processus de développement d'une application Java et depuis peu le Python lors de l'annonce du re:Invent 2020.
CodeGuru s'appuie sur deux fonctionnalités majeures :

- **Reviewer** : fournit une analyse automatisée de votre code source.
- **Profiler** : fournit une visibilité et des recommandations sur les performances de votre application pendant son exécution.

Vue d'ensemble

Le diagramme ci-contre illustre la façon dont CodeGuru s'intègre dans les phases de développement de votre projet. Que ce soit durant vos revues du code ou le suivi des performances de votre application en production, le service vous permettra de garder un œil attentif et d'améliorer en continu sa qualité. **Figure 1**

Amazon CodeGuru Reviewer

Il aide les développeurs en leur évitant d'introduire des problèmes difficiles à détecter, à dépanner, à reproduire, et à en trouver la cause principale. Il leur permet également d'améliorer les performances des applications. Cela permet non seulement d'améliorer la fiabilité du logiciel, de votre code, mais aussi de réduire le temps passé à résoudre des problèmes liés au développement comme les problèmes d'accès concurrents, les fuites de ressources, les problèmes de sécurité lors de communication entre des threads, l'utilisation d'entrées non validées, la manipulation inappropriée de données sensibles et l'impact sur les performances de l'application, pour n'en citer que quelques-uns.
CodeGuru est alimenté par l'apprentissage machine, les meilleures pratiques et les leçons durement apprises au travers de millions de revues de code et de milliers d'applications issues de projets open source ou interne chez Amazon.

Regardons sous le capot !

CodeGuru Reviewer utilise un programme d'analyse de code combinée à des modèles d'apprentissage machine formés sur des millions de lignes de code Java provenant de la base de code Amazon et d'autres sources. Lorsque vous associez CodeGuru Reviewer à un dépôt Git, il peut trouver et signaler les défauts du code et suggérer des recommandations pour améliorer votre code. Le service fournit des recommandations pratiques avec un faible taux de faux positifs et peut améliorer sa capacité à analyser le code au fil du temps en fonction des commentaires des utilisateurs.
Vous pouvez associer CodeGuru Reviewer à un dépôt (GitHub, Github Enterprise, AWS CodeCommit, Bitbucket) pour lui permettre de fournir des recommandations. Une fois un dépôt associé, CodeGuru Reviewer analyse automatiquement les Pull Request que vous faites, ou vous pouvez choisir d'exécuter des analyses sur le code d'une branche spécifique pour analyser tout le code, et ce à tout moment. Les recommandations issues des analyses des Pull Request et des dépôts peuvent être consultées directement dans la console de CodeGuru Reviewer. Les recommandations issues des Pull Request peuvent également être consultées directement au sein du dépôt sous forme de commentaires.
Les développeurs peuvent décider comment intégrer les recommandations et fournir en retour des informations sur l'utilité de ces recommandations. Cela permet à votre équipe de développement de garantir la qualité du code et d'améliorer ses pratiques de manière organique et interactive. Et par la même occasion cela améliorera la qualité des recommandations du service ce qui rendra CodeGuru Reviewer de plus en plus efficace pour les analyses futures.

Les recommandations d'Amazon CodeGuru

CodeGuru Reviewer ne signalent pas les erreurs de syntaxe, car celles-ci sont relativement faciles à trouver. Il identifiera les problèmes plus complexes et suggérera des améliorations liées aux points suivants :

- Les meilleures pratiques d'AWS
- Gestion des accès concurrents
- Prévention de fuites de ressources
- Prévention de fuites d'informations sensibles
- Meilleures pratiques de codage
- Refactoring
- Validation des entrées

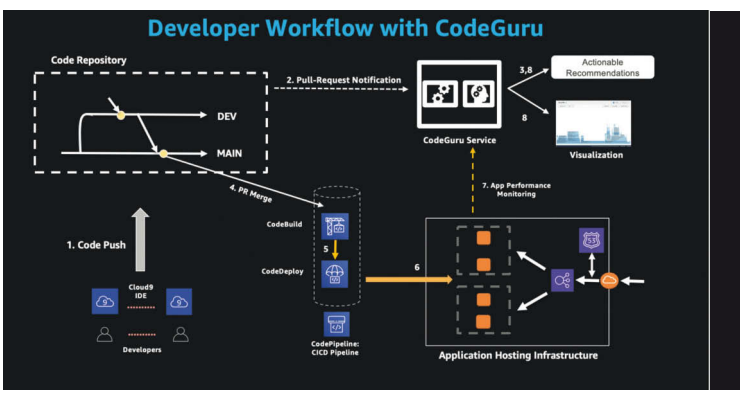


Figure 1

Les meilleures pratiques d'AWS
Les API AWS contiennent un ensemble de fonctionnalités pour assurer la performance et la stabilité des logiciels. Par exemple, l'utilisation de patterns tels que *Batching* et *Waiters* permettent d'améliorer les performances et d'obtenir un code plus efficace et plus facile à maintenir. Les développeurs peuvent ne pas utiliser les bons constructeurs lorsqu'ils utilisent les API AWS, ce qui entraîne des problèmes en production. Les meilleures pratiques AWS fournissent des recommandations sur l'utilisation correcte des API AWS, ce qui conduit à des gains de disponibilité et de performance.

Gestion des accès concurrents
CodeGuru Reviewer identifie les problèmes d'accès concurrents dans les morceaux de code s'exécutant sur plusieurs threads. Les défauts de simultanéité sont souvent subtils et échappent même aux programmeurs experts. Des implémentations incorrectes de la concurrence peuvent conduire à un code incorrect ou à des problèmes de performance. Le service identifie les violations de l'atomicité qui pourraient entraîner des problèmes d'exactitude des données et il identifie les synchronisations excessives qui pourraient entraîner des problèmes de performance.

Prévention des fuites de ressources
CodeGuru Reviewer recherche les lignes de code où des fuites de ressources pourraient se produire. Les fuites de ressources peuvent provoquer des problèmes de latence et des pannes. Le service peut indiquer les morceaux de code où cela pourrait se produire et suggérer de gérer les ressources d'une manière alternative.

Prévention des fuites d'informations sensibles
Les informations sensibles codées ne doivent pas être partagées avec des parties non autorisées. CodeGuru Reviewer recherche les lignes de code où des informations sensibles pourraient fuir, et suggère différentes façons de traiter les données.

Meilleures pratiques de codage
CodeGuru Reviewer vérifie les paramètres et recherche les lignes de code qui pourraient créer des bogues. Il existe de nombreuses erreurs de codage courantes qui provoquent des bogues, comme oublier de vérifier si un objet est nul avant de le définir, réassigner un objet synchronisé ou oublier d'initialiser une variable le long d'un chemin d'exception. Il peut indiquer l'emplacement de ces erreurs et d'autres sources de problèmes dans le code.

Refactoring
CodeGuru Reviewer recherche les lignes de code qui semblent être dupliquées ou suffisamment similaires pour être retraitées. La refonte peut contribuer à améliorer la maintenabilité du code.

Validation des entrées
Il est important de détecter les entrées inattendues qui arrivent à un calcul, et d'appliquer une validation appropriée avant que le calcul ne commence. La validation des entrées est une couche de défense efficace contre les erreurs involontaires, telles que les changements de composants du client, et les attaques malveillantes, supprimer l'injection de code ou le déni de service. CodeGuru Reviewer recherche les lignes de code qui traitent les paramètres d'entrée et suggère une validation supplémentaire là où elle est nécessaire.

Qualité du code
L'analyseur de code vous suggère comment vous pouvez améliorer la qualité de votre code. Voici quelques-uns des problèmes qui peuvent être remontés.

Nombre de ligne de code d'une méthode (Source LOC)
CodeGuru Reviewer détecte le nombre de lignes de code (Source LOC) d'une méthode. Les méthodes avec un nombre élevé de lignes peuvent être difficiles à lire et avoir une logique difficile à comprendre et à tester.

Complexité cyclomatique des méthodes
La complexité cyclique indique le nombre de décisions qui sont prises dans une méthode. Une méthode à forte complexité cyclomatique peut rendre sa logique difficile à comprendre et à tester.

Ventilation d'une méthode
La ventilation d'une méthode indique combien de méthodes sont appelées par une méthode donnée. Les méthodes avec une forte ventilation sont fortement couplées avec d'autres méthodes. Cela peut les rendre difficiles à comprendre et vulnérables à des changements de comportement inattendus lorsqu'une de leurs méthodes de référence est mise à jour.

Ventilation des classes
La Ventilation des classes indique combien d'autres classes sont référencées par une classe donnée. Plus le nombre de classes référencées est élevé, plus il est couplé avec d'autres classes et plus la ventilation est importante. Les classes avec une ventilation élevée peuvent être complexes, difficiles à comprendre, et peuvent changer de manière inattendue lorsqu'une classe référencée est mise à jour.

Cohésion des classes
CodeGuru Reviewer remarque si une classe contient des groupes de méthodes d'instance qui n'ont aucun membre de la classe en commun. Par exemple, un groupe de deux méthodes peut accéder uniquement aux champs de classe x et y, et un autre groupe de méthodes de la même classe peut accéder uniquement aux champs de classe a et b. Un nombre élevé de ces groupes indique une faible cohésion de classe. Les classes à faible cohésion contenant des opérations sans rapport, peuvent être difficiles à comprendre et sont souvent moins susceptibles d'être utilisées.

Comment utiliser CodeGuru Reviewer ?
Projetons-nous dorénavant sur un cas réel en utilisant l'un de nos dépôts **aws-lambda-powertools-java** [2], il s'agit ici d'une suite d'outils Java Open-Source pour AWS Lambda afin de faciliter l'intégration des fonctions avec nos services d'observabilité. Tout d'abord, il vous faut disposer d'un compte AWS. Si vous n'en avez pas encore, vous pouvez en créer un gratuitement à partir de la page d'inscription AWS [3]. Une fois connecté à votre console AWS, rendez-vous sur le service Amazon CodeGuru [4]. Vous pouvez y accéder en sélection-

nant Amazon CodeGuru dans le menu Services en haut de la page ou en saisissant CodeGuru dans la barre de recherche des Services AWS.

Associer un répertoire Git

Suivez les étapes suivantes :

- Choisissez Examinateur dans le panneau de gauche et choisissez Dépôt associé.
- Choisissez GitHub, puis choisissez Connexion à GitHub.
- Une fois authentifiée et la connexion établie, nous sélectionnons le dépôt **aws-lambda-powertools-java** dans la liste déroulante puis nous cliquons sur **Associer**, comme indiqué dans la capture d'écran suivante. **Figure 2**

Cette action associe CodeGuru Reviewer au dépôt spécifié et va écouter tout événement lié à celui-ci de type *Pull Request*.

Utilisation du tableau de bord des revues de code

AWS met à disposition un tableau de bord basé sur les premiers retours d'information, centralisant en un endroit l'historique des revues de code de l'ensemble des dépôts sur une période de 90 jours. Cette page répertorie l'ensemble des revues de code avec des informations complémentaires telles que le statut de la revue de code, le référentiel, le nombre de recommandations, et plus encore.

Exemple de recommandations

Ci-contre un exemple de recommandations faites sur le dépôt : **figure 3**

Comme vous pouvez le voir dans les recommandations, non seulement les problèmes de code sont détectés, mais une recommandation détaillée est également fournie sur la manière de résoudre les problèmes, avec le cas échéant des exemples et de la documentation, le cas échéant. Pour chacune des recommandations, un développeur peut donner son avis sur l'utilité ou non de la recommandation en sélectionnant simplement un émoji dans la rubrique "Cela vous a-t-il été utile ?". Notez que le service CodeGuru est utilisé pour identifier les défauts fonctionnels difficiles à trouver et non les erreurs syntaxiques. Les erreurs syntaxiques doivent être signalées par l'IDE et traitées à un stade précoce du développement voir au travers d'une étape dédiée de normalisation de votre code selon votre style. CodeGuru est introduit à un stade ultérieur dans le flux de travail des développeurs, lorsque le code est déjà développé, testé unitairement et prêt à être revu.

Amazon CodeGuru Profiler

CodeGuru Profiler recherche constamment à optimiser les performances de l'application, en identifiant vos lignes de code les plus « onéreuses » et en recommandant des manières de les corriger en vue de réduire l'utilisation de la CPU, de diminuer les coûts de calcul et d'améliorer les performances de l'application.

Comprendre le comportement d'exécution des applications

CodeGuru Profiler analyse en permanence l'utilisation du CPU par l'application et les caractéristiques de latence pour vous montrer où vous passez le plus de temps dans votre application. Cette analyse est présentée dans un graphique de type flamme interactif qui vous aide à comprendre facilement graphiquement quels chemins de code consomment le plus de ressources, à vérifier que votre application fonctionne normalement et à découvrir les domaines qui peuvent être optimisés davantage.

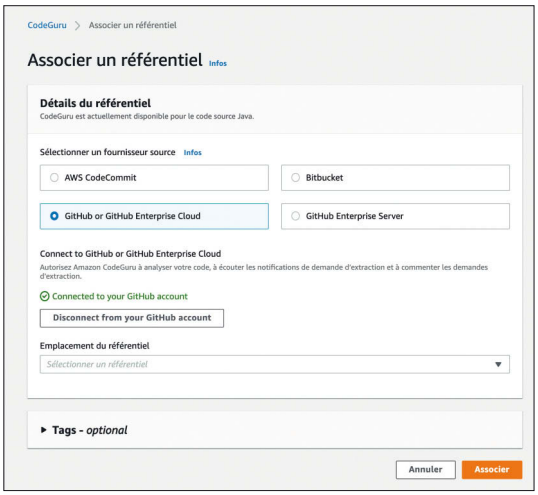


Figure 2

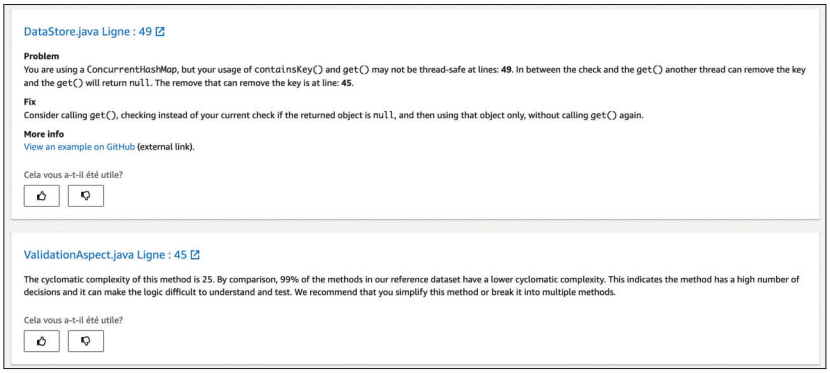


Figure 3

Recommandations intelligentes

CodeGuru Profiler identifie automatiquement les problèmes de performance dans votre application et adresse des recommandations intelligentes concernant la façon d'y remédier. Ces recommandations vous aident à identifier et à optimiser les méthodes les plus onéreuses ou les plus gourmandes en ressources au sein du code sans devoir être un expert de l'ingénierie de performance. Ces optimisations vous aident à réduire le coût de votre infrastructure, à réduire la latence et à améliorer votre l'expérience utilisateur. CodeGuru Profiler permet de trier quelles sont les recommandations qui ont le plus d'impact en terme de coût CPU. En effet un code exécuté très peu souvent a moins d'impact qu'un code exécuté plus fréquemment. On peut donc se concentrer sur les optimisations qui ont le meilleur retour sur investissement.

Détection des anomalies

CodeGuru Profiler analyse en permanence les profils de votre application en temps réel et détecte les anomalies dans son comportement et de ses méthodes. Chaque anomalie est suivie dans le rapport de recommandation, et vous pouvez voir les séries chronologiques du comportement de la latence de la méthode au fil du temps, les anomalies étant clairement mises en évidence. Si cela est configuré, une notification Amazon SNS est également envoyée lorsqu'une nouvelle anomalie est détectée.

Profilage permanent d'applications en production

CodeGuru Profiler est conçu pour s'exécuter de manière

continue en production avec un impact CPU minime, ce qui signifie que vous pouvez le maintenir activé en n'ayant que très peu d'impact sur les performances. Cela vous permet d'établir le profil et de dépanner votre application à l'aide de véritables modèles de trafic client et de facilement détecter les problèmes de performance qui ne seraient pas détectés dans votre environnement de test.

Sécurité

Comme pour tous les services d'AWS la sécurité est primordiale. Il faut donc que l'application qui utilise le profiler ait les permissions requises.

Pour s'authentifier depuis le cloud AWS, il vaut mieux passer par un rôle IAM. Un rôle IAM est une identité IAM que vous pouvez créer dans votre compte et qui dispose d'autorisations spécifiques. Un rôle IAM est similaire à un utilisateur IAM, car il s'agit d'une identité AWS avec des stratégies d'autorisation qui détermine ce que l'identité peut et ne peut pas faire dans AWS. Par contre si vous voulez utiliser le profiler dans votre application qui se lance sur votre ordinateur ou dans votre datacenter il faut passer par un utilisateur et son ID de clé d'accès et une clé d'accès secrète. Il faut aussi que votre application puisse communiquer avec le endpoint de CodeGuru Profiler que ce soit par internet ou par un VPC Endpoints.

Comment utiliser CodeGuru Profiler ?

Il supporte tous les langages de la JVM : Java, Scala, Kotlin, Groovy, Jython, JRuby et Clojure et depuis le re:Invent 2020 il supporte Python. Pour Java les recommandations sont plus importantes que pour les autres langages.

Vous pouvez charger le profiler de deux manières dans votre application qui tourne sur une JVM :

Soit par la ligne de commande avec l'option -javaagent

- Télécharger le fichier https://repo1.maven.org/maven2/software/amazon/codeguruprofiler/codeguru-profiler-java-agent-standalone/1.0.3/codeguru-profiler-java-agent-standalone-1.0.3.jar
- Le mettre à un endroit accessible à l'application JVM.
- Le seul paramètre obligatoire est le nom du groupe de profilage. Il peut être défini par une variable d'environnement AWS_CODEGURU_PROFILER_GROUP_NAME ou dans la ligne de commande profilingGroupName comme dans l'exemple suivant.
- Lancer votre application java -javaagent:/path/to/codeguru-profiler-1.0.3.jar=profilingGroupName:MyGroup -jar MyApplication.jar

Soit avec du code dans votre application. Le code permet de contrôler quand on veut démarrer le Profiler, mais nécessite un changement de code et un ajout d'une dépendance. Avec Maven on peut le faire en modifiant le fichier de configuration pom.xml et en relançant la création de l'artifact.

```
<dependencies>
    ...
    <dependency>
      <groupId>software.amazon.codeguruprofiler</groupId>
      <artifactId>codeguru-profiler-java-agent</artifactId>
      <version>1.0.3</version>
    </dependency>
</dependencies>
```

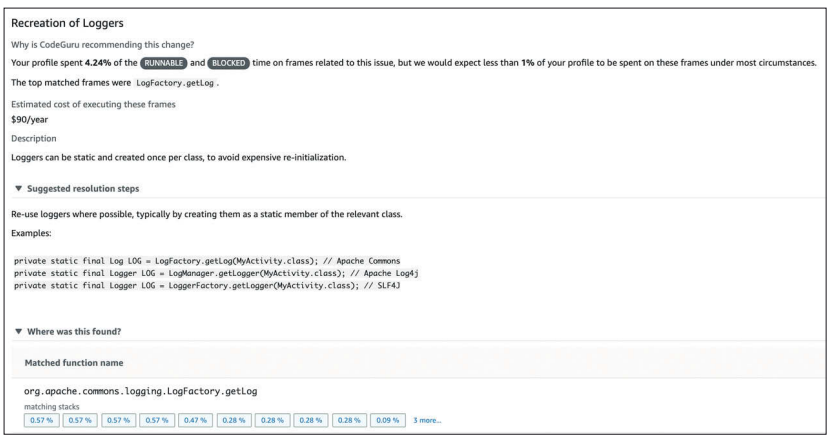
Ensuite il suffit de démarrer le profiler à l'endroit que vous voulez. Cela peut être dans la méthode main par example.

```
Profiler.builder()
  .profilingGroupName("MyProfilingGroup")
  .build()
  .start();
```

Une fois l'application lancée par une des 2 méthodes et après avoir attendu une dizaine de minute, vous allez voir le graphe flamme et les recommandations intelligentes apparaître dans le service CloudGuru de la console Amazon. Félicitations, vous avez accès à une meilleure compréhension de la performance et aux optimisations.

CodeGuru Profiler permet de visualiser une application de démonstration qui contient du code non optimal et de la comparer avec la même application optimisée. Le code de l'application est disponible ici : https://github.com/aws-samples/aws-codeguru-profiler-demo-application. Cette application lance en continu des traitements d'images qui utilisent fortement le CPU et une autre tâche qui écrit dans une queue SQS. CoduGuru profiler nous propose les recommandations afin de passer de la version sub optimale à la version optimisée. Par exemple cette recommandation sur la recréation des instances de journalisation. **Figure 4**

Figure 4



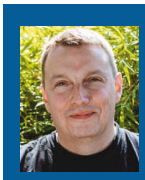
Conclusion

En 2008, nous voyons l'apparition du Software Craftmanship en tant que mouvement, mais aussi directement comme la cinquième valeur du manifeste Agile "Craftsmanship over Execution" (en français: "l'artisanat plus que l'exécution"). Ce mouvement prône le côté artisanal du développement logiciel, autrement dit, d'après le manifeste de l'artisanat du logiciel, il ne suffit pas qu'un logiciel soit fonctionnel, mais il faut qu'il soit bien conçu. L'idée principale est de garantir la fiabilité et la maintenabilité des applications d'où l'importance de professionnels aptes à concevoir des logiciels dans le respect d'indicateurs de qualité logicielle. CodeGuru va venir s'intercaler dans ces phases de développement et apporter une garantie du respect des bonnes pratiques et ce de façon automatisée. La culture d'amélioration continue et de transmission du savoir se fera par le biais des retours utilisateurs qui serviront à améliorer son moteur d'apprentissage machine et à en faire bénéficier directement l'intégralité des utilisateurs du service.

Références

(1) https://aws.amazon.com/fr/codeguru
(2) https://github.com/awslabs/aws-lambda-powertools-java
(3) https://portal.aws.amazon.com/billing/signup?language=fr_fr
(4) https://console.aws.amazon.com/codeguru/home

Améliorez la recherche de données d'entreprise avec Amazon Kendra



Julien Simon

En tant qu'évangéliste mondial de l'IA et de l'apprentissage automatique, Julien s'attache à aider les développeurs et les entreprises à donner vie à leurs idées. Il prend souvent la parole lors de conférences, et écrit sur le blog AWS. Avant de rejoindre AWS, Julien a occupé pendant 10 ans des postes de CTO/VP Engineering dans des startups Web de haut niveau.

Amazon Kendra est un service de recherche d'entreprise facile à utiliser qui vous permet d'ajouter des fonctionnalités de recherche à vos applications afin que les utilisateurs finaux puissent trouver des informations stockées dans différentes sources de données au sein de votre entreprise. Cela pourrait inclure des factures, des documents commerciaux, des manuels techniques, des rapports de vente, des glossaires d'entreprise, des sites Web internes, etc. Vous pouvez collecter ces informations à partir de solutions de stockage comme Amazon Simple Storage Service (S3) et OneDrive, d'applications telles que SalesForce, SharePoint et Service Now, ou de bases de données relationnelles comme Amazon Relational Database Service (RDS).

Lorsque vous entrez une question, le service utilise des algorithmes d'apprentissage automatique (machine learning, ou ML) pour comprendre le contexte et renvoyer les résultats les plus pertinents, qu'il s'agisse d'une réponse précise ou d'un document entier. Plus important encore, vous n'avez pas besoin d'avoir une expérience ML pour le faire. Kendra vous fournit également le code dont vous avez besoin pour intégrer facilement vos applications nouvelles ou existantes.

Cet article vous montre comment créer votre recherche interne d'entreprise en utilisant les fonctionnalités de Kendra. Cela vous permet de créer une solution pour créer et interroger votre propre index de recherche. Nous utiliserons les documents d'aide Amazon.com au format HTML comme source de données, mais Kendra prend également en charge les formats MS Office (.doc, .ppt), PDF et texte.

Vue d'ensemble de la solution

Cet article détaille les étapes pour vous aider à créer un moteur de recherche d'entreprise sur AWS à l'aide de Kendra. Vous pouvez mettre en service un nouvel index Kendra en moins d'une heure sans trop de difficulté ou d'expérience ML. La publication explique également comment configurer Kendra pour une expérience personnalisée en ajoutant des FAQ, en déployant Kendra dans des applications, et en synchronisant les sources de données. Les prérequis sont :

- Un compte AWS.
- Connaissances de base sur AWS : IAM, S3, etc.
- Un compartiment S3 pour vos documents.

Création et configuration de votre référentiel de documents

Avant de pouvoir créer un index dans Kendra, vous devez charger des documents dans un compartiment S3. Cette section contient des instructions pour créer un compartiment S3, récupérer les fichiers et les charger dans le compartiment. Après avoir terminé toutes les étapes de cette section, vous disposez d'une source de données que l'on pourra utiliser.

Voici comment créer un compartiment S3 contenant les données que nous allons indexer :

- Dans la console AWS, dans la liste Région, choisissez une région dans laquelle Kendra est disponible.
- Choisissez Services.
- Sous Stockage, choisissez S3.
- Sur la console S3, choisissez « Créer un compartiment ».
- Sous Configuration générale, fournissez les informations suivantes :
  - Nom du compartiment : « kendrapost- {votre identifiant de compte} ».
  - Région : choisissez la même région que celle que vous utilisez pour déployer votre index Kendra.
  - Sous Paramètres du compartiment pour Bloquer l'accès public, laissez tout avec les valeurs par défaut.
  - Sous Paramètres avancés, laissez tout avec les valeurs par défaut.
  - Choisissez Créer un compartiment.
- Téléchargez amazon_help_docs.zip https://aws-ml-blog.s3.amazonaws.com/artifacts/kendra-docs/amazon_help_docs.zip et décompressez les fichiers.

Figure 1

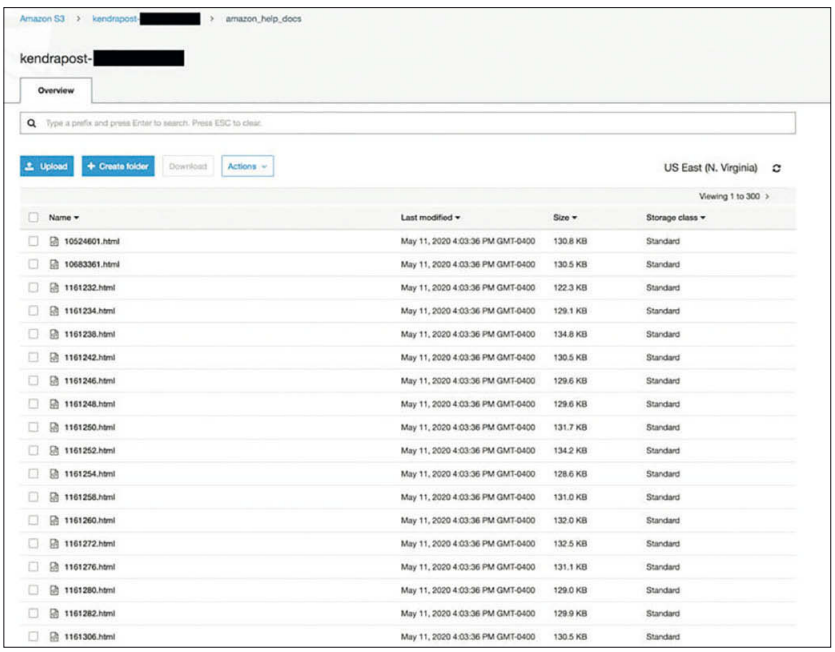


Figure 2

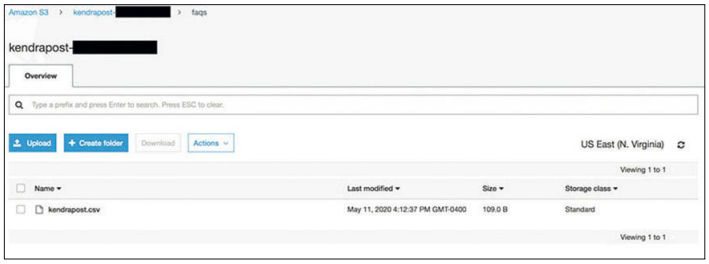


Figure 3

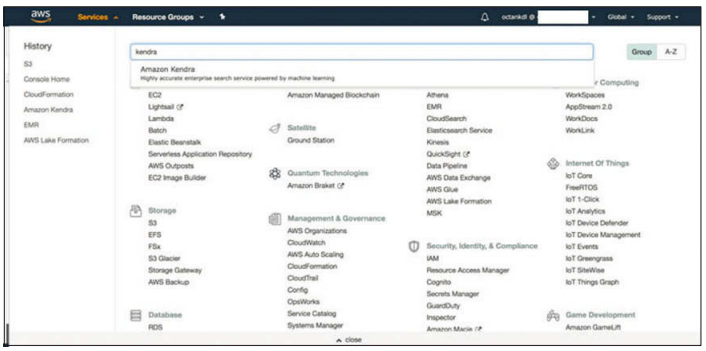


Figure 4

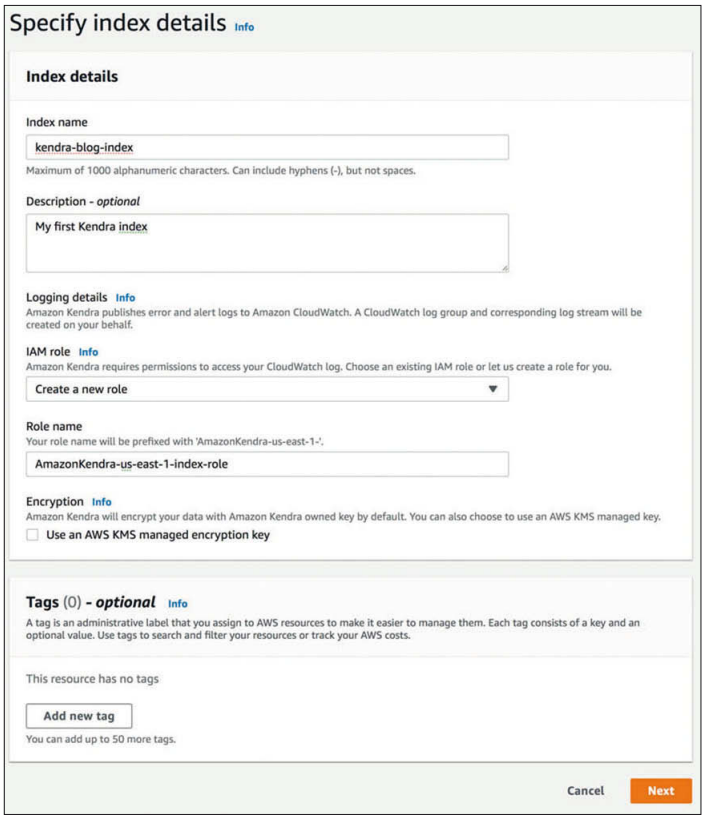
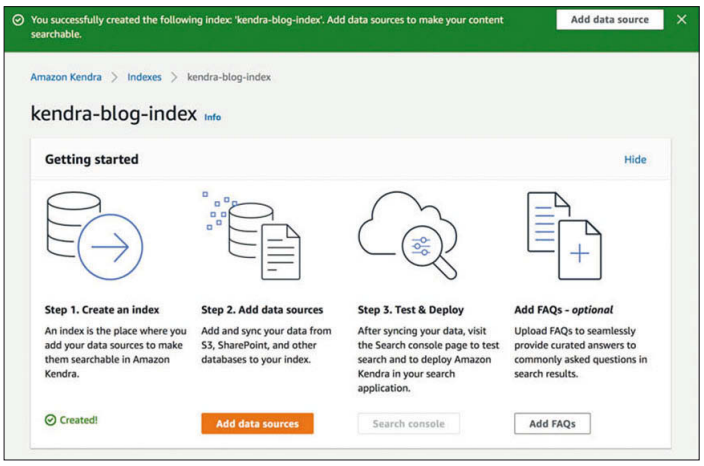


Figure 5



- Dans la console S3, sélectionnez le compartiment que vous venez de créer et choisissez Télécharger.
- Chargez les fichiers décompressés.

Dans votre compartiment, vous devriez maintenant voir deux dossiers : amazon_help_docs (avec 3 100 objets) et FAQ (avec un objet).

La capture d'écran suivante montre le contenu de amazon_help_docs. **Figure 1**

La capture d'écran suivante montre le contenu des FAQ. **Figure 2**

Création d'un index

Un index est le composant Kendra qui fournit des résultats de recherche pour les documents et les questions fréquemment posées. Après avoir terminé toutes les étapes de cette section, vous disposez d'un index prêt à consommer des documents provenant de différentes sources de données. Pour plus d'informations sur les index, reportez-vous à la section Index.

Pour créer votre premier index Amazon Kendra, procédez comme suit :

Dans la console, choisissez Services. Dans Machine Learning, choisissez Amazon Kendra. **Figure 3**

Sur la page principale d'Amazon Kendra, choisissez « Créer un index ».

- Dans la section « Détails de l'index », saisissez « kendra-blog-index » comme nom de l'index.
- Pour « Description », entrez « Mon premier index Kendra ».
- Pour le rôle IAM, choisissez « Créer un nouveau rôle ».
- Dans « Nom du rôle », entrez « index-role » (votre nom de rôle a le préfixe AmazonKendra-yourRegion-).
- Pour le chiffrement, ne sélectionnez pas « Utiliser une clé de chiffrement gérée AWS KMS ».

(Vos données sont chiffrées avec une clé appartenant à Amazon Kendra par défaut.) **Figure 4**

Kendra propose deux éditions :

- Kendra Enterprise Edition fournit un service haute disponibilité pour les charges de travail de production.
- Kendra Developer Edition est adapté à la construction d'un prototype.

Pour plus d'informations sur le niveau gratuit, les limites de taille des documents et le stockage total pour chaque édition Kendra : consultez la page des tarifs.

Nous utiliserons l'édition Developer.

Le processus de création d'index peut prendre jusqu'à 30 minutes. Lorsque le processus de création est terminé, un message s'affiche en haut de la page indiquant que vous avez réussi à créer votre index. **Figure 5**

Ajout d'une source de données

Une source de données est un emplacement qui stocke les documents à des fins d'indexation. Vous pouvez synchroniser automatiquement les sources de données avec un index Kendra pour vous assurer que les recherches reflètent correctement les documents nouveaux, mis à jour ou supprimés dans les référentiels sources. Après avoir terminé toutes les étapes de cette section, vous disposez d'une source de données liée à Kendra. Avant de continuer, assurez-vous que la création de l'index est terminée et que l'index s'affiche comme Actif.

Sur la page « kendra-blog-index », choisissez « Ajouter des sources de données ».
Kendra prend en charge six types de sources de données : S3, SharePoint Online, ServiceNow, OneDrive, Salesforce Online et Amazon RDS.
Sous S3, choisissez « Ajouter un connecteur ».
Dans la section « Définir les attributs », pour « Nom de la source de données », entrez « amazon_help_docs ». Pour « Description », entrez « la documentation des services AWS ». **Figure 6**
Dans la section « Configurer les paramètres », dans « Entrez l'emplacement de la source de données », entrez le compartiment S3 que vous avez créé : « kendrapost- {your account id} ». Ne modifiez pas l'emplacement du dossier des fichiers de métadonnées. Par défaut, les fichiers de métadonnées sont stockés dans le même répertoire que les documents. Si vous souhaitez placer ces fichiers dans un autre dossier, vous pouvez ajouter un préfixe. Pour sélectionner la clé de déchiffrement, laissez-la désélectionnée.
Dans « Nom du rôle », entrez « source-role » (votre nom de rôle est préfixé d'AmazonKendra-). Pour « Configuration supplémentaire », vous pouvez ajouter une règle pour inclure ou exclure certains dossiers ou fichiers. Ici, conservez les valeurs par défaut. **Figure 7**
Pour « Fréquence », choisissez « Exécuter à la demande ». Cette étape définit la fréquence à laquelle la source de données est synchronisée avec l'index Amazon Kendra. Pour cette procédure pas à pas, vous le faites manuellement (une seule fois). **Figure 8**
Dans la page « Réviser et créer », choisissez « Créer ».
Après avoir créé la source de données, choisissez « Synchroniser maintenant » pour synchroniser les documents avec l'index Amazon Kendra. **Figure 9**
La durée de ce processus dépend du nombre de documents que vous indexez. Pour ce cas d'utilisation, cela peut prendre 15 minutes, après quoi vous devriez voir un message indiquant que la synchronisation a réussi. **Figure 10**
Dans la section « Historique des exécutions de synchronisation », vous pouvez voir que 3 099 documents ont été synchronisés.

Explorer l'index de recherche à l'aide de la console de recherche

L'objectif de cette section est de vous permettre d'explorer les requêtes de recherche possibles via la console Amazon Kendra intégrée.
Pour interroger l'index que vous avez créé ci-dessus, procédez comme suit :

- Sous « Index », choisissez « kendra-blog-index ». **Figure 11**
- Choisissez « Console de recherche ».

Kendra peut répondre à trois types de questions : factoid, descriptif et mot-clé. Pour plus d'informations, consultez la FAQ. Vous pouvez poser des questions en utilisant les documents d'aide Amazon.com que vous avez chargés précédemment.
Dans le champ de recherche, entrez « What is Amazon Music Unlimited ? ».
Avec une question factoïde (qui, quoi, quand, où), Amazon Kendra peut répondre et proposer également un lien vers le document source. Dans le cadre d'une recherche par mots-

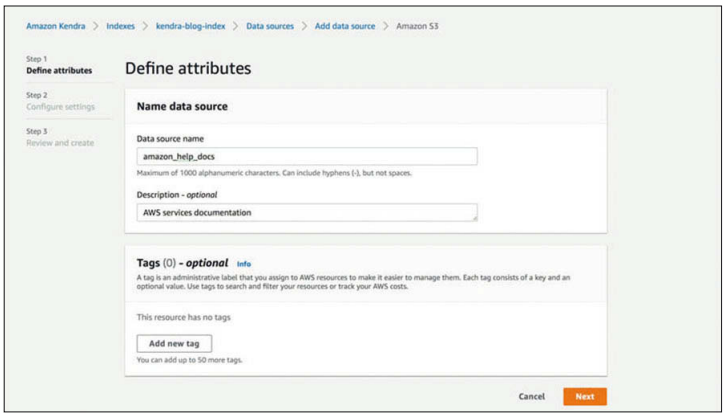


Figure 6

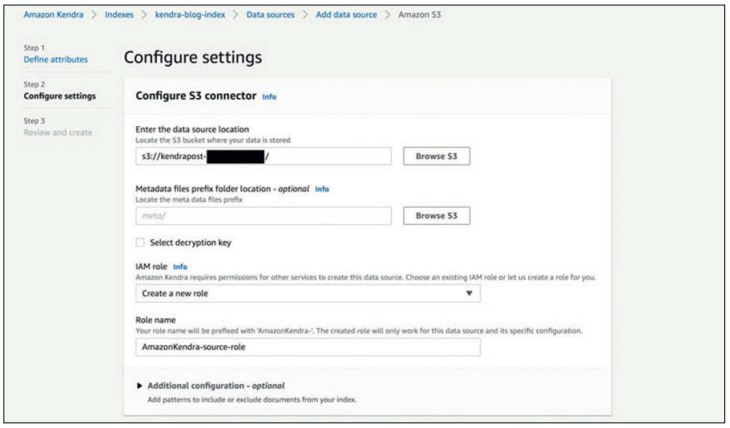


Figure 7

Figure 8

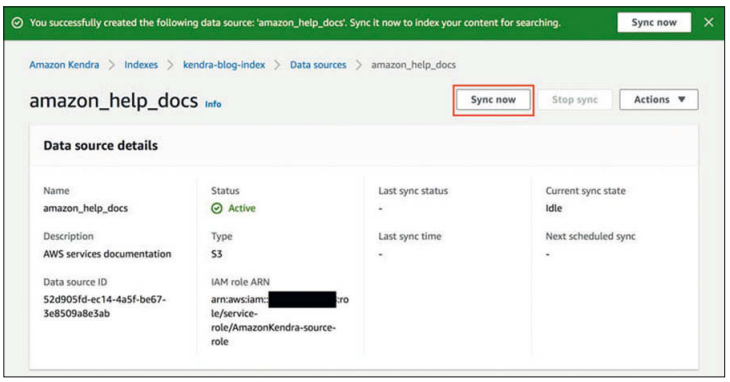


Figure 9

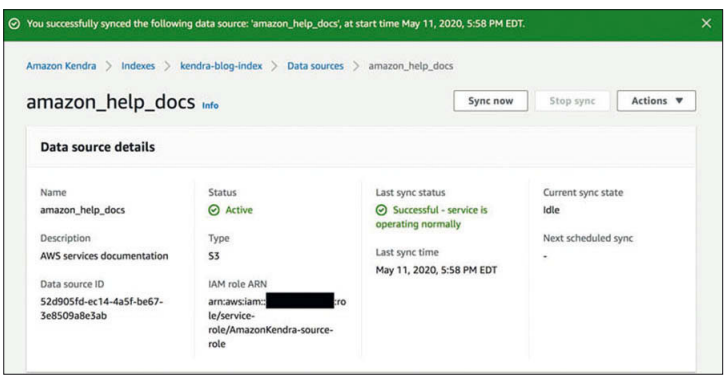
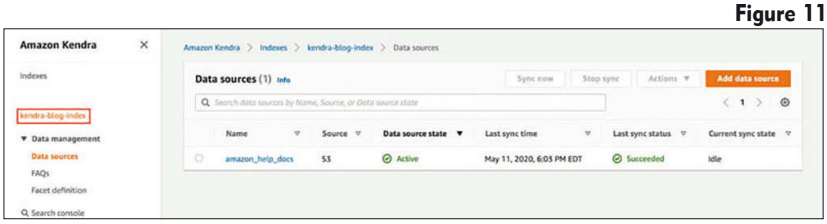


Figure 10

Figure 11



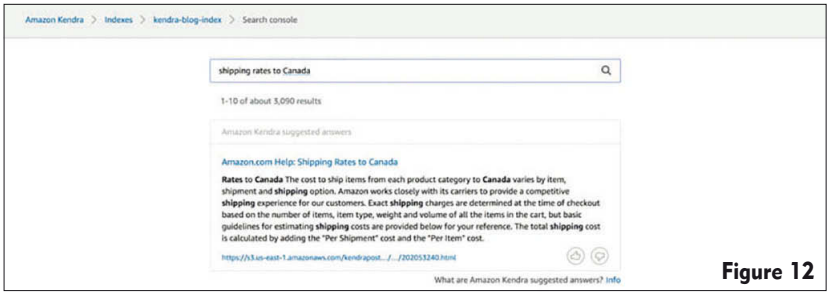


Figure 12

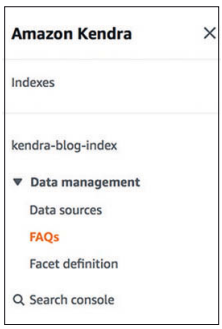


Figure 13

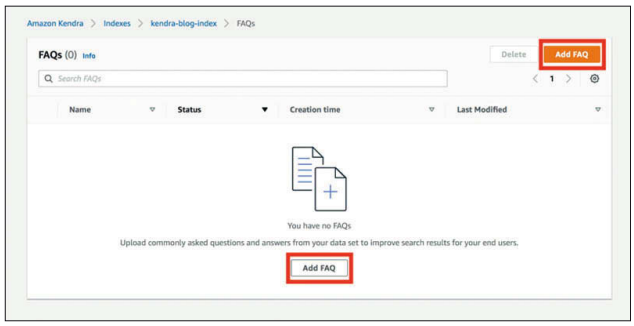


Figure 14

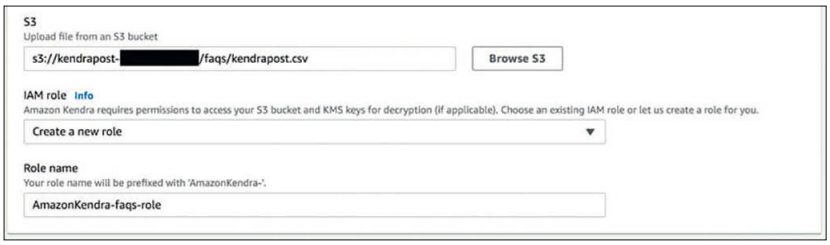


Figure 15

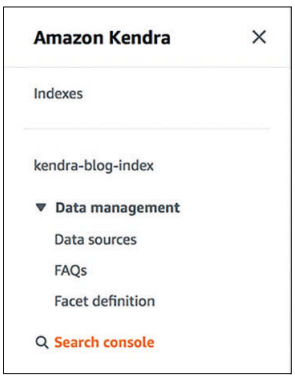


Figure 16

clés, entrez « Shipping rates to Canada ». La **Figure 12** montre la réponse donnée par Amazon Kendra.

Ajout d'une FAQ

Vous pouvez également télécharger une liste de FAQ pour fournir des réponses directes aux questions courantes que vos utilisateurs finaux posent. Pour ce faire, vous devez charger un fichier .csv avec les informations relatives aux questions. Cette section contient des instructions pour créer et configurer ce fichier et le charger dans Kendra.

Pour ce faire :

- Sur la console Kendra, accédez à votre index.
- Sous « Gestion des données », choisissez « FAQ ». **Figure 13**
- Choisissez « Ajouter une FAQ ». **Figure 14**
- Dans la section « Définir un projet FAQ », pour « nom FAQ », entrez « kendra-post-faq ».
- Pour « Description », saisissez « Ma première liste de questions fréquentes ».

Kendra accepte les fichiers .csv formatés avec chaque ligne commençant par une question suivie de sa réponse. Par exemple, consultez le tableau suivant.

Question	Answer
What is the height of the Space Needle? https://www.spaceneedle.com/	605 feet
How tall is the Space Needle? https://www.spaceneedle.com/	605 feet
What is the height of the CN Tower? https://www.cntower.ca/	1815 feet
How tall is the CN Tower? https://www.cntower.ca/	1815 feet

Voici à quoi le fichier .CSV ressemble.

"How do I sign up for the Amazon Prime free Trial?"," To sign up for the Amazon Prime free trial, your account must have a current, valid credit card. Payment options such as an Amazon.com Corporate Line of Credit, checking accounts, prepaid credit cards, or gift cards cannot be used. "," https://www.amazon.com/gp/help/customer/display.html/ref=hp_left_v4_sib?ie=UTF8&nodeId=201910190"

Sous « Paramètres FAQ », pour S3, entrez « s3://kendrapost-{votre identifiant de compte} /faqs/kendrapost.csv ».
Pour le rôle IAM, choisissez « Créer un nouveau rôle ».
Dans « Nom du rôle », entrez « FAQS-role » (votre nom de rôle est préfixé par AmazonKendra-). **Figure 15**
Choisissez « Ajouter ».
Attendez jusqu'à ce que l'état s'affiche comme Actif.
Vous pouvez maintenant voir comment fonctionne la FAQ sur la console de recherche. Sous « Index », choisissez votre index. Sous « Gestion des données », choisissez « Console de recherche ». **Figure 16**
Dans le champ de recherche, entrez "How do I sign up for the Amazon Prime free Trial?". La capture d'écran suivante montre le rajout de la FAQ que vous avez précédemment téléchargée dans la liste des résultats, et fournit une réponse et un lien vers la documentation connexe. **Figure 17**

Utilisation de Kendra dans vos propres applications

Vous pouvez ajouter les composants suivants à partir de la console de recherche de votre application :

- Page de recherche principale : page principale qui contient tous les composants. C'est là que vous intégrez votre application à l'API Amazon Kendra.
- Barre de recherche — Composant dans lequel vous entrez un terme de recherche et qui appelle la fonction de recherche.
- Résultats — Composant qui affiche les résultats d'Amazon Kendra. Il comporte trois composantes : les réponses suggérées, les résultats de la FAQ et les documents recommandés.
- Pagination — Composant qui pagine la réponse à partir d'Amazon Kendra.

Kendra fournit le code source que vous pouvez déployer sur votre site Web. Ceci est offert gratuitement sous une licence MIT modifiée afin que vous puissiez l'utiliser en l'état ou le modifier pour vos propres besoins.
Cette section contient des instructions pour déployer la recherche Kendra sur votre site Web. Vous utilisez une appli-

cation de démonstration Node.js qui s'exécute localement sur votre ordinateur. Cet exemple est basé sur un environnement macOS.

Pour exécuter cette démonstration, vous avez besoin des composants suivants :

- Npm (Node.js) ;
- Informations d'identification IAM d'un utilisateur disposant des autorisations appropriées pour utiliser Kendra.

Téléchargez amazon_aws-kendra-sample-app-master.zip https://aws-ml-blog.s3.amazonaws.com/artifacts/kendra-docs/aws-kendra-sample-app-master.zip et décompressez le fichier.
Ouvrez une fenêtre de terminal et accédez au dossier aws-kendra-sample-app-master.

```
cd /{folder path}/aws-kendra-sample-app-master
```

Créez une copie du fichier .env.development.local.example sous la forme .env.development.local.

```
cp .env.development.local.example .env.development.local
```

Modifiez le fichier .env.development.local et ajoutez les paramètres de connexion suivants :

```
REACT_APP_INDEX — Votre ID d'index Amazon Kendra (vous pouvez trouver ce numéro sur la page d'accueil de l'index)
REACT_APP_AWS_ACCESS_KEY_ID — Votre clé d'accès à votre compte
REACT_APP_AWS_SECRET_ACCESS_KEY — Clé d'accès secrète de votre compte
REACT_APP_AWS_SESSION_TOKEN — Laissez-le vide
REACT_APP_AWS_DEFAULT_REGION — Région utilisée pour déployer l'index Kendra (par exemple, us-east-1)
```

Enregistrez les modifications. Installez les dépendances Node.js :

```
npm install
```

Lancez le serveur de développement local :

```
npm start
```

Voir l'application de démonstration à l'adresse http://localhost:3000/. Vous devriez voir la capture d'écran suivante. **Figure 18**
Entrez la même question que celle que vous avez utilisée pour tester la FAQ : "How do I sign up for the Amazon Prime free Trial?". **Figure 19**

Nettoyage

Pensez à supprimer les ressources que vous avez créées : l'index Kendra, le compartiment S3 et les rôles IAM correspondants. **Figure 20**

Conclusion

Nous avons vu comment utiliser Kendra pour déployer un service de recherche d'entreprise. Vous pouvez utiliser Kendra pour améliorer l'expérience de recherche dans votre entreprise, optimisée par ML. Vous pouvez activer la recherche rapide de vos documents en utilisant un langage naturel, sans aucune expérience ML/IA antérieure.

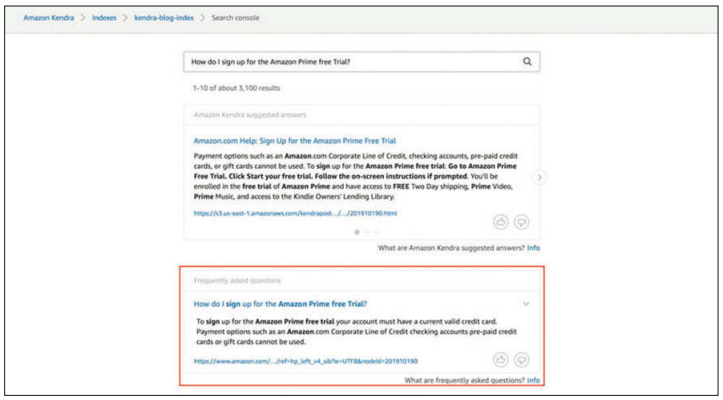


Figure 17

Figure 18

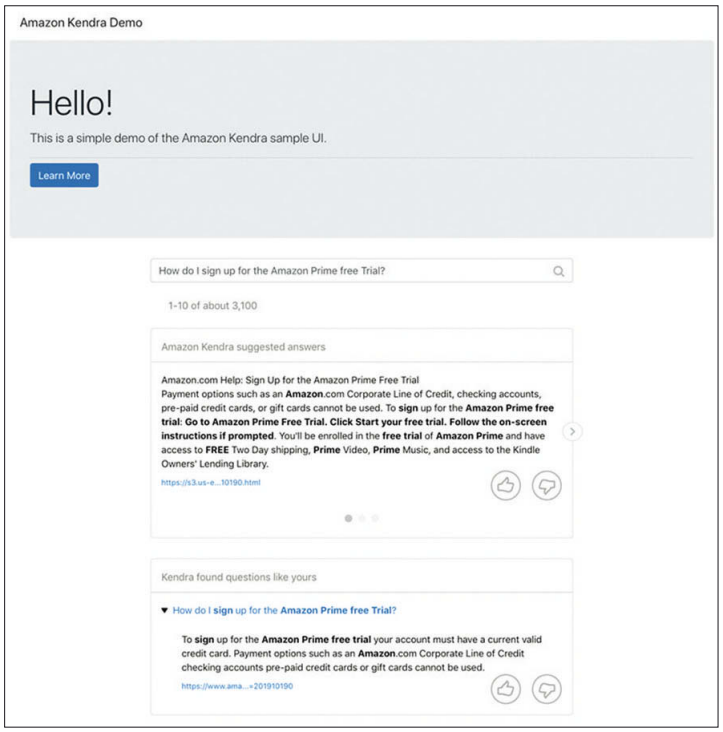


Figure 19

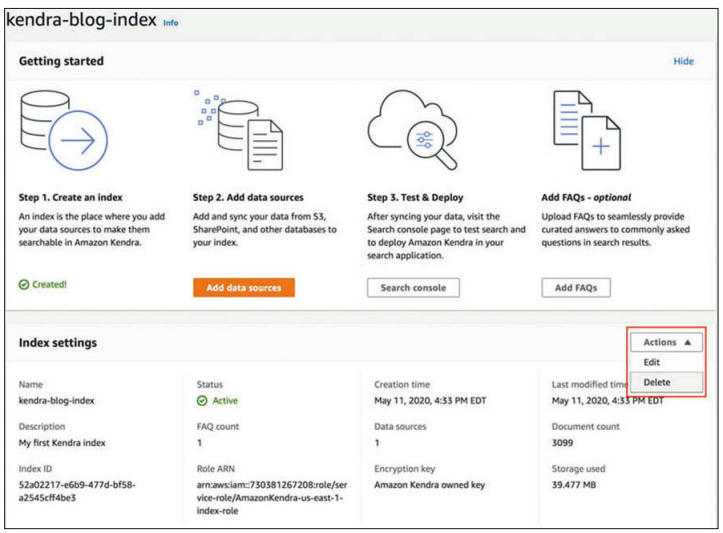


Figure 20

Dorian Richard

Solutions Architect au sein des équipes AWS France où il aide les clients français à innover à travers l'adoption des technologies du cloud en assurant la sécurité de leur infrastructure et de leurs données.

Automatiser l'extraction d'informations avec Amazon Textract et Amazon Comprehend

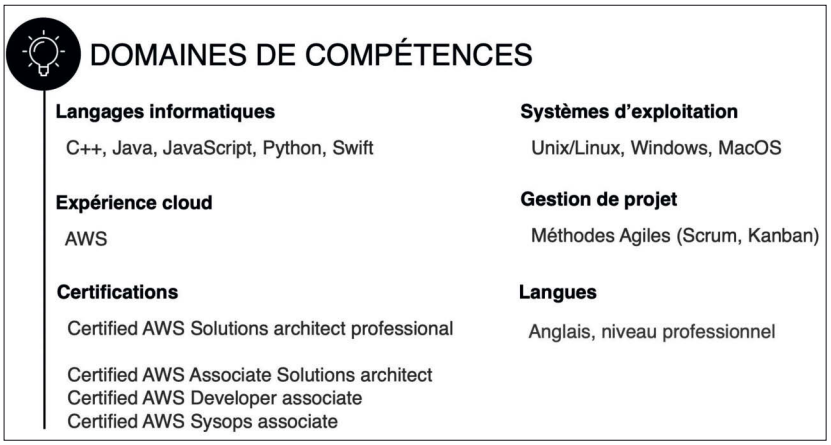
Félicitations ! Vous venez d'être promu Tech lead sur un tout nouveau projet, il vous faut désormais constituer votre équipe de développeurs afin de mettre au point un nouveau système de traitement des dossiers de candidatures à l'université. Le problème c'est qu'avec vos projets actuels, vous ne trouvez pas le temps de faire une présélection des CV des candidats. En tant que bon développeur vous vous demandez alors comment vous pourriez automatiser ce processus de présélection. Vous décidez alors de contacter un Solutions Architect afin de voir les différentes options les plus rapides et efficaces que propose AWS pour construire une telle solution.

Le Solutions Architect vous propose alors deux outils clés en main pour construire ce service le plus rapidement possible :

- Textract est un service d'apprentissage automatique qui permet d'extraire facilement du texte et des données à partir de documents scannés. Amazon Textract va au-delà de la simple reconnaissance de caractères (technologie OCR) pour identifier le contenu des champs des formulaires et les informations stockées dans les tableaux. Cela vous permet d'utiliser Textract pour "lire" instantanément tout type de document et extraire avec précision du texte et des données sans avoir besoin d'être expert en apprentissage automatique.
- Textract a de multiples applications dans une variété de domaines. Par exemple, pour une entreprise chargée du recrutement, Textract peut être utilisé pour automatiser le processus d'extraction des compétences d'un candidat. Les organismes de santé peuvent extraire des informations sur les patients à partir de documents pour répondre à des demandes de remboursement de frais médicaux.

En plus des CV, votre entreprise traite une variété de documents et vous devez parfois extraire des entités du texte non structuré au sein de ces documents. Un document de contrat, par exemple, peut comporter des paragraphes de texte où les noms et autres termes du contrat sont énumérés dans le paragraphe de texte au lieu de figurer sous forme de clé/valeur ou de structure de formulaire. Amazon Comprehend est un service de traitement du langage naturel (NLP) qui peut extraire des phrases clés, des lieux, des noms, des organisations, des événements, des sentiments à partir de textes non structurés, et plus encore. Grâce à la reconnaissance d'entités personnalisées, vous pouvez identifier de nouveaux types d'entités non pris en charge comme l'un des types d'entités génériques prédéfinis. Cela vous permet d'extraire des entités spécifiques à l'entreprise pour répondre à vos besoins. Dans cet article, nous montrons comment extraire des entités personnalisées à partir de documents scannés en utilisant Textract et Comprehend.

Figure 1



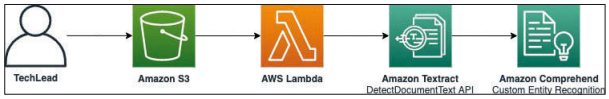
Aperçu du cas d'utilisation

Pour cet article, nous entraînons notre modèle en nous basant sur le jeu de données NER qui contient différentes entités qui nous seront utiles afin d'obtenir des informations telles que les compétences des candidats. Nous utilisons Textract pour extraire le texte de ces CV et Comprehend pour la reconnaissance d'entités personnalisées afin de détecter des compétences telles que AWS, Python et C++ en tant qu'entités personnalisées. Pour ce faire, Comprehend nécessite un étiquetage préalable des données, c'est ce que nous allons traiter en prochaine partie. La capture d'écran suivante montre un extrait de CV type. **Figure 1**

Aperçu de la solution à mettre en place

Le schéma ci-dessous montre une architecture entièrement serverless qui traite les documents entrants pour l'extraction d'entités personnalisées en utilisant Textract et un modèle personnalisé entraîné à l'aide de Comprehend.

Lorsque les documents sont téléchargés vers un compartiment Amazon Simple Storage Service (Amazon S3), il



déclenche une fonction AWS Lambda. AWS Lambda vous permet d'exécuter du code sans avoir à allouer ou gérer de serveurs. La fonction appelle l'API Textract « DetectDocumentText » pour extraire le texte et appelle l'API Comprehend avec le texte extrait pour détecter les entités personnalisées.

La solution se compose de deux parties :

1 **L'entraînement :**

- Extraire le texte des documents PDF en utilisant Textract ;
- Étiqueter les données résultantes en utilisant SageMaker Ground Truth. SageMaker Ground Truth est un service d'étiquetage des données entièrement géré qui facilite la création d'ensembles de données d'entraînement précis pour l'apprentissage automatique ;
- Former à la reconnaissance d'entités personnalisées en utilisant Amazon Comprendre avec les données étiquetées.

2 **L'inférence** :

- Envoyer le document à Textract pour extraction des données ;
- Envoyer les données extraites à Comprehend, le modèle personnalisé pour l'extraction des entités.

Lancer votre pile AWS CloudFormation

Ici, nous utilisons une pile AWS CloudFormation pour déployer la solution et créer les ressources dont elle a besoin pour pouvoir fonctionner. AWS CloudFormation est un service d'infrastructure as code permettant de décrire les ressources que vous souhaitez créer (une VM, une base de données, ...) ainsi que leurs dépendances pour que vous puissiez les lancer et les configurer ensemble sous la forme d'une pile.

Dans notre cas, ces ressources comprennent un compartiment Amazon S3, une instance SageMaker et les rôles AWS de gestion des identités et des accès (IAM) nécessaires.

S3 permet de stocker et de récupérer n'importe quelle quantité de données, à tout moment, de n'importe où sur le Web. Il permet aux développeurs d'accéder à la même infrastructure de stockage de données hautement évolutive, fiable, rapide, peu coûteuse qu'Amazon utilise pour faire fonctionner son propre réseau mondial de sites. Ce service vise à maximiser les avantages d'échelle et à en faire bénéficier les développeurs.

Pour ce faire, les étapes sont :

1 Téléchargez le modèle AWS CloudFormation suivant et enregistrez-le sur votre disque local ;

2 Connectez-vous à la console AWS avec votre nom d'utilisateur et votre mot de passe IAM ;

3 Sur la console AWS CloudFormation, choisissez "**Créer une pile**" ;

4 Sur la page Créer une pile, choisissez Upload a template file et téléchargez le modèle de AWS CloudFormation que vous avez téléchargé ;

5 Choisissez Suivant ;

6 Sur la page suivante, entrez un nom pour la pile. **Figure 2**

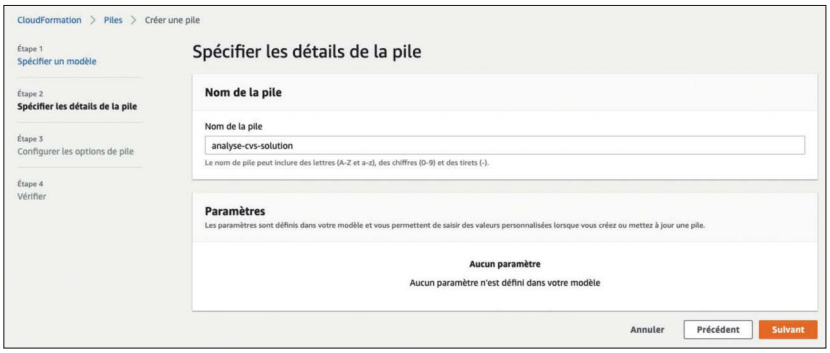


Figure 2

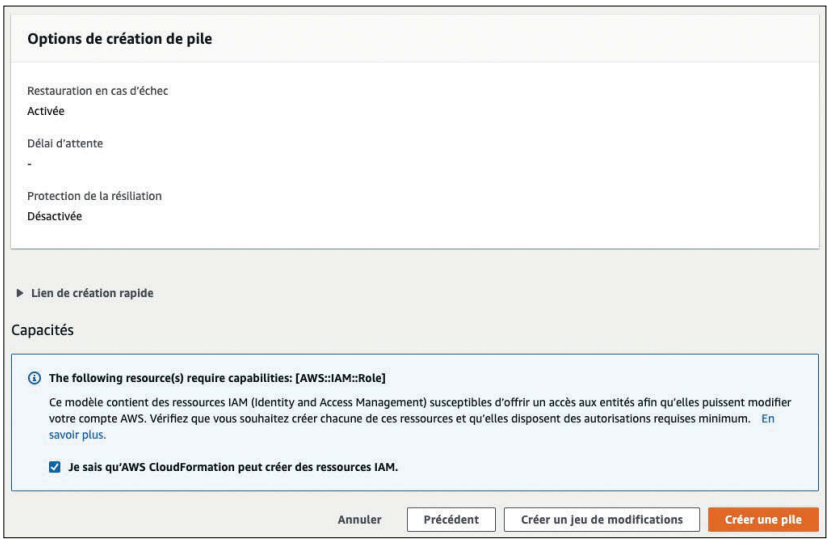


Figure 3

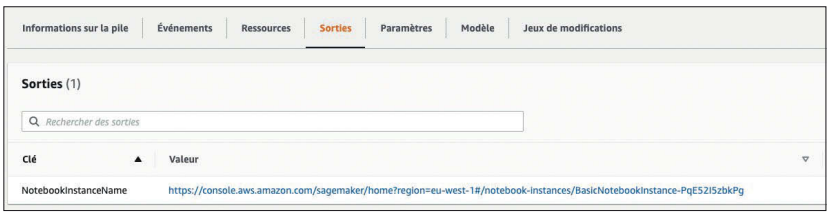


Figure 4

Puis :

1 Laissez tout le reste des réglages par défaut.

2 Sur la page "**Révision**", sélectionnez Je reconnais que AWS CloudFormation peut créer des ressources IAM avec des noms personnalisés.

3 Choisissez "**Créer une pile**". **Figure 3**

Une fois que vous avez lancé la création, veuillez attendre quelques minutes afin que la pile termine son déploiement. Vous pouvez examiner divers événements du processus de création de la pile dans l'onglet Événements.

Une fois la création de la pile terminée, consultez l'onglet Ressources pour voir toutes les ressources que le modèle a créées. Dans l'onglet Sorties de la pile AWS CloudFormation, enregistrez l'URL de l'instance de SageMaker. **Figure 4**

Exécution du flux de travail sur un bloc-notes Jupyter

Afin d'illustrer le processus de traitement des CV, nous utilisons un bloc-notes Jupyter tournant sur une instance créée via Sagemaker. Les bloc-notes Jupyter permettent de réaliser

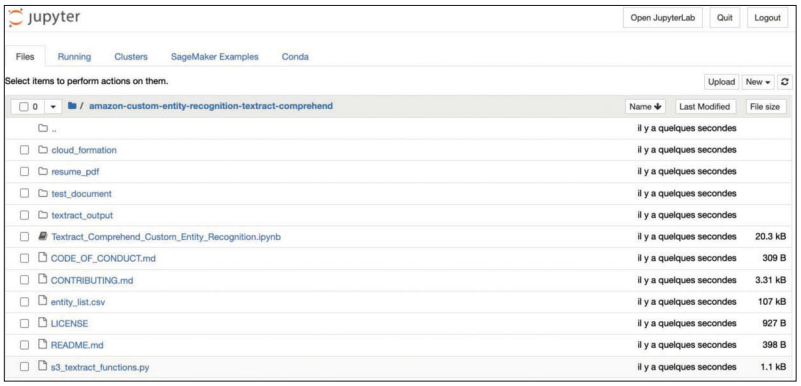
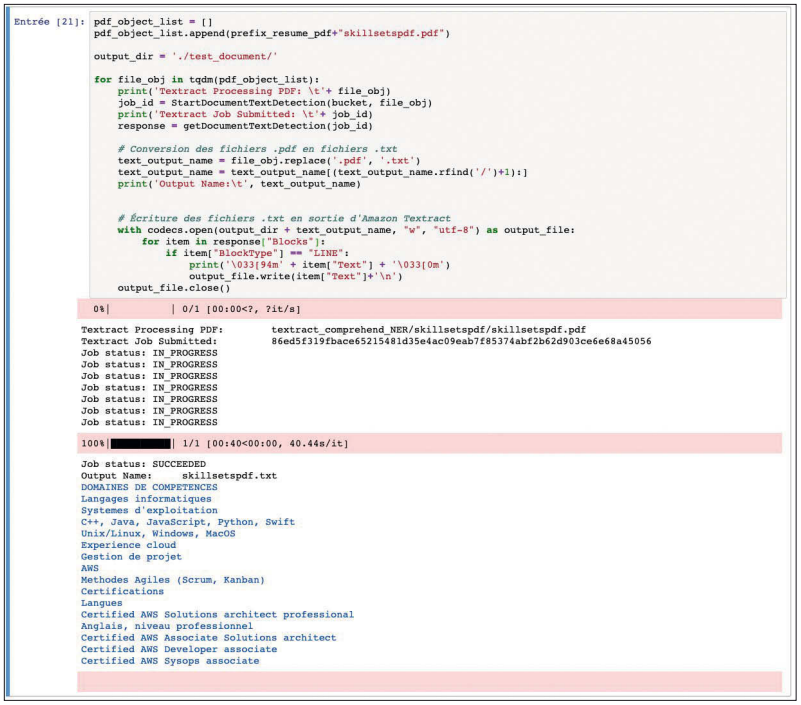


Figure 5

```
Entrée [3]: # Téléverser les CVs au format PDF vers Amazon S3
pdfResumeFileList = glob.glob("./resume_pdf/*.pdf")
prefix_resume_pdf = prefix + "/resume_pdf/"

for filePath in tqdm(pdfResumeFileList):
    file_name = os.path.basename(filePath)
    boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix_resume_pdf, file_name)).upload_file(filel

resume_pdf_bucket_name = 's3://'+bucket+'/'+prefix+'/'+'resume_pdf/'
print('Uploaded Resume PDFs :\t', resume_pdf_bucket_name)

100%|██████████| 220/220 [04:09<00:00,  1.14s/it]

Uploaded Resume PDFs :   s3://sagemaker-eu-west-1-127091078979/textract_comprehend_NER/resume_pdf/
```

Figure 6



```
Entrée [21]: pdf_object_list = []
pdf_object_list.append(prefix_resume_pdf+"skillsetspdf.pdf")

output_dir = './test_document/'

for file_obj in tqdm(pdf_object_list):
    print('Textract Processing PDF: \t'+ file_obj)
    job_id = StartDocumentTextDetection(bucket, file_obj)
    print('Textract Job Submitted: \t'+ job_id)
    response = getDocumentTextDetection(job_id)

    # Conversion des fichiers .pdf en fichiers .txt
    text_output_name = file_obj.replace('.pdf', '.txt')
    text_output_name = text_output_name[(text_output_name.rfind('/')+1):]
    print('Output Name:\t', text_output_name)


    # Écriture des fichiers .txt en sortie d'Amazon Textract
    with codecs.open(output_dir + text_output_name, "w", "utf-8") as output_file:
        for item in response["Blocks"]:
            if item["BlockType"] == "LINE":
                print('\033[94m' + item["Text"] + '\033[0m')
                output_file.write(item["Text"]+'\n')
    output_file.close()

  0%|          | 0/1 [00:00<?, ?it/s]

Textract Processing PDF:        textract_comprehend_NER/skillsetspdf/skillsetspdf.pdf
Textract Job Submitted:         86ed5f319fbace65215481d35e4ac09eab7f85374abf2b62d903ce6e68a45056
Job status: IN_PROGRESS
Job status: IN_PROGRESS
Job status: IN_PROGRESS
Job status: IN_PROGRESS
Job status: IN_PROGRESS
Job status: IN_PROGRESS
Job status: IN_PROGRESS
Job status: IN_PROGRESS

100%|██████████| 1/1 [00:40<00:00, 40.44s/it]

Job status: SUCCEEDED
Output Name:    skillsetspdf.txt
DOMAINES DE COMPETENCES
Langages informatiques
Systemes d'exploitation
C++, Java, JavaScript, Python, Swift
Unix/Linux, Windows, MacOS
Experience cloud
Gestion de projet
AWS
Methodes Agiles (Scrum, Kanban)
Certifications
Langues
Certified AWS Solutions architect professional
Anglais, niveau professionnel
Certified AWS Associate Solutions architect
Certified AWS Developer associate
Certified AWS Sysops associate
```

Figure 7

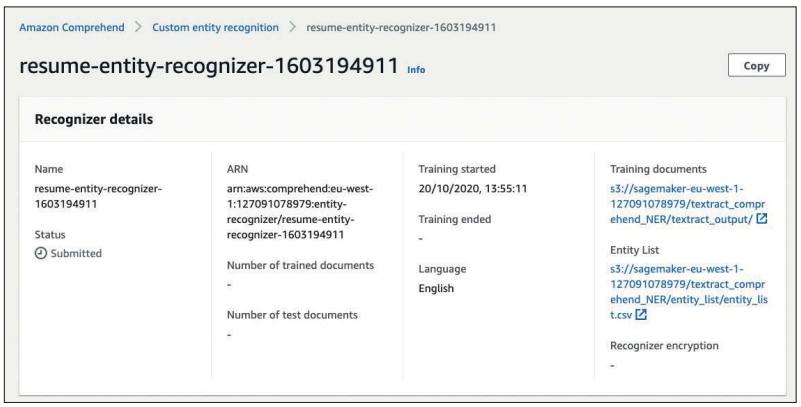


Figure 8

des programmes contenant à la fois du texte en markdown et du code en Julia, Python, R... Ces blocs-notes sont utilisés en science des données pour explorer et analyser les données.

Pour exécuter votre flux de travail, suivez les étapes suivantes :

1. Ouvrez l'URL de l'instance de SageMaker que vous avez créée à l'étape précédente ;
2. Dans le menu déroulant Nouveau, choisissez Terminal ;
3. Sur le terminal, clonez le contenu du repository GitHub *Sagemaker ; git clone* URL. (https://github.com/aws-samples/amazon-custom-entity-recognition-textract-comprehend).

Vous pouvez vérifier la structure des dossiers (voir la capture d'écran suivante). **Figure 5**

1. Ouvrez le bloc-notes: *Textract_Comprehend_Custom_Entity_Recognition.ipynb* ;
2. Exécutez chacune des cellules du bloc-notes.

Passage en revue du code

Téléverser le document vers un compartiment Amazon S3. **Figure 6**

Les PDF sont maintenant prêts pour que Textract puisse effectuer l'OCR. Démarrez le processus avec un appel asynchrone à l'API StartDocumentTextDetection. **Figure 7**

Dans cet article nous traitons deux CV au format PDF à des fins de démonstration, mais vous pouvez traiter les 220 CV si nécessaire. Les résultats ont tous été traités et sont prêts à être utilisés.

Comme nous devons former un modèle de reconnaissance d'entités personnalisé avec Comprehend (comme pour tout modèle ML), nous avons besoin de données destinées à l'entrainement. Dans cet article, nous utilisons Ground Truth pour étiqueter nos entités. Par défaut, Comprehend peut reconnaître des entités comme la personne, le titre et l'organisation. Pour plus d'informations, voir Détecter les entités. Pour démontrer la capacité de reconnaissance d'entités personnalisées, nous nous concentrons sur les compétences des candidats en tant qu'entités à l'intérieur de ces CV. Nous avons les données étiquetées de Ground Truth. Ces données sont disponibles dans le répertoire GitHub (voir : entity_list.csv).

Nous avons maintenant nos données brutes et étiquetées et sommes prêts à former notre modèle. Pour démarrer le processus, utilisez l'appel API *create_entity_recognizer*. Lorsque le travail de formation est lancé, vous pouvez voir le « reconnaisseur » en cours d'entraînement sur la console de Comprehend. **Figure 8**

Nous avons préparé un petit échantillon de texte pour tester le nouvel outil de reconnaissance des entités personnalisées. Nous effectuons la même étape pour effectuer l'OCR, puis nous téléchargeons la sortie de Textract sur S3 et nous lançons un travail de reconnaissance personnalisé. Vous pouvez suivre le statut de la tâche via la console : **Figure 9**

Lorsque le travail d'analyse est terminé, vous pouvez télécharger le résultat et voir les résultats. Pour cet article, nous avons converti le résultat au format JSON puis nous l'avons présenté au format tableau pour des raisons de lisibilité.

Nous retrouvons ainsi trois informations essentielles : un indice de confiance pour chacun des résultats extraits, la compétence détectée ainsi que le type d'entité (ici une compétence). **Figure 10**

Traiter les dossiers de candidatures à l'université, simplifier un processus long et fastidieux

Maintenant que votre équipe est constituée avec un minimum d'effort, l'idée est de mettre en place un système automatique de traitement des dossiers de candidature à l'université. Le but serait de concevoir un système capable d'extraire les informations sur les formulaires d'inscription. Le problème est que les dossiers d'inscriptions sont remplis de manière manuscrite par les étudiants et que cela complique le processus d'extraction.

Vous vous tournez alors vers votre architecte cloud préféré afin d'avoir ses conseils. Après de nombreuses discussions, l'architecte vous présente une nouvelle fonctionnalité de Textract permettant de faire de la reconnaissance d'écriture manuscrite.

Textract peut en effet extraire du texte venant de divers documents écrits à la main. Il peut également extraire du texte situé dans des formulaires ou encore des tableaux. Ce dernier prend en charge plusieurs formats de documents comme JPG, PNG ou encore PDF.

Formulaire type reçu par l'université

Figure 11

Toutes les informations sont saisies de manière manuscrite. Pour le moment ce travail de saisie dans le système de l'université est effectué par un agent administratif. L'idée est donc de créer une solution capable d'ingérer les formulaires et d'extraire les informations pour faciliter le processus d'inscription en limitant les actions manuelles.

Appel à l'API Textract

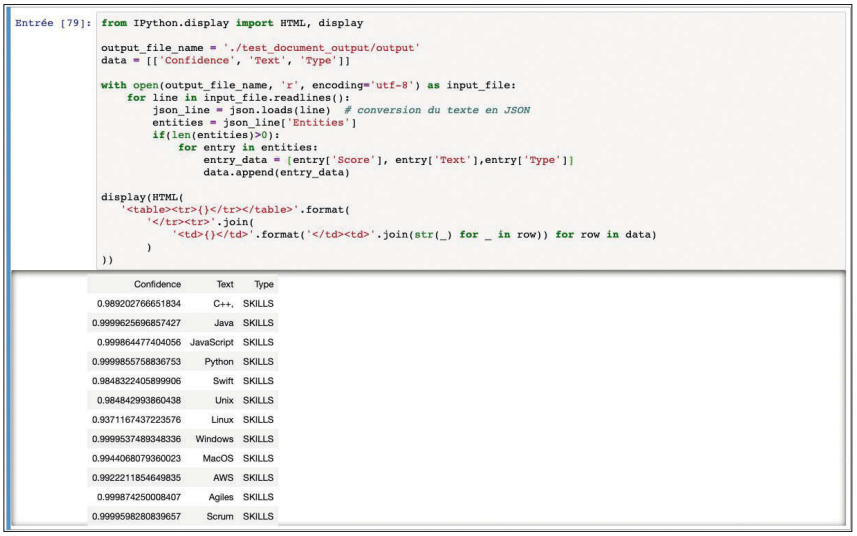
Afin de lancer un flux de travail permettant la reconnaissance des écrits manuscrits, nous pouvons réutiliser le code précédemment mis en place dans le bloc-notes Jupyter. Ce dernier nous permet d'obtenir les résultats suivants (en bleu ci-dessous). **Figure 12**

Nous pouvons voir les résultats suivants :

- 'Nom:' 'RICHARD'
- 'Prénom': 'Dorian'
- 'Date de naissance' : '19/11/1995'
- 'Adresse mail' : 'N/A'
- 'Téléphone': '03.42.44.12.13'
- 'Adresse postale': '11 r.u.e de la liberté Paris'
- 'Formation ciblée' : 'Informatique'
- 'Numéro d'étudiant' : 'N/A'

Certaines chaînes de caractères sont moins bien détectées que d'autres, chose qu'il est facile de normaliser avant de stocker les données dans votre backend. Par exemple, la champs adresse postale ('11 r.u.e de la liberté Paris') ou encore le champ adresse mail sont mal extraits : ceci signifie que le score de confiance pour ces prédictions particulières est assez faible, nous pouvons améliorer nos résultats en signalant cette imprécision pour une révision manuelle. Comme toutes les autres chaînes sont correctes, un contrôleur humain pourrait corriger l'erreur en quelques secondes, ce qui représente toujours une énorme amélioration. Pour ce faire, nous pouvons utiliser la fonctionnalité Human review faisant appel au service Amazon Augmented AI. **Figure 13**

Figure 9



```
Entrée [79]: from IPython.display import HTML, display

output_file_name = './test_document_output/output'
data = [['Confidence', 'Text', 'Type']]

with open(output_file_name, 'r', encoding='utf-8') as input_file:
    for line in input_file.readlines():
        json_line = json.loads(line)  # conversion du texte en JSON
        entities = json_line['Entities']
        if(len(entities)>0):
            for entry in entities:
                entry_data = [entry['Score'], entry['Text'],entry['Type']]
                data.append(entry_data)

display(HTML(
    '<table><tr>{}</tr></table>'.format(
        '</tr><tr>'.join(
            '<td>{}</td>'.format('</td><td>'.join(str(_) for _ in row)) for row in data)
        )
))
```

Confidence	Text	Type
0.989202766651834	C++,	SKILLS
0.9999625696857427	Java	SKILLS
0.999864477404056	JavaScript	SKILLS
0.9999855758836753	Python	SKILLS
0.9848322405899906	Swift	SKILLS
0.984842993860438	Unix	SKILLS
0.9371167437223576	Linux	SKILLS
0.9999537489348336	Windows	SKILLS
0.9944068079360023	MacOS	SKILLS
0.9922211854649835	AWS	SKILLS
0.999874250008407	Agiles	SKILLS
0.9999598280839657	Scrum	SKILLS

Figure 10

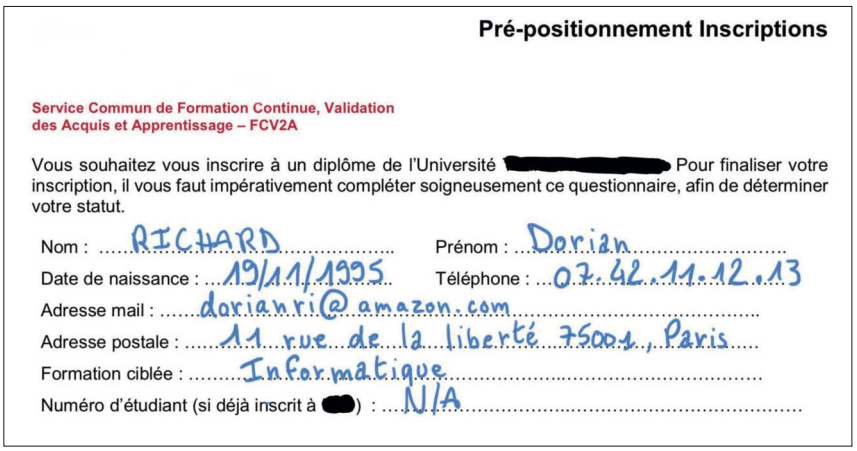


Figure 11

```
Textract Processing PDF:        textract_comprehend_NER/test_document/dossier_inscription.pdf
Textract Job Submitted:         f5bf2e2d26974d021e2a81ba8e94c06f5092d29de3b691bb68516e5f81c23009
Job status: IN_PROGRESS

100%|[redacted]| 1/1 [00:10<00:00, 10.32s/it]

Job status: SUCCEEDED
Output Name:      dossier_inscription.txt
Pré-positionnement Inscriptions
Service Commun de Formation Continue, Validation
des Acquis et Apprentissage - FCV2A
Vous souhaitez vous inscrire à un diplôme de l'Université
Pour finaliser votre
inscription, il vous faut impérativement compléter soigneusement ce questionnaire, afin de déterminer
votre statut.
Nom :
RICHARD
Prénom : Dorian
Date de naissance : 19/11/1995
Téléphone : 03.42.44.12.13
Adresse mail :
Adresse postale
: 11 r.u.e de la liberte Paris
Formation ciblée :
Informatique
Numéro d'étudiant (si déjà inscrit à
)
:
N/A
```

Figure 12

Boutique Programmez!

Les anciens numéros de PROGRAMMEZ!

Le magazine des développeurs

226

236

238

239

240

241

HS 01 été 2020

242

243

Tarif unitaire 6,5 € (frais postaux inclus)

TECHNOSAURES

Le magazine à remonter le temps !

N°1

N°2

N°3

N°4

N°5

Prix unitaire : **7,66 €** (frais postaux inclus)

N°4 Standard **10 €**

N°4 Deluxe **15 €**

Histoire de la micro-informatique 1973-2007

Volume 1

12,99 €

(frais postaux inclus)

☐ 226	: __ ex	☐ 241	: __ ex
☐ 236	: __ ex	☐ HS 01	: __ ex
☐ 238	: __ ex	☐ 242	: __ ex
☐ 239	: __ ex	☐ 243	: __ ex
☐ 240	: __ ex		

soit __ exemplaires x 6,50 € = ____ €

Technosaures ☐ N°1 ☐ N°2 ☐ N°3 ☐ N°5
soit __ exemplaires x 7,66 € = ____ €
☐ N°4 Deluxe..... 15 €
☐ N°4 Standard ..... 10 €
☐ Histoire de la Micro-informatique ..... 12,99 €

..... ____ €

Commande à envoyer à :
Programmez!
57 rue de Gisors
95300 Pontoise

soit au **TOTAL** = ____ €

☐ M. ☐ Mme ☐ Mlle Entreprise : ____ Fonction : ____

Prénom : ____ Nom : ____

Adresse : ____

Code postal : ____ Ville : ____

Règlement par chèque à l'ordre de Programmez ! I Disponible sur www.programmez.com

Offres pouvant s'arrêter à tout moment, sans préavis I I ca 03ésep PR

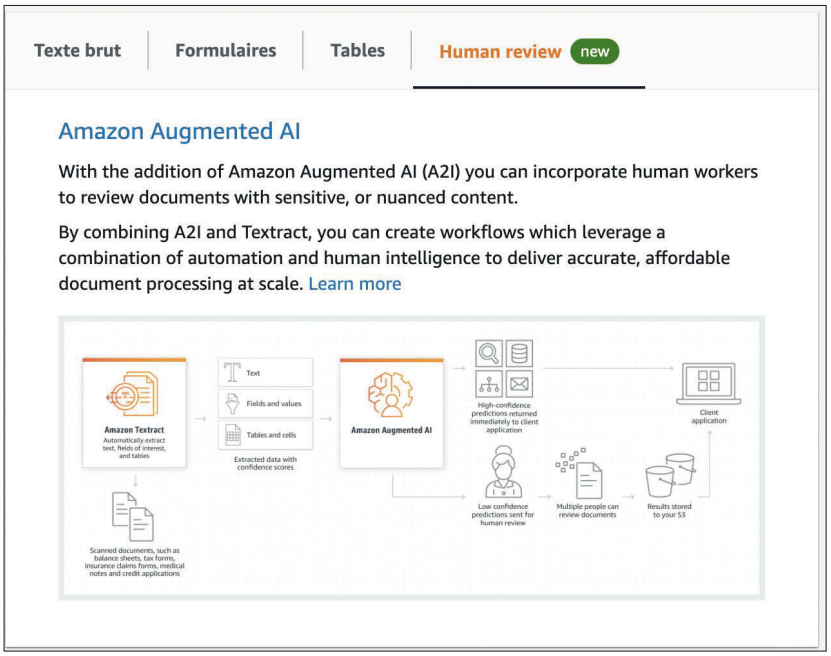


Figure 13

Amazon Augmented AI

Augmented AI, ou Amazon A2I, est maintenant disponible pour aider les développeurs à améliorer la précision de leurs modèles d'apprentissage automatique par le biais de la supervision humaine. A2I permet à tous les développeurs de bénéficier d'une vérification humaine, supprime les tâches lourdes liées au développement de systèmes d'évaluation humaine. Par exemple, A2I offre aux développeurs des flux de travail préconfigurés, si leurs prédictions d'apprentissage automatique nécessitent une vérification humaine supplémentaire. Certaines applications d'apprentissage automatique, comme celles utilisées pour scanner des documents financiers ou effectuer la reconnaissance faciale, exigent des niveaux de précision particulièrement élevés. Pour atteindre cette précision, les développeurs sont invités à tester leurs prédictions d'apprentissage auprès d'autres participants, soit par un auditeur tiers, soit en interne.

Cela étant dit, il peut être long et coûteux de construire et de maintenir ces "systèmes de vérification humains". Les examens humains sont difficiles à mettre en place et coûteux à développer et à exploiter à grande échelle, comprenant souvent plusieurs étapes de flux de travail, des logiciels personnalisés pour gérer les tâches et les résultats des évaluations humaines, ainsi que le recrutement et la gestion de groupes d'examinateurs. Cela conduit les développeurs à passer plus de temps à gérer le processus d'évaluation humaine qu'à créer l'application souhaitée ou à renoncer aux évaluations humaines, ce qui se traduit par une moindre confiance dans la livraison d'applications d'apprentissage automatique. Au lieu de cela, A2I automatise le processus d'évaluation humaine pour les services AWS, y compris l'identification des images et la reconnaissance de texte, ainsi que la plateforme d'apprentissage automatique SageMaker. Elle se compose de 60 flux de travail préconstruits pour des tâches spécifiques telles que la transcription de la parole, l'analyse d'images et l'évaluation de contenus.

Les développeurs peuvent lancer un processus d'évaluation humaine pour toute prédiction d'apprentissage automatique en dessous d'un certain niveau de confiance. Nous pouvons décider du nombre d'évaluateurs pour chaque prédiction et choisir comment ces évaluateurs peuvent être soutenus s'ils doivent être assistés en interne par des partenaires spécialisés AWS ou pris en charge par Mechanical Turk.

Conclusion

L'apprentissage automatique et l'intelligence artificielle de manière générale permettent aux organisations d'être davantage agiles. Ils peuvent vous aider à automatiser les tâches manuelles pour améliorer l'efficacité. Dans cet article, nous avons fait la démonstration d'une architecture de bout en bout pour extraire des entités telles que les compétences d'un candidat sur son CV en utilisant Textract et Comprehend. Cet article vous a montré comment utiliser Textract pour faire de l'extraction de données saisies de manières informatisées et même manuscrites. Nous avons aussi vu comment utiliser Comprehend pour former un "reconnaisseur" d'entités personnalisées à partir de votre propre ensemble de données afin de reconnaître des entités personnalisées. Vous pouvez appliquer ce processus à une variété de secteurs, tels que les soins de santé et les services financiers.

Ressources

https://github.com/aws-samples/amazon-custom-entity-recognition-textract-comprehend/tree/master/cloud_formation
https://docs.aws.amazon.com/comprehend/latest/dg/how-entities.html
https://github.com/aws-samples/amazon-custom-entity-recognition-textract-comprehend
https://aws.amazon.com/fr/blogs/machine-learning/developing-ner-models-with-amazon-sagemaker-ground-truth-and-amazon-comprehend/
https://aws.amazon.com/blogs/machine-learning/developing-ner-models-with-amazon-sagemaker-ground-truth-and-amazon-comprehend/
https://docs.aws.amazon.com/fr_fr/sagemaker/latest/dg/a2i-getting-started.html

Amazon Forecast et Amazon Personalize

Ségolène Dessertine Panhard
Senior Data Scientist au sein du Machine Learning Solutions Lab d'AWS.

Amazon Personalize (Personalize) est un service de recommandation en temps réel. Amazon Forecast (Forecast) propose des prévisions à partir de séries temporelles. Basés sur la même technologie développée au fil des années pour Amazon.com, Forecast et Personalize permettent aux développeurs n'ayant aucune expérience préalable en machine learning d'intégrer facilement des prévisions précises et des capacités de recommandation dans leurs applications.

Forecast et Personalize fonctionnent de la même manière : l'utilisateur a juste à fournir ses données au format CSV puis ces deux services effectuent et accélèrent le travail nécessaire pour concevoir, former et déployer un modèle de machine learning notamment grâce à la fonctionnalité d'AutoML (un processus qui automatise les tâches complexes de machine learning).

Ces deux services sont accessibles via la console AWS, le SDK AWS pour Python ou des notebooks Jupyter (pour les utilisateurs expérimentés). Dans cet article, nous les utiliserons via la console pour mettre en évidence leurs similarités de leur fonctionnement.

Créez votre modèle de prévision deep learning customise en 5 clicks avec Amazon Forecast

Forecast est un service entièrement géré (fully managed) qui utilise des algorithmes statistiques et de machine learning pour fournir des prévisions à partir de séries temporelles. Basé sur la même technologie que celle utilisée par Amazon.com, Forecast utilise des algorithmes de pointe pour prévoir les futures données de séries chronologiques en se basant sur des données historiques, et ne nécessite aucune expérience en machine learning. Le but de cet article est de montrer comment créer en quelques clicks un modèle de prévisions.

Figure 1

Nous utilisons ici un jeu de données représentant la consommation des ménages en électricité. Les données sont agrégées par heure et sont au format CSV. Ci-dessous **(Figure 2)**, les premières lignes du fichier où l'on peut voir l'heure et la date, la consommation d'énergie et le numéro du client.

Nous allons créer facilement en 5 clicks un modèle de prédiction et générer des prévisions en utilisant la console d'Amazon Forecast.

Click 1 : Création des pipelines de données

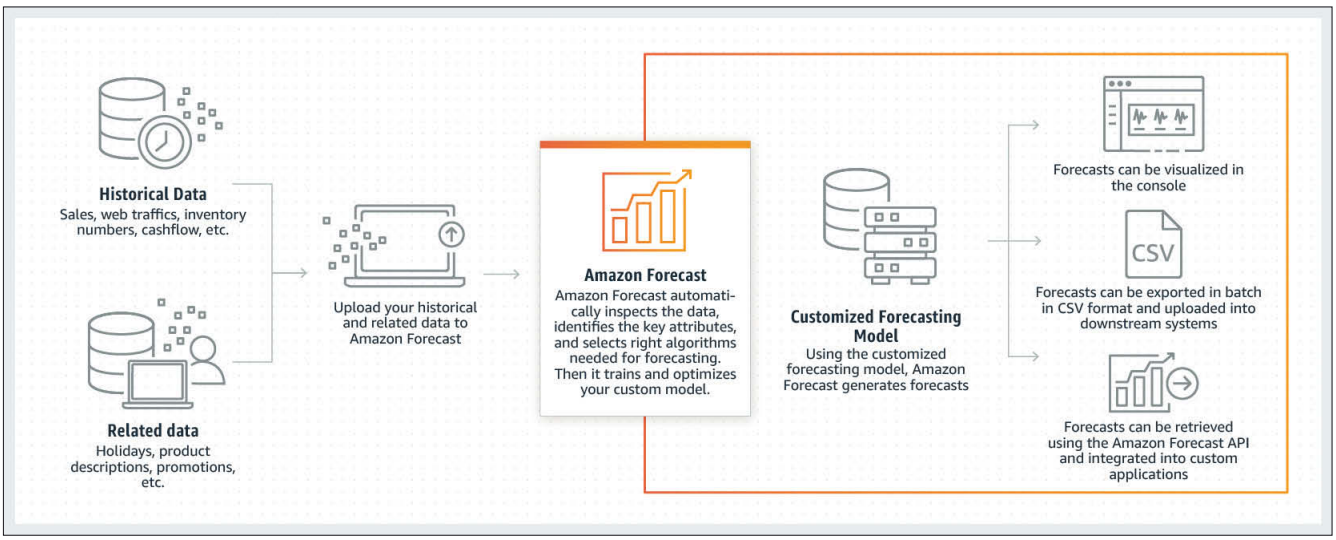
Dans la console, la première étape est de créer un groupe de jeux de données. Ceux-ci regroupent des jeux de données lies au même projet. **Figure 3**

On peut sélectionner un domaine de prévision pour le groupe de jeux de données. Chaque domaine couvre un cas d'utilisation spécifique, tel que la vente au détail, la planification des stocks ou le trafic web. Dans cet exemple, nous utilisons un domaine personnalisé CUSTOM qui couvre tous les cas d'uti-

Figure 2

```
2014-01-01 01:00:00,38.34991708126038,client_12
2014-01-01 02:00:00,33.5820895522388,client_12
2014-01-01 03:00:00,34.41127694859037,client_12
2014-01-01 04:00:00,39.800995024875625,client_12
2014-01-01 05:00:00,41.044776119402975,client_12
```

Figure 1

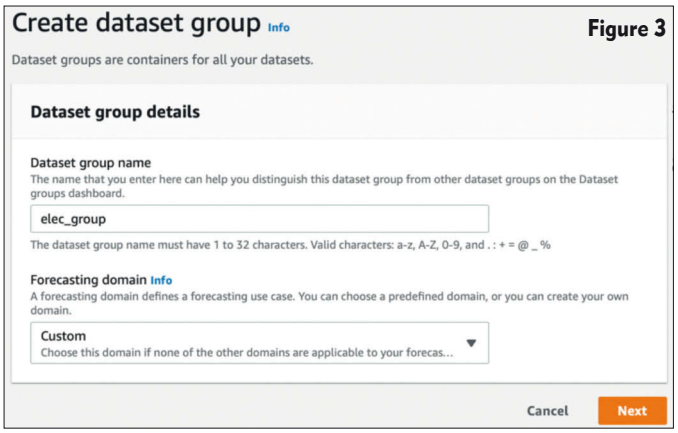


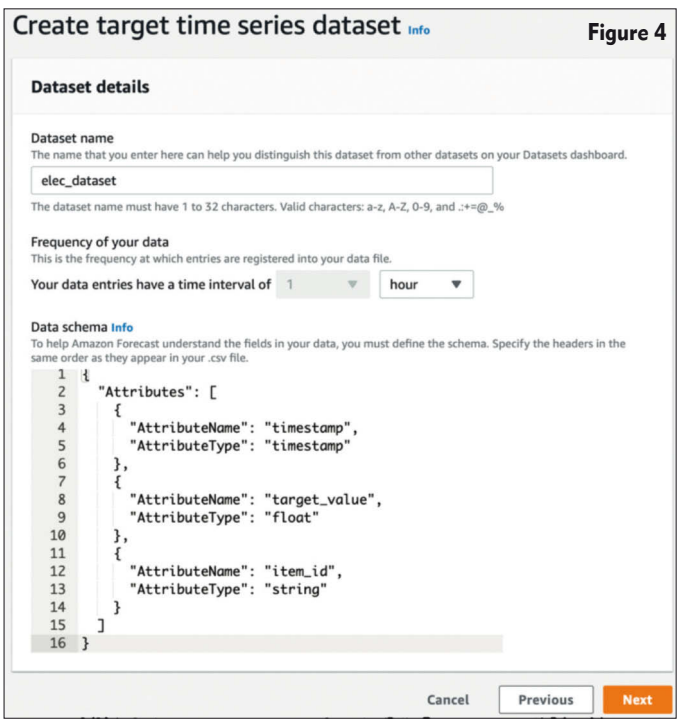
lisation qui n'entrent pas dans les catégories prédéfinies.
Ensuite, nous créons un jeu de données. Les données que nous utilisons sont agrégées par heure, donc nous définissons la fréquence de nos données étant égale à une heure. Nous définissons le schéma de nos données conformément au type de données que nous utilisons. **Figure 4**
Nous transférons ensuite nos séries temporelles depuis S3 dans le jeu de données d'Amazon Forecast. Il faut créer un rôle de IAM pour donner l'accès du bucket S3 à Amazon Forecast. Après avoir indiqué à Amazon Forecast dans quel bucket S3 il faut chercher les séries temporelles, on peut lancer l'importation. **Figure 5**
Le tableau de bord de Forecast donne une vue d'ensemble du processus. Les données des Target Time Series sont en cours d'importation, et on peut ajouter en option :

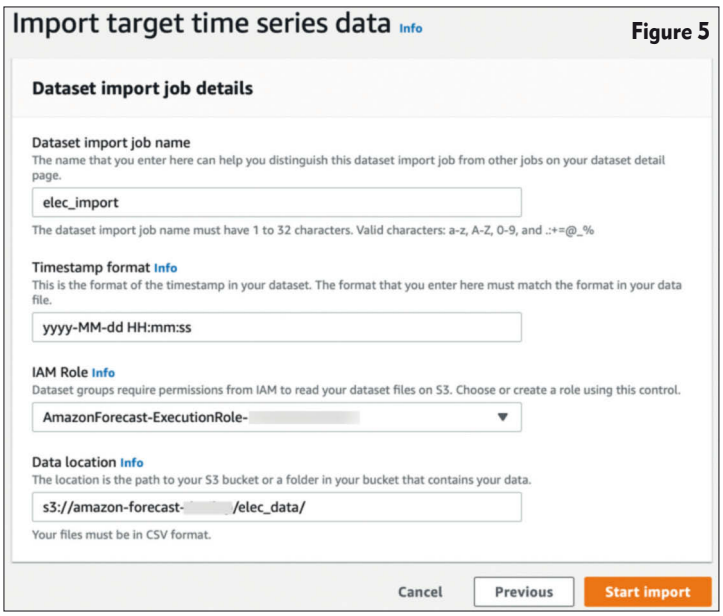
- **Des metadata** : ce sont des informations complémentaires à propos des éléments sur lesquels nous voulons faire des prévisions ; par exemple, la couleur des articles dans un scénario de vente au détail, ou le type de ménage (est-ce un appartement ou une maison individuelle ?) pour notre exemple axé sur l'électricité.
- **Des related time series** : ce sont des séries temporelles complémentaires qui n'incluent pas la variable cible que je veux prédire, mais qui peuvent aider à améliorer le modèle ; par exemple, les prix et les promotions utilisés par une société de commerce électronique sont probablement liés aux ventes réelles. **Figure 6**

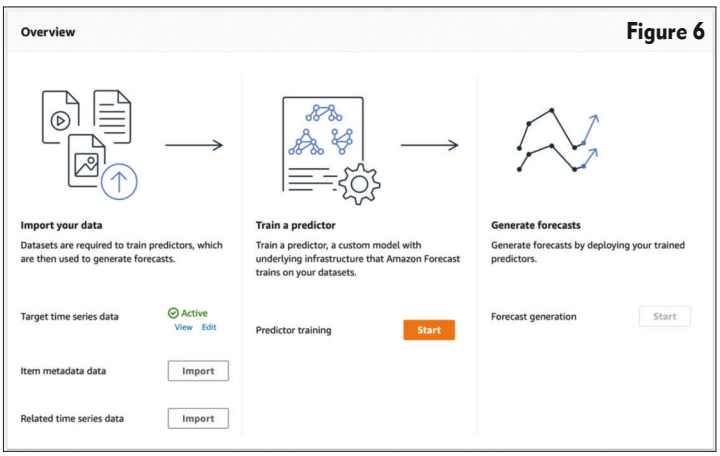
Click 2 : Choix de l'algorithme (AutoML ou algorithm supporte par Amazon Forecast)

Nous n'ajoutons pas de données supplémentaires ici. Dès que le jeu de données est importé, nous entraînons un Predictor qui va être ensuite utilisé pour générer des prévisions. Il faut donner un nom au prédicteur, puis sélectionner l'horizon de prévision (qui est ici de 24 heures) ainsi que la fréquence à laquelle les prévisions sont générées.
Pour entraîner le prédicteur, on peut sélectionner un algorithme de ML, comme ARIMA ou DeepAR+. On peut aussi utiliser la fonctionnalité AutoML pour laisser Amazon Forecast évaluer tous les algorithmes et choisir celui qui fonctionne le mieux pour le jeu de données de l'utilisateur. **Figures 7-8**









Click 3 : comparaison des métriques de précision entre les modèles

Après quelques minutes, le prédicteur est actif. Pour comprendre la performance d'un prédicteur, il faut regarder certaines mesures qui sont calculées automatiquement. **Figure 9**
La Quantile Loss (QL) calcule l'écart entre la prévision à un certain quantile et la demande réelle. Elle pondère la sous-estimation et la surestimation en fonction d'un quantile spécifique. Par exemple, une prévision P90, si elle est calibrée, signifie que dans 90 % des cas, la demande réelle est inférieure à la valeur prévue. Ainsi, lorsque la demande s'avère supérieure à la prévision, la perte serait plus importante que l'inverse.

Cilck 4 : Déploiement du modèle

Lorsque le prédicteur est prêt, et que les métriques sont satisfaisantes, on peut générer une prévision. **Figure 10**

Click 5 : Génération de prévisions

Lorsque la prévision est active, on peut faire des requêtes pour obtenir des prévisions. L'ensemble de la prévision peut être exportée sous forme de fichier CSV, ou être interrogée pour obtenir des prévisions spécifiques. On peut ainsi prévoir l'énergie utilisée par un ménage pendant une période donnée. **Figure 11**
Pour chaque timestamp dans l'horizon de la prévision, on obtient un intervalle de valeurs. Les prévisions P10, P50 et P90 ont respectivement une probabilité de 10 %, 50 % et 90 % de satisfaire la demande réelle. La prévision P50 est l'estimation la plus probable de la demande. Les prévisions P10 et P90 donnent un intervalle de confiance de 80 % de ce qui peut se passer dans le futur.
Pour finir, la **Figure 12** représente le processus général des différentes étapes et les API correspondantes qui peuvent être utilisées via le AWS SDK pour construire un modèle de prévision avec Forecast.

Créez votre système de recommandation customise en 5 clicks avec Amazon Personalize

Personalize facilite la conception d'applications capables de fournir une vaste gamme d'expériences de personnalisation, notamment des recommandations de produits spécifiques, des reclassements de produits personnalisés et le marketing direct personnalisé. Personalize est un service de machine learning entièrement géré qui va au-delà des systèmes rigides de recommandations basés sur des règles statiques. Ce service entraîne, affine et déploie des modèles de ML personnalisés afin d'offrir aux clients des recommandations hautement personnalisées, dans des secteurs comme la vente au détail, le multimédia et le divertissement. **Figure 13**
Nous allons construire un système de recommandation pour les films, basé sur les données recueillies dans la base de données MovieLens. L'objectif est de recommander des films qui sont les plus pertinents pour un utilisateur en particulier.
Ci-dessous, les premières lignes du fichier Interactions que nous utilisons après préparation des données (cf. github Personalize dans Ressources) où l'on peut voir USER_ID, ITEM_ID, et le TIMESTAMP. Ce jeu de données Interactions contient une relation n:n (une relation multi-valeur, en utili-

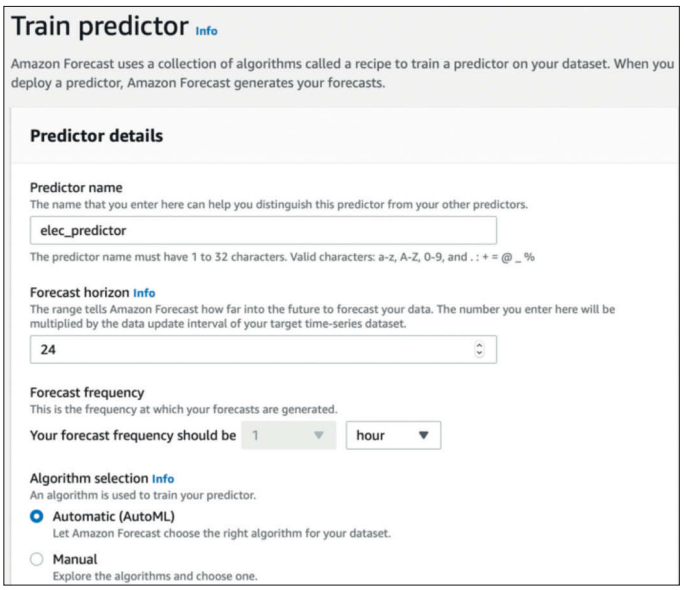
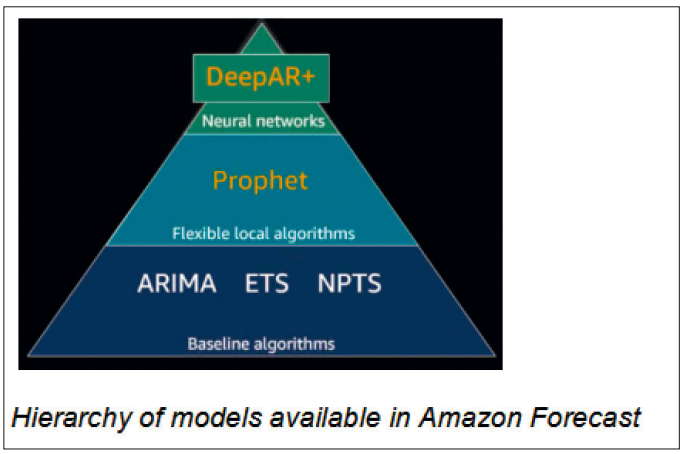


Figure 7



Hierarchy of models available in Amazon Forecast

Figure 8

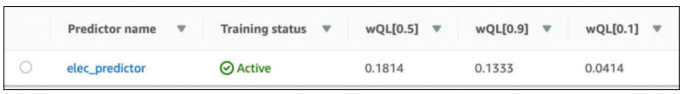


Figure 9

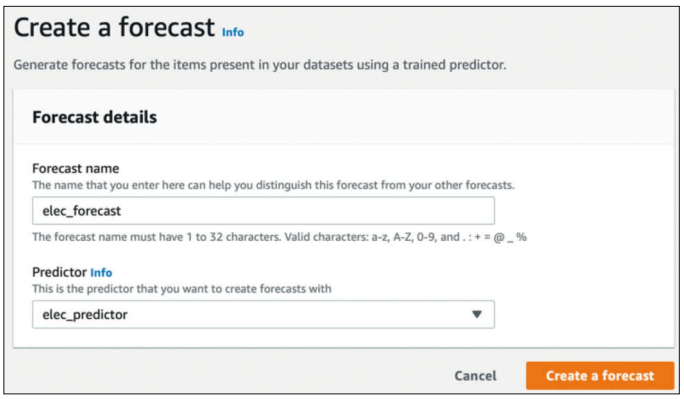


Figure 10

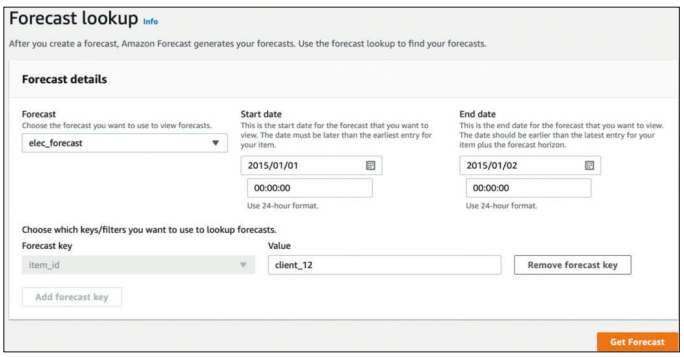


Figure 11

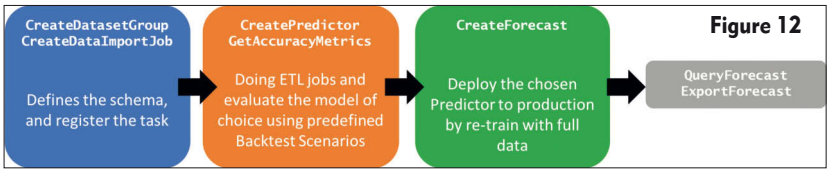


Figure 12

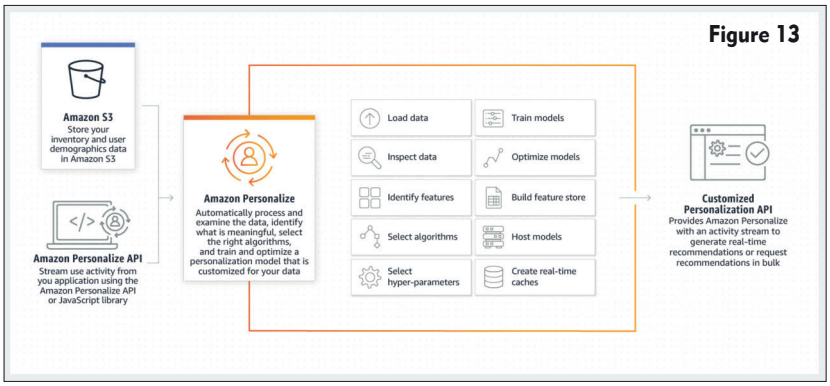


Figure 13

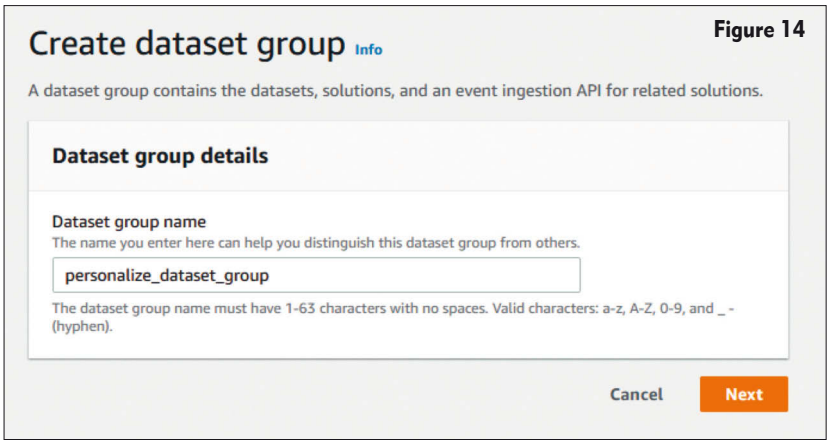
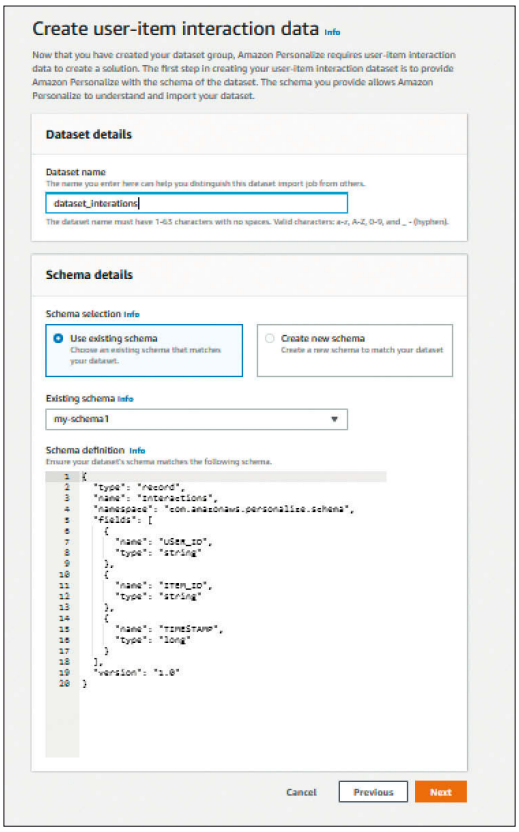


Figure 14

Figure 15



sant les anciens termes de base de données relationnelle) qui associe USER_ID à ITEM_ID. Les interactions peuvent être enrichies avec des jeux de données facultatifs User et Item qui contiennent des données supplémentaires liées par leurs ID. Par exemple, pour un site web de diffusion de films, il peut être utile de connaître la classification d'âge d'un film et l'âge du spectateur et de comprendre quels films ils regardent.

USER_ID	ITEM_ID	TIMESTAMP
298	474	884182806
253	465	891628467
286	1014	879781125
200	222	876042340
122	387	879270459

Click 1 : création des pipelines de données

Comme pour Amazon Forecast, la première étape consiste à créer un groupe de jeux de données qui peut être créé à partir du chargement de données historiques ou à partir de données collectées à partir d'événements en temps réel. Ici nous n'utilisons que des données historiques. **Figure 14**

Une fois que le groupe de jeux de données est créé, il faut créer le jeu de données contenant les interactions entre l'utilisateur et le produit en définissant un schéma pour les données au format Apache Avro pour chaque ensemble de données, ce qui permet à Personalize de comprendre le format de vos données. **Figure 15**

Une fois que les données sont prêtes à être utilisées pour l'entraînement du modèle, l'importation est réalisée.

Le tableau de bord de Personalize donne une vue d'ensemble du processus. **Figure 16**

Click 2 : création de la solution de recommandation

Dans Personalize, un modèle entraîné s'appelle une solution. Chaque solution peut avoir différentes versions en fonction du volume de données sur lesquelles le modèle a été entraîné. Une solution couvre deux domaines : choix du modèle (recette) et l'utilisation des données pour l'entraînement de celui-ci. Au sein de Personalize, des recettes builtin et une base d'algorithmes pour établir le score de popularité/recommandation sont disponibles. Voici quelques-unes des recettes proposées :

- Re-classement personnalisé (recherche)
- Recommandations d'éléments similaires — SIMS
- Réseau neuronal récurrent hiérarchique — HRNN (modélisation interactions utilisateur-élément sur une période donnée)

Ici, nous utilisons la recette aws-user-personalization.

La recette de personnalisation de l'utilisateur (aws-user-personalization) est optimisée pour tous les scénarios de recommandation d'USER_PERSONALIZATION. Lors de la recommandation d'articles, elle utilise l'exploration automatique des articles.

Grâce à l'exploration automatique, Personalize teste automatiquement différentes recommandations d'articles, apprend de la façon dont les utilisateurs interagissent avec ces articles recommandés et renforce les recommandations d'articles qui entraînent un meilleur engagement et une meilleure conver-

sion. Cela améliore la découverte et l'engagement des articles lorsque vous avez un catalogue qui évolue rapidement, ou lorsque de nouveaux articles, tels que des articles d'actualité ou des promotions, sont plus pertinents pour les utilisateurs lorsqu'ils sont récents.

Vous pouvez trouver un équilibre entre la quantité d'éléments à explorer (où les éléments ayant moins de données d'interaction ou de pertinence sont recommandés plus fréquemment) et la quantité d'éléments à exploiter (où les recommandations sont basées sur ce que nous savons ou sur la pertinence). Personalize ajuste automatiquement les futures recommandations en fonction des commentaires implicites des utilisateurs.

Vous pouvez aussi utiliser l'option AutoML, qui exécute l'entraînement de chacune des recettes disponibles à partir des données. Personalize évalue alors la meilleure recette en fonction des métriques de précision. Cela couvre également la modification de certains paramètres pour obtenir de meilleurs résultats (HPO, hyper parameters optimization).

Figure 17

Click 3 : comparaison des métriques de précision entre les modèles

Pour aller plus loin dans l'analyse de la précision de recommandation par Personalize, nous conseillons de lire la documentation de Personalize pour comprendre les métriques ci-dessous. De façon très simplifiée, les valeurs proches de 1 indiquent une plus grande fiabilité.

Personalize vous permet donc de démarrer rapidement votre projet, même si vous n'êtes pas un expert. Cela inclut non seulement la sélection de modèles et l'entraînement, mais aussi la transformation des données nécessaires pour chaque recette et la gestion des instances nécessaires pour l'entraînement du modèle.

Click 4 : création d'une campagne

Après avoir obtenu une version de solution (une recette confirmée et des artefacts formés), il est temps de la mettre en action. Ce n'est habituellement pas une tâche facile, et il y a beaucoup de choses à considérer dans l'hébergement de services ML à grande échelle.

Pour vous aider, Personalize vous permet de déployer une campagne (un moteur d'inférence pour votre recette et les artefacts formés) en tant que moteur de recommandation. La campagne met à disposition une API REST que vous pouvez utiliser pour produire des recommandations.

Click 5 : génération de recommandations

Une fois que la campagne est active, nous pouvons désormais générer des recommandations. **Figure 18**

Personalize est un excellent ajout à l'ensemble de services d'apprentissage automatique AWS. Son approche en deux étapes vous permet d'exécuter rapidement et efficacement votre premier moteur de recommandation et d'offrir de la valeur immédiate à vos utilisateurs finaux ou aux différents métiers de votre entreprise. Vous pouvez, par la suite, exploiter la puissance de Personalize, ce qui vous permettra d'améliorer en continu vos recommandations.

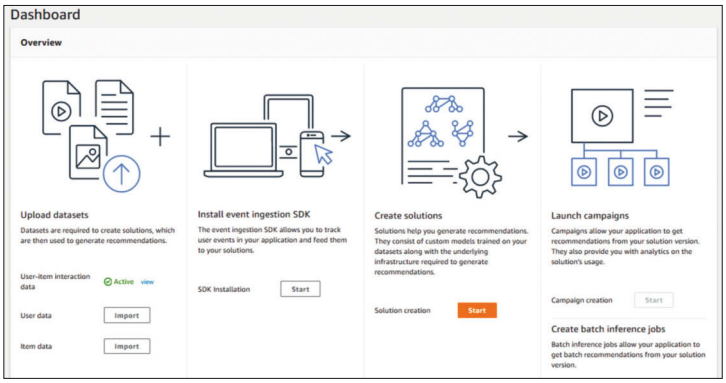


Figure 16

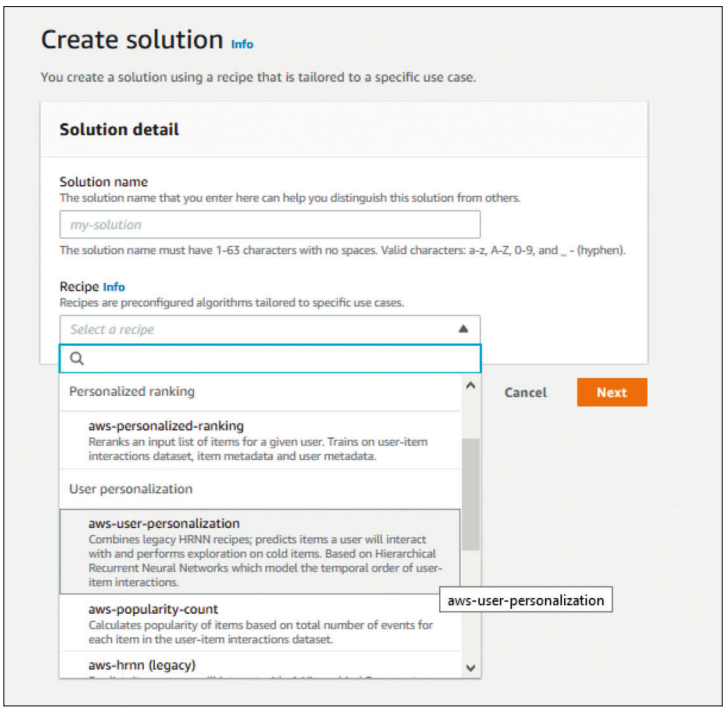


Figure 17

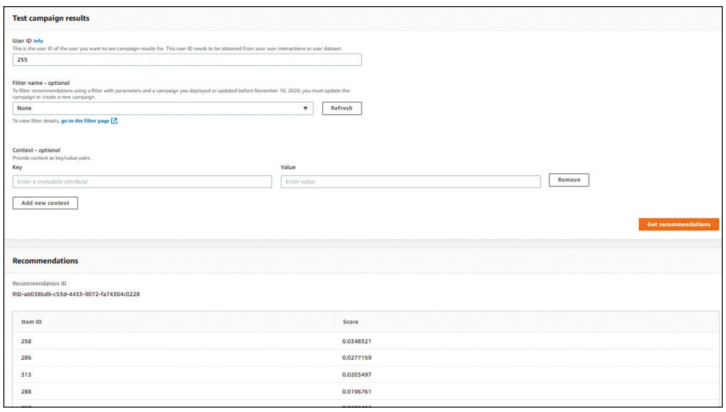


Figure 18

Ressources

https://aws.amazon.com/fr/blogs/aws/amazon-forecast-now-generally-available/
https://github.com/aws-samples/amazon-forecast samples/tree/master/notebooks/basic/Tutorial
https://d1.awsstatic.com/whitepapers/time-series-forecasting-principles-amazon-forecast.pdf
https://docs.aws.amazon.com/forecast/latest/dg/what-is-forecast.html
https://pages.awscloud.com/Introduction-to-Amazon-Forecast-and-Amazon-Personalize_1209-MCL_OD.html
https://github.com/aws-samples/amazon-personalize-samples
https://aws.amazon.com/personalize/
https://aws.amazon.com/personalize/resources/
https://aws.amazon.com/fr/blogs/france/creation-dun-moteur-de-recommandation-avec-amazon-personalize/

Bruno Medeiros de Barros

Solutions Architect au sein des équipes AWS France où il aide les clients français à innover à travers l'adoption des technologies du cloud en assurant la sécurité de leur infrastructure et de leurs données.

Détection des fraudes avec Amazon Fraud Detector en Java

Chaque année, l'équivalent de dizaines de milliards de dollars est perdu dans le monde à cause de la fraude en ligne. Les entreprises qui ont des activités en ligne sont particulièrement exposées aux attaques de mauvais acteurs, qui exploitent souvent des tactiques telles que la création de faux comptes et les paiements avec des cartes de crédit volées.

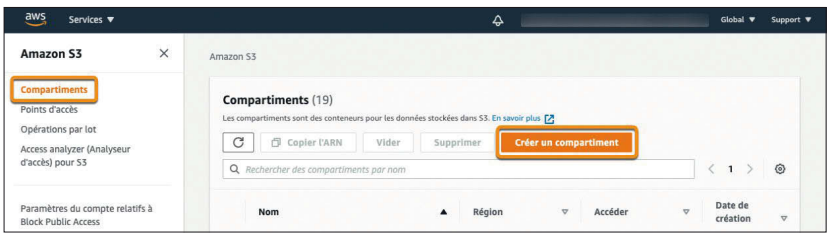
Les entreprises utilisent généralement des applications de détection des fraudes pour identifier les fraudeurs et les arrêter avant qu'ils ne causent de coûteuses perturbations de leurs activités. Cet article explique comment utiliser Amazon Fraud Detector [1] pour mettre en œuvre une solution de détection des fraudes personnalisée pour les activités en ligne, en utilisant des techniques de machine learning pour identifier et implémenter de manière proactive des mesures de protection pour votre entreprise et vos clients. Fraud Detector est un service entièrement géré qui utilise le machine learning (ML) qui bénéficie de plus de 20 ans d'expertise d'Amazon en matière de détection des fraudes, pour identifier les activités potentiellement frauduleuses afin que vous puissiez détecter plus rapidement les fraudes en ligne. Fraud Detector automatise les étapes longues et coûteuses de création, d'entraînement et de déploiement d'un modèle ML pour la détection des fraudes, ce qui vous permet de tirer plus facilement parti de la technologie. Amazon Fraud Detector adapte chaque modèle qu'il crée à votre ensemble de données, afin de produire un modèle le plus précis et pertinent possible. En outre, comme vous ne payez que ce que vous utilisez, vous évitez d'importantes dépenses initiales.

Le fonctionnement du service peut être décrit dans les 5 étapes :

1. Téléchargez votre ensemble de données d'événements historiques sur compartiment dans le cloud à l'aide de S3 [2];
2. Définissez l'événement que vous voulez évaluer pour déterminer s'il y a une fraude et sélectionnez un type de modèle de détection de fraude;
3. Fraud Detector utilise vos données historiques comme données d'entrée pour construire un modèle de machine learning personnalisé. Le service inspecte et enrichit automatiquement les données, effectue l'ingénierie des fonctionnalités, sélectionne les algorithmes, entraîne, règle et héberge votre modèle;
4. Créez des règles pour accepter, réviser ou collecter plus d'informations en fonction des prédictions réalisées par votre modèle;
5. Appelez l'API de Fraud Detector depuis votre application pour recevoir des prédictions de fraude en temps réel et prendre des actions en fonction des règles de détection que vous avez configurées.

Dans les prochaines sections de cet article, nous allons vous guider à travers toutes ces étapes, en vous montrant comment utiliser Frand Detector pour identifier des fraudes lors de l'enregistrement d'un nouveau client dans une application en ligne.

Figure 1



Téléchargement de données historiques

Nous commençons par télécharger un ensemble de données d'entraînement fictives dans un compartiment S3. Vous pouvez télécharger l'ensemble de données qui sera utilisé dans ce tutoriel directement à partir du guide utilisateur de Fraud Detector [3]. Après avoir téléchargé le fichier ZIP contenant les données d'entraînement, décompressez-le dans votre machine locale.

1. Afin de pouvoir exécuter ce tutoriel, vous devez d'abord vous connecter à votre compte AWS. Si vous n'avez pas encore de compte AWS, vous pouvez en créer un gratuitement à partir de la page d'inscription AWS [4].
2. Une fois que vous êtes connecté à votre console AWS, nous allons créer un nouveau compartiment S3 en accédant à la console du service Amazon S3. Vous pouvez y accéder en sélectionnant Amazon S3 dans le menu Services en haut de la page ou en saisissant S3 dans la barre de recherche des Services AWS.
3. Ouvrez la console Amazon S3 en utilisant le menu Services en haut de votre console AWS et cliquez sur le bouton Créer un compartiment. **Figure 1**
4. Dans la section Configuration générale, entrez un nom de compartiment de votre choix. Rappelez-vous que dans AWS, les noms des compartiments doivent être uniques au niveau global. N'hésitez pas à ajouter votre prénom ou toute autre chaîne de caractères mémorisables de votre choix à la fin du nom de votre compartiment au cas où vous recevriez une erreur indiquant que le nom du compartiment existe déjà. Dans mon cas, je vais utiliser le nom de compartiment disponible tutoriel-fraud-detector-programmez.
5. Pour la Région, sélectionnez exactement la même région que celle que nous allons utiliser pour créer et déployer notre modèle de détection des fraudes. Dans notre cas, cette région sera Irlande (eu-west-1). **Figure 2**
6. Laissez tous les autres paramètres de configuration par défaut et cliquez sur le bouton Créer un compartiment au bas de la page.

7. Trouvez votre compartiment récemment créé dans votre liste de compartiments dans la console Amazon S3 et cliquez sur son nom. Vous allez voir une page contenant plusieurs onglets contenant différentes propriétés de votre compartiment S3.
8. Dans l'onglet Objets, cliquez sur le bouton Charger.
9. Dans la section Fichiers et dossiers, cliquez sur Ajouter des fichiers, allez dans le dossier local où vous avez décompressé votre ensemble de données et sélectionnez le fichier registration_data_20K_minimum.csv à charger. Laissez les autres paramètres par défaut et cliquez sur Charger.

Remarque : Pour les besoins de ce tutoriel, nous allons utiliser le fichier CSV contenant un nombre réduit de fonctionnalités afin de simplifier les prochaines étapes et d'optimiser le temps d'entraînement de notre modèle. Notez que dans le fichier ZIP que vous avez téléchargé, vous disposez également d'un ensemble de données plus complet (registration_data_20K_full.csv) qui vous permettra de tester des cas d'utilisation plus complexes dans un deuxième temps.

Définition d'un événement et création du modèle

Maintenant que nous avons créé notre compartiment S3 contenant nos données d'entraînement, nous sommes prêts à passer à la console Amazon Fraud Detector afin de définir un type d'événement qui sera prédit par notre modèle de machine learning, ainsi que pour créer et former le modèle en utilisant notre ensemble de données.

1. Accédez à la console de Fraud Detector en utilisant le menu Services en haut de votre console AWS.
2. Avant de vous lancer sur Fraud Detector, assurez-vous d'être dans la même région que celle que vous avez utilisée pour créer votre compartiment S3. Dans notre cas, Irlande (eu-west-1). Cette condition est obligatoire pour permettre à Fraud Detector d'accéder à nos données d'entraînement.
3. Cliquez sur le bouton Créer un événement. Un événement est essentiellement un ensemble d'attributs concernant un événement particulier. Nous définissons la structure de l'événement que nous voulons évaluer pour détecter la fraude. **Figure 3**
4. Dans la session Détails du type d'événement, dans le champ Nom de l'événement, entrez enregistrement_client.
5. Dans le champ Entité, nous allons sélectionner Créer une nouvelle entité. Cette entité représente la personne, le processus ou toute autre entité qui pourrait être à l'origine de l'événement. **Figure 4**
6. Pour le nom du type d'entité, entrez client et cliquez sur le bouton Créer une entité.
7. Dans la section Variables d'événement, choisissez de Sélectionner des variables à partir d'un ensemble de données d'entraînement.
8. Dans le champ Rôle IAM, sélectionnez Créer un rôle IAM.
9. Dans la fenêtre contextuelle qui apparaît, entrez le nom du compartiment S3 que vous avez créé dans les étapes précédentes. Par exemple, dans mon cas, le nom du compartiment est tutoriel-fraud-detector-programmez.
10. Pour la localisation des données, entrez l'URI S3 du fichier CSV que nous avons téléchargé dans notre compartiment. Vous pouvez obtenir cet URI en retournant dans votre compartiment S3 et en cliquant sur le nom du fichier CSV que nous avons téléchargé. Dans l'onglet Détails, vous trouverez l'URI S3 de votre fichier dans la section Présentation de l'objet.
11. Une fois que vous avez saisi votre URI S3, cliquez sur Charger. Vous verrez les deux variables d'événement (en plus des variables obligatoires EVENT_TIMESTAMP et EVENT_LABEL) qu'Amazon Fraud Detector a pu identifier à partir de notre ensemble de données d'entraînement.
12. Sélectionnez le type de variable approprié pour chacune des variables identifiées. Dans ce cas, mettez en correspondance ip_address avec le type Adresse IP et email_address avec le type Adresse Email. L'association explicite du type de variable aux variables identifiées permet à Amazon Fraud Detector de mieux interpréter vos données. **Figure 5**
13. Pour entraîner un modèle ML à l'aide de cet événement, vous devez définir au moins deux étiquettes. Les étiquettes sont utilisées pour classer les événements comme frauduleux ou légitimes et doivent correspondre aux valeurs utilisées pour identifier les événements frauduleux et légitimes dans le champ EVENT_LABEL de notre ensemble de données d'entraînement. Dans notre cas, nous allons créer deux nouvelles éti-

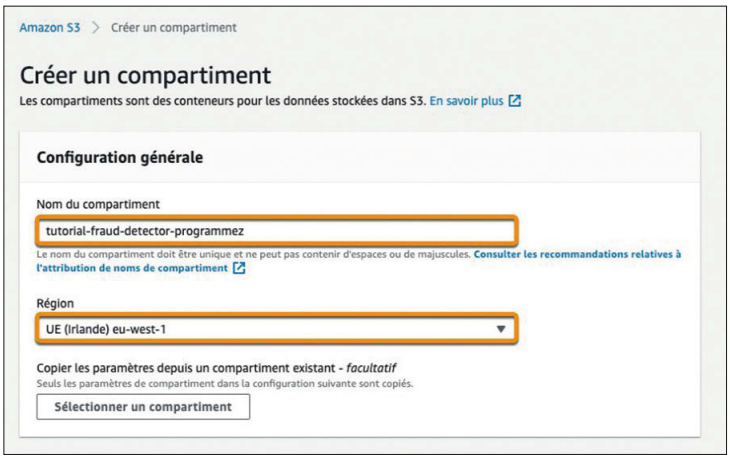


Figure 2

Figure 3

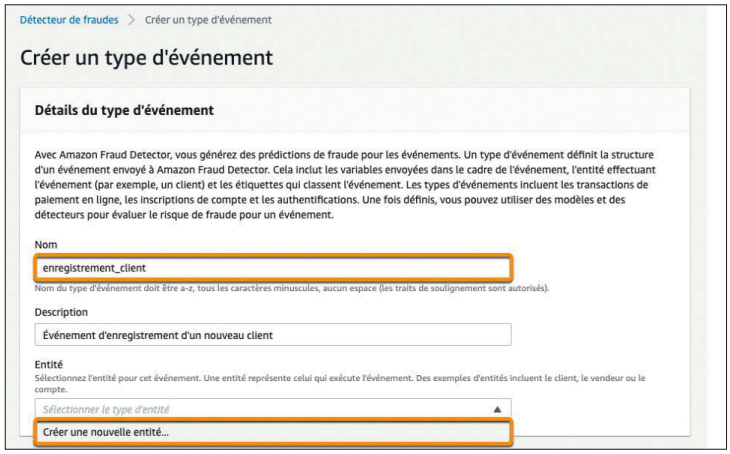


Figure 4

Figure 5

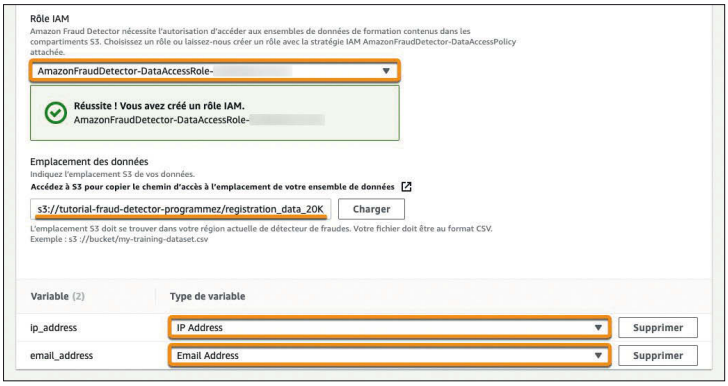


Figure 6

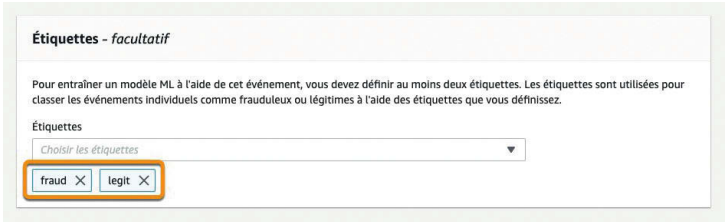


Figure 7

Figure 8

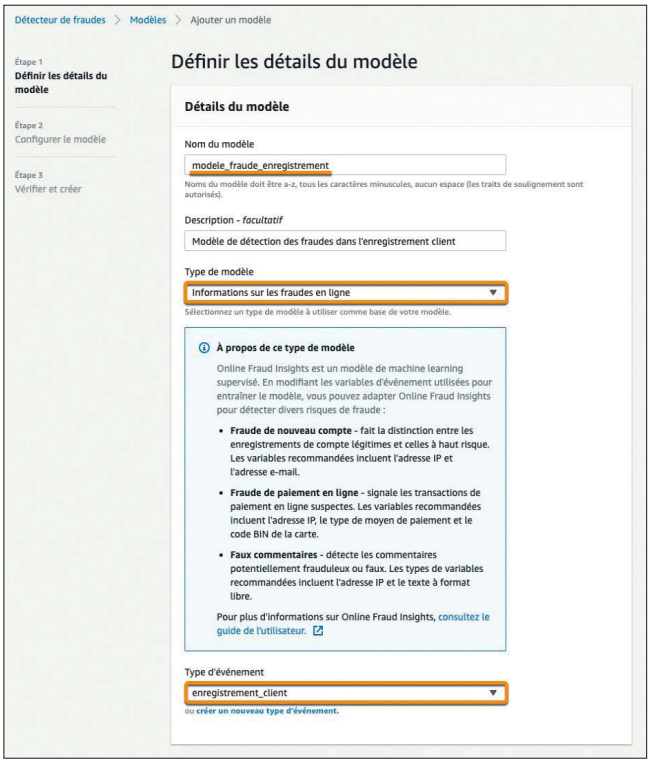
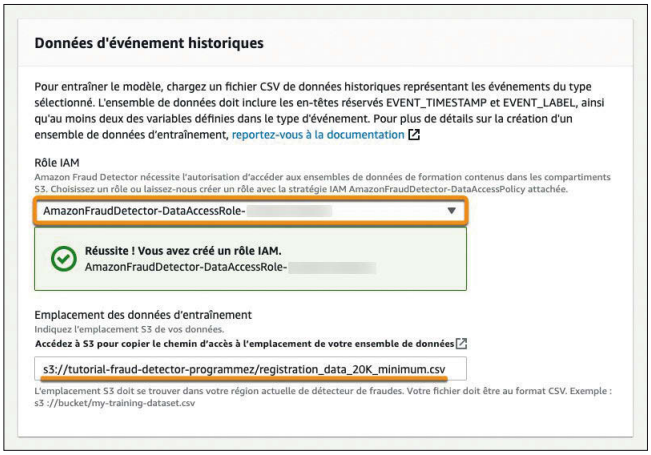


Figure 9



quettes nommées fraud et legit (en anglais). **Figure 6**

14 Enfin, cliquez sur le bouton Créer un type d'événement.

Création d'un modèle

Maintenant que nous avons défini un type d'événement, nous allons créer un modèle de machine learning qui sera capable de prédire un score de fraude pour des événements envoyés en temps réel.

1 Dans la console, cliquez sur Models dans le menu à gauche. Ensuite, à la droite, cliquez sur Ajouter un modèle et sélectionnez Créer un modèle. **Figure 7**

2 Dans l'étape Définir les détails du modèle, entrez modele_fraude_enregistrement pour le Nom du modèle. Vous pouvez également saisir une description de votre modèle si vous le souhaitez.

3 Pour le type de modèle, sélectionnez Informations sur les fraudes en ligne. Ce type de modèle correspond à des modèles de machine learning supervisés capables de détecter le risque de fraude sur diverses activités en ligne.

Remarque : Le type de modèle Informations sur les fraudes en ligne (ou Online Fraud Insights, en anglais) nécessite un ensemble de données d'entraînement qui contient au moins deux variables d'événement en plus des variables obligatoires EVENT_TIMESTAMP et EVENT_LABEL. Le modèle nécessite également 10 000 exemples au total et 400 exemples de fraude pour être entraîné. Pour plus d'informations, veuillez consulter la documentation sur comment préparer les données d'entraînement [5].

4 Pour le champ Type d'événement, sélectionnez le type d'événement que nous avons créé dans les étapes précédentes enregistrement_client. **Figure 8**

5 Dans la section Données historiques des événements, nous allons créer un Rôle IAM qui permet à Fraud Detector d'accéder aux données d'entraînement que nous avons téléchargées auparavant. Pour ce faire, sélectionnez Créer un rôle IAM, donnez le nom du compartiment S3 que vous avez créé dans les étapes précédentes et cliquez sur le bouton Créer un rôle.

6 Pour l'emplacement des données d'entraînement, entrez l'URI S3 pour le fichier CSV que vous avez téléchargé dans votre compartiment. C'est le même URI S3 que nous avons utilisé lors de l'étape de création d'un Type d'événement. **Figure 9**

7 Cliquez sur le bouton Suivant.

8 Dans l'étape Configurer l'entraînement, dans la section Entrées de modèle, vous allez voir la liste de toutes les variables identifiées à partir de notre ensemble de données d'entraînement et qui seront utilisées par Fraud Detector pour entraîner notre modèle de machine learning. Laissez toutes les variables cochées.

9 Dans la section Classification d'étiquette, nous allons indiquer à Fraud Detector comment les étiquettes que nous avons créées précédemment doivent être utilisées pour classer un événement comme frauduleux ou légitime. Utilisez " fraud " comme Étiquette de fraude et " legit " comme Étiquette légitime.

10 Cliquez sur le bouton Suivant. **Figure 10**

11 Dans l'étape Vérifier et créer, vérifiez que vous avez bien configuré votre modèle et cliquez sur le bouton Créer et entraîner un modèle en bas de la page.

12 Vous devriez maintenant voir une page qui indique le statut d'entraînement de votre modèle. Le temps d'entraînement dépend de la taille de l'ensemble de données et de la complexité des tâches d'entraînement, et peut prendre de 30 minutes

jusqu'à quelques heures. Pour l'ensemble de données et le type de modèle sélectionnés pour ce tutoriel, notre processus d'entraînement devrait prendre environ 40 minutes.

13 Une fois que le statut de votre modèle devient Actif, vous pouvez cliquer sur le numéro de version de votre modèle pour consulter toutes les mesures de performance générées par Fraud Detector pour votre modèle.

14 Sur l'onglet Présentation de votre version de modèle, dans la section Performance du modèle, vous pouvez voir deux graphiques qui montrent la distribution des scores et la matrice de confusion du modèle. L'histogramme de Distribution des scores vous permet de voir le pourcentage de fraude totale que le modèle détecte, également connu sous le nom de taux de capture, ainsi que le pourcentage du total des événements légitimes qui sont incorrectement prédits comme fraude. Vous pouvez déplacer le curseur sur l'histogramme pour sélectionner différentes valeurs de score, et les résultats se refléteront également dans la matrice de confusion qui se trouve à droite. **Figure 11**

15 Une fois que vous avez terminé votre analyse, vous pouvez déployer votre modèle en cliquant sur le bouton Actions et en sélectionnant Déployer la version du modèle. Confirmez la version de déploiement dans la fenêtre contextuelle qui s'affiche

Créer un détecteur

Après avoir déployé votre modèle, vous êtes prêt à l'utiliser pour effectuer des prévisions de fraude en temps réel. Pour ce faire, nous devons créer un Détecteur. Dans Fraud Detector, vous créez et configurez des détecteurs pour contenir votre modèle déployé et votre logique de décision (c'est-à-dire les règles). Ces règles seront basées sur le score de risque de fraude prédit par votre modèle et définiront les actions qui seront recommandées à votre application à la suite d'un événement.

1 Pour commencer, dans la console du détecteur de fraude Amazon, cliquez sur Détecteurs dans le menu de gauche, puis cliquez sur le bouton Créer un détecteur sur la page à droite.

2 Dans l'étape Définir les détails du détecteur, pour le Nom du détecteur, entrez detecteur_fraude_enregistrement. Vous pouvez éventuellement saisir une description pour le détecteur.

3 Pour le Type d'événement, sélectionnez le type d'événement que nous avons créé précédemment enregistrement_client.

4 Cliquez ensuite sur le bouton Suivant. **Figure 12**

5 Dans l'étape suivante (Ajouter un modèle), sous l'onglet Modèles de détection des fraudes, cliquez sur le bouton Ajouter un modèle.

6 Dans la fenêtre contextuelle qui s'affiche, sous l'onglet Modèles de détection des fraudes, sélectionnez le modèle que nous avons créé dans les étapes précédentes modele_fraude_enregistrement ainsi que la version actuellement déployée (dans notre cas, 1.0). Cliquez ensuite sur le bouton Ajouter un modèle.

7 De retour à l'étape Ajouter un modèle, cliquez sur le bouton Suivant.

8 Dans l'étape Ajouter des règles, entrez risque_fraude_ eleve comme nom de la règle. Vous pouvez aussi ajouter une description de votre règle si vous le souhaitez.

9 Pour l'expression de la règle, entrez l'expression X. Notez que lorsque vous tapez l'expression dans la console, vous pouvez bénéficier d'une fonction d'auto-complétion qui vous aidera à créer cette règle et toutes les suivantes.

10 Pour le champ Résultat, sélectionnez Créer un nouveau résultat. Pour le nom du résultat, entrez verifier_client et cliquez sur le bouton Enregistrer le résultat. Vous pouvez en option ajouter une description pour le résultat. Ce résultat représente le besoin de vérifier le client en raison d'une fraude potentielle.

11 Une fois que vous avez fini d'ajouter le résultat, cliquez sur le bouton Ajouter une règle. **Figure 13**

12 Avant de passer à l'étape suivante, nous allons créer deux règles supplémentaires. Pour ce faire, cliquez sur le bouton Ajouter une autre règle.

13 Créez deux nouvelles règles en suivant les mêmes étapes décrites précédemment avec les configurations suivantes :

Règle A :

Nom : risque_fraude_faible
Expression : $modele_fraude_enregistrement_insightscore < 500
Résultat (Nom du résultat): approuver_faible_risque

Règle B :

Nom : risque_fraude_moyen
Expression : $modele_fraude_enregistrement_insightscore >= 500 and $modele_fraude_enregistrement_insightscore <= 700
Résultat (Nom du résultat) : examiner_avant_approbation

14 À la fin de la procédure, vous pourrez voir les trois règles créées lors de cette étape de création du détecteur. Examinez la configuration de vos règles et cliquez sur le bouton Suivant.

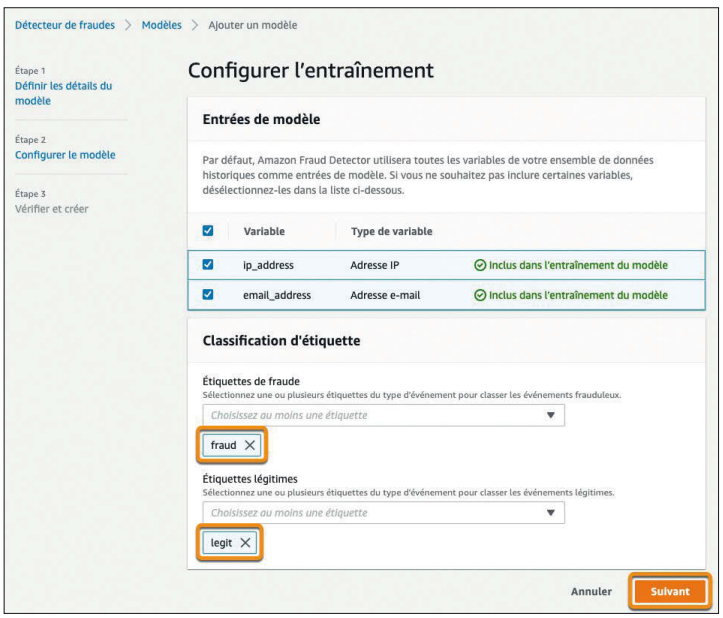


Figure 10

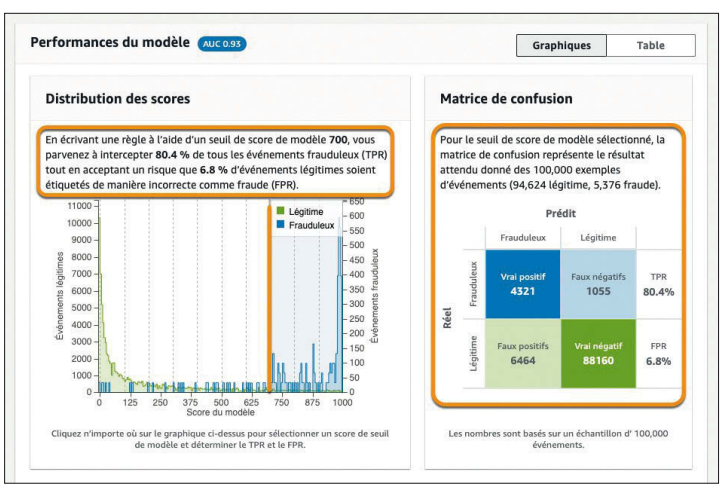


Figure 11

Remarque : N'oubliez pas que ces valeurs seuils de règles ne sont que des exemples. Lorsque vous créez des règles pour votre propre détecteur, vous pouvez utiliser des valeurs qui sont appropriées en fonction de votre modèle, de vos données et de votre activité.

15 Dans l'étape Configurer l'exécution d'une règle, dans la section Mode d'exécution des règles, sélectionnez Première correspondance. Laissez tous les autres paramètres par défaut et cliquez sur le bouton Suivant.

16 Dans la dernière étape (Vérifier et créer), examinez la configuration de votre détecteur et cliquez sur le bouton Créer un détecteur en bas dans la page. Vous verrez une page contenant les détails de la version du détecteur que vous venez de créer.

17 En option, vous pouvez également tester votre version de détecteur dans la section Exécuter le test en envoyant des valeurs de variables à votre modèle et en obtenant un résultat et un score de risque associé. Par exemple, vous pouvez sélectionner une date et une heure dans le passé, entrer la valeur example@fraud.com pour email_address, la valeur 1.2.3.4 pour ip_address, et cliquer sur Exécuter le test. **Figure 14**

18 Après avoir testé votre détecteur, vous pouvez publier sa version actuelle pour qu'elle puisse être utilisée par vos applications en cliquant sur Actions et en sélectionnant Publier en haut de la page. Confirmez que vous souhaitez publier la version du détecteur dans la fenêtre qui s'affichera.

Figure 12

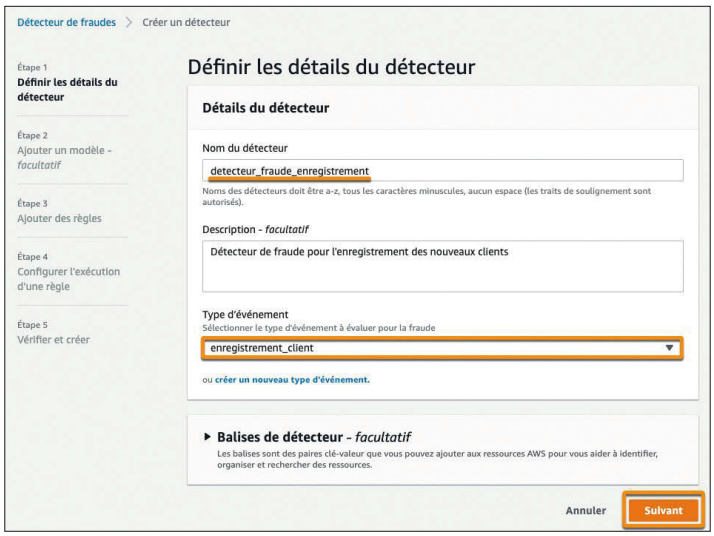
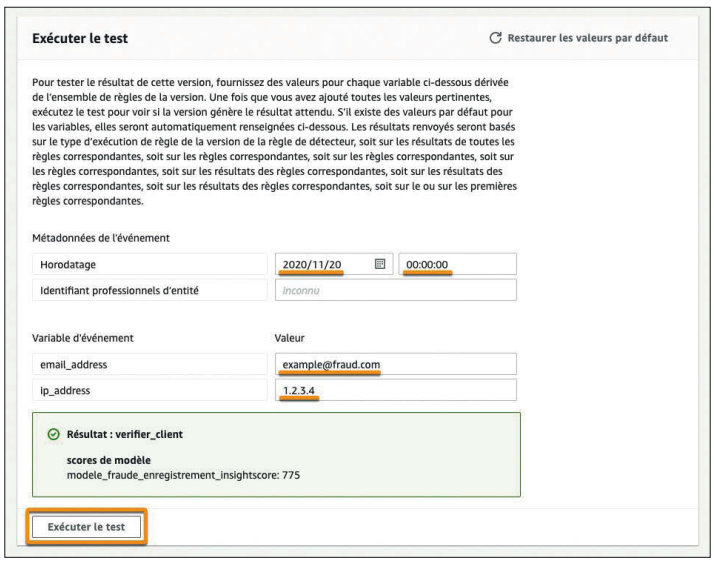


Figure 14



Appelez l'API depuis votre application

Maintenant que votre modèle de détection des fraudes est déployé et que le détecteur qui le contient est publié, vous pouvez obtenir des prédictions en temps réel du risque de fraude pour vos applications grâce à l'API Amazon Fraud Detector. En particulier pour ce tutoriel, nous allons utiliser l'appel API GetEventPrediction [6], qui nous permet d'évaluer un événement à partir d'une version du détecteur en obtenant un score de modèle et le résultat des règles. Pour en savoir plus sur les actions et les types de données pris en charge par l'API Amazon Fraud Detector, consultez la documentation de référence de l'API Amazon Fraud Detector [7].

Pour effectuer des appels API vers Fraud Detector ainsi que vers tout autre service AWS depuis votre application, vous pouvez utiliser le SDK AWS disponible pour votre langage de programmation préféré. Au moment où cet article est rédigé, AWS fournit des SDK pour les langages C++, Go, Java, JavaScript, .NET, Node.js, PHP, Python et Ruby. Pour plus d'informations sur les SDK et autres outils de développement fournis par AWS, consultez la page Outils pour créer sur AWS [8].

Dans l'exemple qui suit, nous utilisons le AWS SDK pour Java 2.0 [9], construit sur Java 8+, pour implémenter une application d'exemple qui va instancier un client Amazon Fraud Detector, définir les valeurs des variables de l'événement, créer une requête de prédiction pour notre détecteur et obtenir les résultats de la prédiction pour l'événement. Dans cet exemple, notre demande va faire partie du paquet com.aws.fddemo. Assurez-vous d'ajuster le nom du paquet en fonction du design de votre application.

```
package com.aws.fddemo;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.frauddetector.FraudDetectorClient;
import software.amazon.awssdk.services.frauddetector.model.Entity;
import software.amazon.awssdk.services.frauddetector.model.GetEventPredictionRequest;
import software.amazon.awssdk.services.frauddetector.model.GetEventPredictionResponse;

import java.util.HashMap;
import java.util.Map;
```

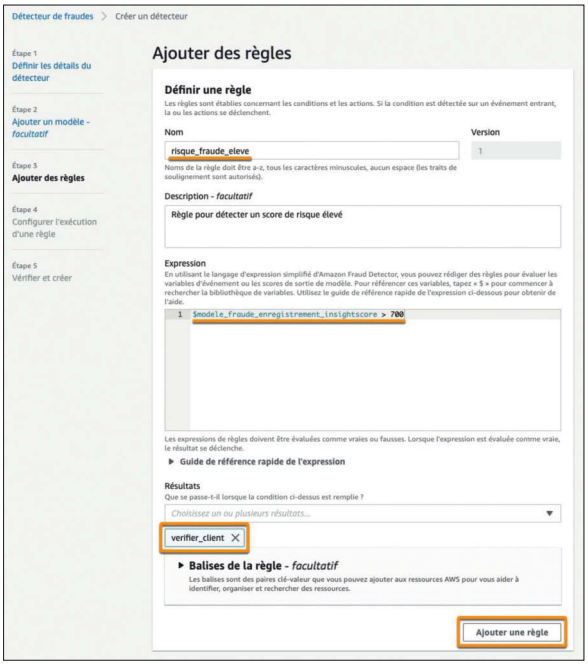


Figure 13

```
public class App {

  public static void main(String[] args) {

    // Obtenir une instance du client Fraud Detector
    FraudDetectorClient fraudDetectorClient = FraudDetectorClient.builder()
        .region(Region.EU_WEST_1)
        .build();

    // Définir les valeurs des variables d'événement
    Map<String,String> eventPredictionVariables = new HashMap<>();
    eventPredictionVariables.put("email_address", "example@fraud.com");
    eventPredictionVariables.put("ip_address", "1.2.3.4");

    // Créer la requête de prédiction
    GetEventPredictionRequest eventPredictionRequest = GetEventPredictionRequest.builder()
        .detectorId("detecteur_fraude_enregistrement")
        .eventId("123456789")
        .eventTypeName("enregistrement_client")
        .eventTimestamp("2020-11-20T00:00:00Z")
        .entities(Entity.builder()
            .entityType("client")
            .entityId("1111")
            .build())
        .eventVariables(eventPredictionVariables)
        .build();

    // Obtenir les résultats de la prédiction
    System.out.println("Getting prediction...");
    GetEventPredictionResponse eventPredictionResponse = fraudDetectorClient.getEventPrediction(event
PredictionRequest);

    System.out.println("\nModel Scores:");
    System.out.println(eventPredictionResponse.modelScores());
    System.out.println("\nRule Results:");
    System.out.println(eventPredictionResponse.ruleResults());

    fraudDetectorClient.close();
  }
}
```

Vous pouvez exécuter cette application sur tout environnement de développement ayant Java 8 (ou une version ultérieure) et Apache Maven installés. L'AWS SDK pour Java fonctionne avec Oracle Java SE Development Kit et avec des distributions de Open Java Development Kit (OpenJDK) telles que Amazon Corretto, Red Hat OpenJDK et AdoptOpenJDK. Pour comprendre comment configurer et construire votre application Java avec Apache Maven, consultez la page de documentation officielle sur comment utiliser AWS SDK for Java avec Apache Maven [9].

Remarque : Pour vous connecter à l'un des services AWS pris en charge à l'aide du AWS SDK pour Java, vous devez fournir des informations d'identification AWS. Pour savoir comment configurer les informations d'identification AWS dans votre environnement de développement ou d'exécution, consultez la documentation officielle de l'AWS SDK pour Java [10].

Bravo, vous avez atteint la fin de ce tutoriel. À ce stade, vous avez pu créer et déployer un modèle de machine learning pour la détection de fraudes à l'aide de Fraud Detector et l'utiliser pour effectuer des prédictions de fraude en temps réel à partir d'une application Java.

Nettoyage

Nous allons supprimer les ressources que vous avez créées au cours de ce tutoriel. Il est conseillé de supprimer les ressources qui ne sont pas utilisées de façon active, afin de réduire les coûts. Des ressources non supprimées peuvent entraîner des frais sur votre compte. La première étape consistera à annuler le déploiement de notre modèle de détection des fraudes.

1. Sur la console Fraud Detector, cliquez sur Détecteurs dans le menu à gauche, puis cliquez sur le nom du détecteur que nous avons créé au cours de ce tutoriel detecteur_fraude_enregistrement.
2. Dans la section Versions, cliquez sur le numéro de la version active du détecteur (dans notre cas, la version numéro 1).
3. Cliquez sur Actions et sélectionnez Supprimer, en haut de la page. Suivez les instructions de la page suivante pour confirmer la suppression de la version du détecteur.
4. Toujours dans la console, cliquez sur Modèles dans le menu à gauche, puis cliquez sur le nom du modèle que nous avons créé dans ce tutoriel modele_fraude_enregistrement.
5. Dans la section Versions de modèle, cliquez sur le numéro de la version active du modèle (dans notre cas, la version 1.0).
6. Cliquez sur Actions et sélectionnez Annuler le déploiement de la version du modèle, en haut de la page. Suivez les instructions de la page suivante pour confirmer l'annulation de la version du modèle.

La dernière étape sera de supprimer l'ensemble des données d'entraînement dans votre compartiment Amazon S3 :

7. Accédez à la console Amazon S3 en utilisant le menu Services en haut de votre console AWS et cliquez sur Compartiments dans le menu à votre gauche. Vous pourrez voir à votre droite la liste de tous les compartiments S3 existant dans votre compte
8. Trouvez et sélectionnez le compartiment S3 que nous avons créé précédemment dans ce tutoriel. Si nécessaire, tapez "tutorial-fraud-detector" dans la barre de recherche en haut de la page.
9. Une fois votre compartiment sélectionné, cliquez sur le bouton Vider en haut de la liste des compartiments et confirmez votre choix en suivant les instructions de la page suivante.
10. Revenez à la liste des compartiments, sélectionnez à nouveau le compartiment S3 que vous venez de vider et cliquez sur le bouton Supprimer en haut de la liste. Confirmez votre choix en suivant les instructions de la page suivante.

Références

[1] https://aws.amazon.com/fr/fraud-detector/
[2] https://aws.amazon.com/fr/s3
[3] https://docs.aws.amazon.com/fr_fr/frauddetector/latest/ug/step-1-get-s3-data.html
[4] https://portal.aws.amazon.com/billing/signup?language=fr_fr
[5] https://docs.aws.amazon.com/fr_fr/frauddetector/latest/ug/online-fraud-insights.html#preparing-training-data
[6] https://docs.aws.amazon.com/fr_fr/frauddetector/latest/api/API_GetEventPrediction.html
[7] https://docs.aws.amazon.com/frauddetector/latest/api/Welcome.html
[8] https://aws.amazon.com/fr/tools/
[9] https://docs.aws.amazon.com/fr_fr/sdk-for-java/latest/developer-guide/setup-project-maven.html
[10] https://docs.aws.amazon.com/fr_fr/sdk-for-java/latest/developer-guide/setup-credentials.html

Vade Secure : comment la détection d'image lutte contre le phishing ?

Les emails frauduleux de type phishing se multiplient et les utilisateurs ont parfois de plus en plus de mal à discerner le vrai du faux. Dans le contexte de l'entreprise, le phishing peut avoir de graves conséquences. Il faut donc être capable de détecter préventivement les emails frauduleux. C'est l'objectif de Vade Secure. *par François Tonic*

La mission de Vade Secure est de protéger ces clients – et en particulier les entreprises - contre les attaques véhiculées par email, et en particulier les attaques de phishing. « 90 % du phishing usurpe l'aspect visuel des marques, incluant les logos. Le but est souvent le même : récupérer des informations confidentielles de l'utilisateur ou de l'entreprise, tels que des mots de passe ou des informations de carte de paiement. » explique Maxime Meyer (Lead Research Scientist).

Vade Secure se focalise donc sur l'analyse des emails et des sites web : « On les reçoit, on extrait les liens et ces derniers sont explorés et analysés. L'analyse du site web – le contenu HTML ainsi que le rendu visuel – nous permet de déterminer s'il s'agit d'un site de phishing, et le cas échéant quelle est la marque usurpée.. » poursuit Maxime.

Cette analyse du site web nécessite des ensembles de données importants. Les services cloud sont d'une aide précieuse. « Pour la détection des logos, nous déployons des modèles de détection d'objets. On entraîne les modèles reposant sur des algorithmes de type deep learning permettant d'aller loin dans l'analyse visuelle du site web. Il nous fallait donc une infrastructure adaptée et des technologies dédiées. Nous n'avions ni le temps ni les compétences pour déployer une telle architecture. Nous nous sommes donc tournés sur le Cloud computing et le choix d'AWS s'est imposé. » explique Maxime.

Vade Secure utilise deux algorithmes spécialisés dans la reconnaissance d'objet : VGG-16, et Resnet-50, dont les paramètres internes ont été pré-entraînés à reconnaître des objets sur la base d'images ImageNet. Comme toujours en deep learning, il faut entraîner encore et encore les algorithmes et les modèles sur des ensembles de données pertinentes pour affiner le modèle et obtenir des résultats pertinents. Cette phase est consommatrice de ressources et de temps machine.

« Typiquement, nous envoyons nos bases d'images sur les services AWS et on spécialise volontairement les modèles sur ces images. Cela permet d'affiner les prédictions en sortie d'analyse. » évoque Maxime.

Le cloud permet de cacher toute la plomberie et de s'occuper uniquement de l'implémentation : paramétrages de services, définition du temps d'apprentissage, les types d'images et d'objets à détecter. Les équipes techniques utilisent les SDK et API d'Amazon Sagemaker. En moyenne, il faut entre 12 et 24h pour entraîner un nouveau modèle. Les modèles sont mis à jour tous les 3 à 6 mois afin de les renforcer avec des images ayant été mal analysées, mais aussi pour mettre à jour les logos des marques ayant changé de chartes graphiques, ajouter de nouvelles marques et logos à supporter, etc. « Heureusement, les marques changent peu souvent » précise Maxime.

Le gros des développements, des algos et des modèles, sont codés en Python, même si, en interne, les logiciels et services sont principalement développés en Go. Côté puissance, des instances chez AWS avec GPU de type ml.p2.8xlarge sont utilisées pour entraîner et faire tourner les modèles.
Les équipes regardent attentivement les performances des instances pour éviter la latence. « Les temps de réponses sont de 1 à 2 secondes, ce qui reste raisonnable pour des analyses poussées, la détection ne se faisant pas sur toutes les images analysées. Si nous voulions améliorer les réponses, il faudrait changer d'instances ou mettre en place des mécanismes de cache » explique Maxime. En revanche, l'éditeur a mis en place un monitoring des instances, notamment avec des sondes logicielles, pour vérifier l'état de l'instance, sa qualité de réponse, etc. La qualité de service est un élément critique dans le domaine de la sécurité.

En cas de panne, Vade Secure peut basculer les services vers d'autres régions AWS. « On peut redéployer en quelques minutes si nécessaire, même si les modèles stockés sur S3 pèsent lourd. Si le problème vient de nos serveurs, nous relançons l'instance fautive. » recadre Maxime.

Si les services AWS (S3, Sagemaker (GroundTruth, Training, Inference)) sont utilisés, beaucoup de services sont développés et déployés sur les serveurs internes. Les SDK et API, notamment S3, permettent de créer les passerelles nécessaires entre les développements internes et les services cloud.

Des exemples de sites de phishing peuvent être obtenus via https://isitphishing.ai/.
Un exemple de détection de logo pour un phishing de la marque Crédit Agricole datant du 13 janvier 2021: **figure 1**
Zoom sur le logo détecté avec la probabilité associée par le modèle

Original. **Figure 2**

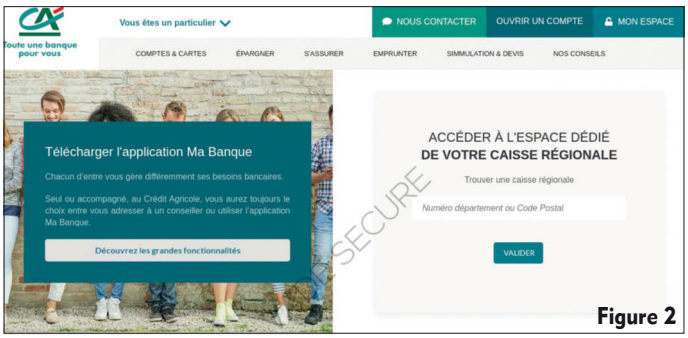


Figure 2

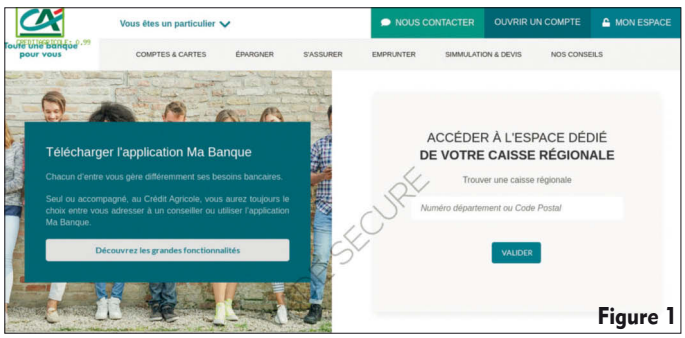


Figure 1

IA générative pour composer de la musique

L'apprentissage automatique (machine learning, ou ML) nécessite beaucoup de mathématiques, d'informatique, de code et d'infrastructure. Ces sujets sont extrêmement importants, mais pour beaucoup d'aspirants développeurs ML, ils ont l'air écrasants, et, parfois, oserais-je dire, ennuyeux.

Julien Simon

En tant qu'évangéliste mondial de l'IA et de l'apprentissage automatique, Julien s'attache à aider les développeurs et les entreprises à donner vie à leurs idées. Il prend souvent la parole lors de conférences, et écrit sur le blog AWS. Avant de rejoindre AWS, Julien a occupé pendant 10 ans des postes de CTO/VP Engineering dans des startups Web de haut niveau.

Pour aider tout le monde à se familiariser avec le ML tout en s'amusant, nous avons introduit plusieurs appareils basés sur le ML. En 2017, nous avons présenté AWS DeepLens, la première caméra au monde compatible avec le Deep Learning, pour aider les développeurs à se familiariser avec le ML et la vision par ordinateur. L'année suivante, nous avons lancé AWS DeepRacer, une voiture de course entièrement autonome à l'échelle 1/18e mettant en œuvre par l'apprentissage du renforcement. En 2019, nous avons lancé AWS DeepComposer, un clavier de 32 touches à 2 octaves conçu pour permettre aux développeurs de mettre la main sur l'IA générative, avec des modèles pré-entraînés, ou les vôtres. Voici la vue de haut niveau :

- Connectez-vous à la console AWS DeepComposer ;
- Enregistrez une courte mélodie musicale, ou utilisez une chanson préenregistrée ;
- Sélectionnez un modèle génératif pour votre genre préféré, pré-entrainé ou le vôtre ;
- Utilisez ce modèle pour générer une nouvelle composition polyphonique ;
- Jouez la composition dans la console ;
- Exportez la composition.

Laissez-moi vous montrer comment générer rapidement votre première composition avec un modèle pré-entraîné. Ensuite, je vous montrerai comment entraîner votre propre modèle. Puis, nous parlerons de la technologie sous-jacente à DeepComposer : les Generative Adversarial Networks (GAN) et les Transformers.

Utilisation d'un modèle entraîné

En ouvrant la console, je vais au studio de musique, où je peux soit sélectionner un morceau préenregistré, soit enregistrer un morceau moi-même. Je sélectionne l'« Ode à la joie » de Beethoven et la technique « Generative adversarial network ». Je sélectionne également le modèle pré-entraîné que je souhaite utiliser : classique, jazz, rock ou pop. Ces modèles ont été formés sur de grands ensembles de données musicales pour leurs genres respectifs, et je peux les utiliser directement. Je choisis 'Rock' et génère la composition. **Figure 1**

Quelques secondes plus tard, je vois les accompagnements supplémentaires générés par le modèle : batterie, guitare électrique, basse électrique et percussions. Faites-en de même et écoutez le morceau ! **Figure 2**

Essayons autre chose. Cette fois, je sélectionne « Piano Sonata No. 30 » et la technique « Transformers ». Celle-ci va générer 20 secondes de musique supplémentaire. **Figure 3**

Je peux aussi modifier les différents paramètres. Le résultat est parfois surprenant ! Pour finir, je peux exporter mes compositions vers un fichier MIDI, et les envoyer à une maison de disques. La gloire m'attend !

Entraînement de votre propre modèle

Je peux aussi former mon propre modèle sur un ensemble de données pour mon genre préféré. J'ai besoin de sélectionner :

- Paramètres d'architecture pour le générateur et le discriminateur (nous expliquerons ces termes dans la section suivante) ;
- La fonction d'erreur utilisée pendant l'entraînement pour mesurer la différence entre la sortie de l'algorithme et la valeur attendue ;
- Les hyper-paramètres de l'algorithme.

Tout d'abord, sélectionnons une architecture GAN :

- MuseGan, de Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang et Yi-Hsuan : MuseGan a été spécialement conçu pour générer de la musique. Le générateur de MuseGan est composé d'un réseau partagé pour apprendre une représentation de haut

Figure 1

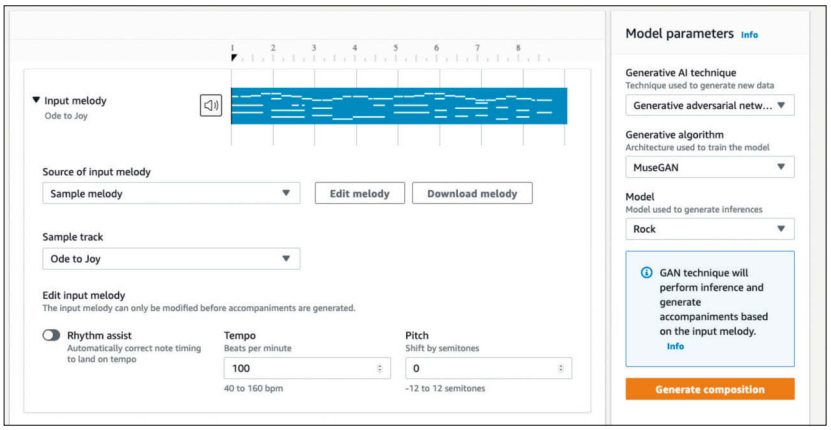
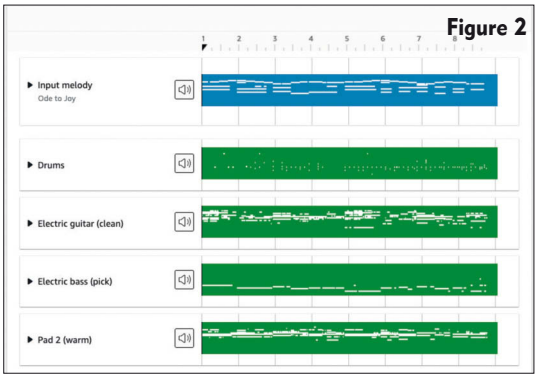


Figure 2



niveau de la chanson, et d'une série de réseaux privés pour apprendre à générer des pistes musicales individuelles.

- U-Net, d'Olaf Ronneberger, Philipp Fischer et Thomas : U-Net a connu un grand succès dans le domaine de la traduction d'images (par exemple, la conversion d'images hivernales en images estivales), et il peut également être utilisé pour la génération de musique. C'est une architecture plus simple que MuseGan, et donc plus facile à comprendre pour les débutants. **Figure 4**

Choisissons MuseGan, et donnons un nom au nouveau modèle.

Ensuite, je dois juste choisir le jeu de données sur lequel je veux former mon modèle.

En option, je peux également définir des « hyperparamètres » (c'est-à-dire des paramètres d'entraînement), mais je conserve les paramètres par défaut. Enfin, je lance l'entraînement. AWS DeepComposer se charge de toute l'infrastructure et de la configuration de l'apprentissage.

Environ 8 heures plus tard, le modèle a été entraîné, et je peux l'utiliser pour générer des compositions comme précédemment.

Figure 3

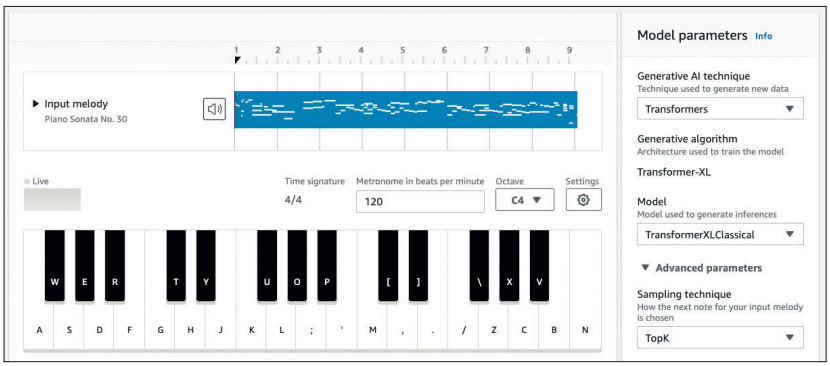


Figure 4

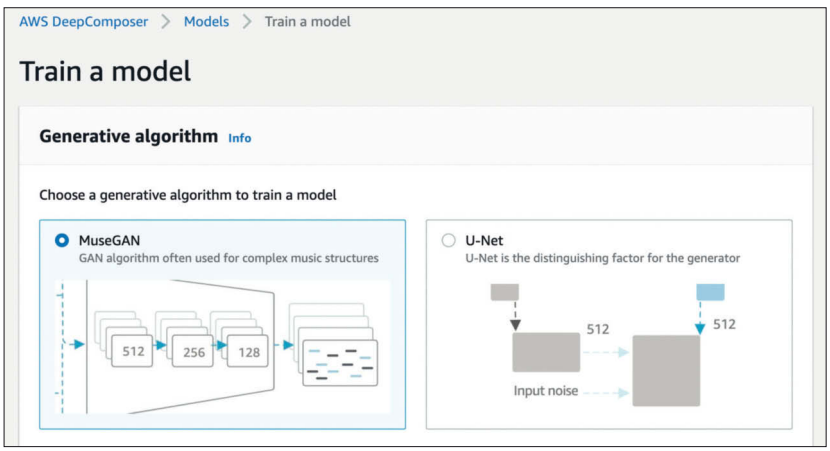
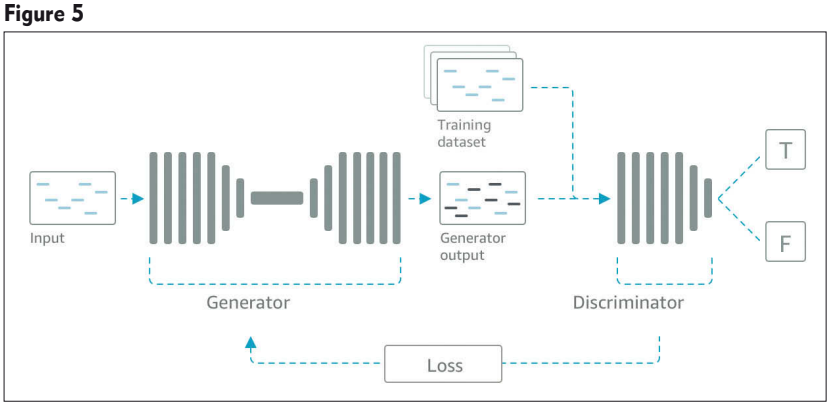


Figure 5



Capsules d'apprentissage

DeepComposer est basé sur des architectures de réseau neuronal conçues spécifiquement pour générer de nouveaux échantillons à partir d'un ensemble de données existant. Jusqu'à présent, les développeurs n'avaient pas de moyen facile de commencer à les utiliser. Afin de les aider, quel que soit leur niveau en ML, nous avons construit une collection de capsules qui présentent leurs concepts clés.

Introduction aux « Generative Adversarial Networks »

Les GANS ont vu le jour en 2014, avec la publication de « Generative Adversarial Networks » par Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville et Yoshua Bengio. **Figure 5**

Pour paraphraser les auteurs, le modèle Generator est opposé à un adversaire : un modèle Discriminator qui apprend à déterminer si un échantillon qui lui arrive est authentique ou pas. Le modèle générateur peut être considéré comme un faussaire, essayant de fabriquer des données qui ne peuvent pas être différenciées des vraies données. Le modèle Discriminator essaye précisément de les différencier. La compétition pousse les deux modèles à améliorer leurs méthodes jusqu'à ce que les données générées ne soient pas distinguables des données authentiques.

Permettez-moi de m'étendre un peu à ce sujet :

- Le Generator n'a pas accès à l'ensemble de données. En utilisant des données aléatoires, il crée des échantillons qui sont transférés au modèle Discriminator ;
- Le Generator est un modèle de classification binaire, apprenant à reconnaître des échantillons de données authentiques (inclus dans le jeu de données) à partir de faux échantillons (constitués par le générateur) ;
- Durant son apprentissage, le Discriminator met ses paramètres à jour ;
- Les mêmes mises à jour sont appliquées au Generator. C'est la clé pour comprendre les GAN : en appliquant ces mises à jour, le Generator apprend progressivement à générer des échantillons plus proches de ceux que le Generator considère comme authentiques.

Pour résumer, vous devez commencer par vous positionner en tant qu'expert de la contrefaçon pour devenir un grand contrefacteur... Mais ne prenez pas cela comme un conseil de carrière ! Si vous êtes curieux d'en savoir plus, vous pouvez aimer ce post de mon blog personnel qui explique comment générer des échantillons MNIST avec un GAN Apache MXNet : https://julsimon.medium.com/generative-adversarial-networks-on-apache-mxnet-part-1-b6d39e6b5df1

Si vous voulez juste jouer de la musique et vous amuser comme ce petit gars, c'est bien aussi ! **Figure 6**

INTRODUCTION AUX TRANSFORMERS

Le Transformer est un modèle récent d'apprentissage profond destiné à être utilisé avec des données séquentielles telles que le texte, les séries chronologiques, la musique et les génomes. Alors que les modèles de séquences plus anciens tels

que les réseaux neuronaux récurrents (RNN) ou les réseaux de mémoire à long terme (LSTM) traitent les données de façon séquentielle, les Transformers traitent les données en parallèle. Cela leur permet de traiter des quantités massives de données d'entraînement en utilisant de puissantes ressources de calcul basées sur GPU.

De plus, les RNN et les LSTM traditionnels peuvent avoir de la difficulté à modéliser les dépendances à long terme d'une séquence parce qu'ils peuvent oublier les parties antérieures de la séquence. Les Transformers utilisent un mécanisme d'attention pour surmonter ce manque de mémoire, en forçant chaque étape de la séquence de sortie à prêter attention aux parties pertinentes de la séquence d'entrée. Par exemple, lorsqu'on demande à un modèle d'IA conversationnelle basé sur un Transformer « Comment est le temps maintenant ? » et le modèle répond « Il fait chaud et ensoleillé aujourd'hui », le mécanisme d'attention guide le modèle à se concentrer sur le mot « temps » lorsqu'il répond par « chaud » et « ensoleillé », et à se concentrer sur « maintenant » en répondant par « aujourd'hui ». Ceci est différent des RNN et des LSTM traditionnels, qui traitent en oubliant le contexte de chaque mot à mesure que la distance entre les mots augmente.

Entraînement d'un modèle Transformer pour générer de la musique

Pour travailler avec des jeux de données musicaux, la première étape consiste à convertir les données en une séquence de jetons. Chaque jeton représente un événement musical distinct dans la partition. Un jeton peut représenter l'horodatage d'une note qui est frappée, ou sa hauteur. La relation entre ces jetons et les notes musicales est similaire à la relation entre les mots d'une phrase ou d'un paragraphe. Les jetons dans la musique peuvent représenter des notes ou d'autres caractéristiques musicales, tout comme la façon dont les jetons dans la langue peuvent représenter des mots ou de la ponctuation. Cela diffère des modèles précédents pris en charge par AWS DeepComposer tels que GAN et AR-CNN, qui traitent la génération de musique comme un problème de génération d'images.

Ces séquences de jetons sont ensuite utilisées pour entraîner le modèle Transformer. Pendant l'entraînement, le modèle tente d'apprendre une distribution statistique qui correspond à la distribution sous-jacente du jeu de données d'entraînement. Au cours de l'inférence, le modèle génère une séquence de jetons par échantillonnage à partir de la distribution apprise pendant l'entraînement. La nouvelle partition musicale est créée en transformant la séquence de jetons en musique. Music Transformer et MuseNet sont des exemples d'autres algorithmes qui utilisent l'architecture Transformer pour la génération de musique.

Dans AWS DeepComposer, nous utilisons l'architecture TransformerXL pour générer de la musique. Elle est capable de capturer des dépendances à long terme qui sont 4,5 fois plus longues qu'un transformateur traditionnel. En outre, il a été démontré qu'il était 18 fois plus rapide qu'un transformateur traditionnel lors de l'inférence. Cela signifie qu'AWS DeepComposer peut vous fournir des compositions musicales de meilleure qualité à faible latence lors de la génération de nouvelles compositions.

Figure 6

Configuration des paramètres avancés pour Transformers

Vous pouvez choisir parmi sept paramètres qui peuvent être utilisés pour modifier la façon dont votre mélodie est créée :

- Techniques d'échantillonnage ;
- Seuil d'échantillonnage ;
- Risque créatif ;
- Durée d'entrée ;
- Durée de l'extension de piste ;
- Temps de repos maximal ;
- Longueur maximale de la note.

Vous avez trois techniques d'échantillonnage : TopK, Nucleus et Random. Vous pouvez également définir la valeur de seuil d'échantillonnage pour la technique choisie. Nous parlons d'abord de chaque technique.

Échantillonnage TopK

Lorsque vous choisissez la technique d'échantillonnage TopK, le modèle choisit les K jetons qui ont la plus grande probabilité de se produire. Pour définir la valeur de K, modifiez le seuil d'échantillonnage.

Si votre seuil d'échantillonnage est élevé, le nombre de jetons disponibles (K) est grand. Un grand nombre de jetons disponibles signifie que le modèle peut choisir parmi une plus grande variété de jetons musicaux. Dans votre mélodie étendue, cela signifie que les notes générées sont susceptibles d'être plus diverses, mais cela se fait au bénéfice d'une musique potentiellement moins cohérente.

D'un autre côté, si vous choisissez une valeur de seuil trop faible, le modèle se limite à choisir parmi un ensemble plus petit de jetons (qui, selon le modèle, a une plus grande probabilité d'être correct). Dans votre mélodie étendue, vous remarquerez peut-être moins de diversité musicale et des résultats plus répétitifs.

Échantillonnage Nucleus

À un niveau élevé, l'échantillonnage Nucleus est très similaire à TopK. L'établissement d'un seuil d'échantillonnage plus élevé permet une plus grande diversité au détriment de la cohérence. Il y a une différence subtile entre les deux approches. L'échantillonnage Nucleus choisit les jetons de probabilité supérieure qui correspondent à la valeur définie pour le seuil d'échantillonnage. Nous faisons cela en triant les probabilités du plus grand au plus petit, et en calculant une somme cumulative pour chaque jeton.

Par exemple, nous pouvons avoir six jetons musicaux avec les probabilités {0.3, 0.3, 0.2, 0.1, 0.05, 0.05}. Si nous choisis-

sons TopK avec un seuil d'échantillonnage égal à 0,5, nous choisissons trois jetons (six jetons musicaux totaux * 0,5). Ensuite, nous échantillonnons entre les jetons avec des probabilités égales à 0,3, 0,3 et 0,2. Si nous choisissons l'échantillonnage Nucleus avec un seuil d'échantillonnage de 0,5, nous n'échantillonnons que deux jetons {0,3, 0,3}, car la probabilité cumulative (0,6) dépasse le seuil (0,5).

L'échantillonnage aléatoire

L'échantillonnage aléatoire est la technique d'échantillonnage la plus élémentaire. Avec l'échantillonnage aléatoire, le modèle est libre d'échantillonner entre tous les jetons disponibles et est échantillonné « aléatoirement » à partir de la distribution de sortie. Le résultat de cette technique d'échantillonnage est identique à celui de l'échantillonnage TopK ou Nucleus lorsque le seuil d'échantillonnage est fixé à 1. Les notes sont très diverses, mais dans leur ensemble, les notes perdent leur cohérence et sonnent parfois d'une façon un peu aléatoire.

Risque créatif

Le risque créatif est un paramètre utilisé pour contrôler le caractère aléatoire des prédictions. Un faible risque créatif rend le modèle plus confiant, mais aussi plus conservateur dans ses échantillons (il est moins susceptible d'échantillonner des jetons candidats improbables). D'autre part, un risque créatif élevé produit une distribution de probabilité plus plate sur la liste des jetons musicaux, de sorte que le modèle prend plus de risques dans ses échantillons (il est plus susceptible de prélever des jetons candidats improbables), ce qui entraîne une plus grande diversité et probablement plus d'erreurs. Les erreurs peuvent inclure la création de notes plus longues ou plus courtes, des périodes de repos plus longues ou plus courtes dans la mélodie générée, ou l'ajout de notes erronées à la mélodie générée.

Durée d'entrée

Ce paramètre indique au modèle quelle partie de la mélodie d'entrée utiliser pendant l'inférence. La partie utilisée est définie comme le nombre de secondes sélectionnées comptant vers l'arrière à partir de la fin de la piste en entrée. Lors de l'extension de la mélodie, le modèle conditionne la sortie qu'il génère en fonction de la portion de la mélodie d'entrée que vous fournissez. Par exemple, si vous choisissez 5 secondes comme durée d'entrée, le modèle utilise uniquement les 5 dernières secondes de la mélodie d'entrée pour le conditionnement et ignore la partie restante lors de l'inférence. Les clips audio suivants ont été générés en utilisant des durées d'entrée différentes.

Durée de l'extension de piste

Lors de l'extension de la mélodie, le Transformer génère en continu des jetons jusqu'à ce que la partie générée atteigne la durée d'extension de piste sélectionnée. La raison pour laquelle le modèle génère parfois moins que la valeur sélectionnée est parce que le modèle génère des valeurs en termes de jetons et non de temps. Les jetons, cependant, peuvent représenter différentes longueurs du temps. Par exemple, un jeton peut représenter une durée de note de 0,1 seconde ou 1 seconde selon ce que le modèle pense approprier.

Temps de repos maximal

Lors de l'inférence, le modèle Transformer peut créer des artefacts musicaux. La modification de la valeur du temps de repos maximal limite les périodes de silence, en secondes, que le modèle peut générer.

Longueur maximale de la note

La modification de la valeur de la longueur maximale de la note limite la durée pendant laquelle une seule note peut être conservée lors de l'inférence.

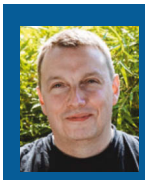
Comme vous le voyez, DeepComposer vous permet de commencer facilement à composer de la musique, puis d'affiner sa génération en jouant sur de nombreux paramètres. Voici quelques ressources pour en apprendre plus.

A vous de jouer !

Ressources

https://aws.amazon.com/deepcomposer/
https://aws.amazon.com/blogs/machine-learning/category/artificial-intelligence/aws-deepcomposer/

Entraînez et déployez facilement vos modèles de machine learning avec Amazon SageMaker



Julien Simon

En tant qu'évangéliste mondial de l'IA et de l'apprentissage automatique, Julien s'attache à aider les développeurs et les entreprises à donner vie à leurs idées. Il prend souvent la parole lors de conférences, et écrit sur le blog AWS. Avant de rejoindre AWS, Julien a occupé pendant 10 ans des postes de CTO/VP Engineering dans des startups web de haut niveau.

Amazon SageMaker est un service managé de Machine Learning (ML). Avec SageMaker, les spécialistes des données et les développeurs peuvent rapidement et facilement développer et entraîner des modèles de machine learning, puis les déployer directement dans un environnement hébergé prêt pour la production.

SageMaker contient SageMaker Studio. Il s'agit d'un IDE web et dédié au ML. Il en facilite la création, l'entraînement, le débogage, le déploiement et la surveillance de vos modèles de Machine Learning. Studio fournit tous les outils dont vous avez besoin pour faire passer vos modèles de l'expérimentation à la production tout en stimulant votre productivité.
Dans une interface visuelle unifiée unique, les clients peuvent effectuer les tâches suivantes :

- Écrire et exécuter du code dans les blocs-notes Jupyter ;
- Générer et entraîner des modèles ML ;
- Déployer les modèles et surveiller les performances de leurs prédictions ;
- Suivre et déboguer les expériences de ML.

Dans cet article, nous allons entraîner un modèle qui prédit le prix d'une maison. Pour cela, nous utiliserons le jeu de données « Boston Housing » https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html, et l'algorithme XGBoost contenu dans SageMaker, en mode régression linéaire.

Utiliser une infrastructure entièrement gérée

Tous les travaux SageMaker s'exécutent sur une infrastructure managée. Jetons un coup d'œil sous le capot, et voyons ce qui se passe ; nous entraînons et déployons des modèles.
Tous les algorithmes SageMaker doivent être empaquetés dans des conteneurs Docker. Comme vous vous y attendez, les algorithmes intégrés sont préemballés et les conteneurs sont facilement disponibles pour la formation et le déploiement. Ils sont hébergés dans Amazon Elastic Container Registry (ECR), le service de registre Docker d'AWS https://aws.amazon.com/ecr/
Étant donné que ECR est un service basé sur une région, vous trouverez un ensemble de conteneurs dans chaque région où SageMaker est disponible. Vous pouvez trouver la liste des conteneurs d'algorithmes intégrés à https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-algo-docker-registry-paths.html . Par exemple, le nom du conteneur pour l'algorithme LinearLearner dans la région eu-west-1 est 438346466558.dkr.ecr.eu-west-1.amazonaws.com/linear-learner:latest. Ces conteneurs ne peuvent être utilisés que sur des instances gérées SageMaker, de sorte que vous ne pourrez pas les exécuter sur votre ordinateur local.
Lorsque vous lancez une tâche d'entraînement, SageMaker démarre l'infrastructure en fonction de vos besoins (type d'instance et nombre d'instances). Une fois qu'une instance d'entraînement est en service, elle récupère le conteneur d'entraînement approprié de l'ECR. Les hyperparamètres sont appliqués à l'algorithme, qui reçoit également l'emplacement de votre jeu de données. Par défaut, l'algorithme copie ensuite le jeu de données complet à partir de S3 et commence l'entraînement.

Une fois l'entraînement terminé, le modèle est emballé dans un artefact de modèle enregistré dans S3. Ensuite, l'infrastructure est arrêtée automatiquement. Les journaux sont disponibles dans Amazon CloudWatch Logs. Enfin, vous n'êtes facturé que pour la durée exacte de l'entraînement.

Lorsque vous lancez une tâche de déploiement, SageMaker crée une nouvelle infrastructure en fonction de vos besoins. Une fois qu'une instance de point de terminaison est en service, elle récupère le conteneur de prédiction approprié et charge votre modèle à partir de S3. Ensuite, le point de terminaison HTTPS est provisionné et est prêt pour la prédiction en quelques minutes.

Si vous avez configuré le point de terminaison avec plusieurs instances, l'équilibrage de charge et la haute disponibilité sont configurés automatiquement. Si vous avez configuré Auto Scaling, il est également appliqué.

Comme vous vous y attendez, un point de terminaison reste en place jusqu'à ce qu'il soit supprimé explicitement, soit dans la console, soit avec un appel d'API SageMaker. En attendant, vous serez facturé pour le point de terminaison, alors assurez-vous de supprimer les points de terminaison dont vous n'avez pas besoin.

Maintenant que nous comprenons la vue d'ensemble, commençons à examiner le SDK SageMaker et comment nous pouvons l'utiliser pour former des modèles et déployer des modèles.

Comprendre le SDK SageMaker

Examinons un flux de travail SageMaker typique. Vous le verrez encore et encore dans nos exemples (https://github.com/awslabs/amazon-sagemaker-examples/) :

1. Rendre votre jeu de données disponible dans S3 : dans la plupart des exemples, nous téléchargeons un jeu de données à partir d'Internet ou chargerons une copie locale. Cependant, dans la vie réelle, votre jeu de données brutes serait probablement déjà dans S3, et vous le préparerez en utilisant un service de préparation des données. Dans tous les cas, le jeu de données doit être dans un format que l'algorithme comprend, tel que CSV, protobuf, etc.
2. Configurer la tâche d'entraînement : c'est là que vous sélectionnez l'algorithme, définissez des hyperparamètres ainsi que les exigences en matière d'infrastructure.
3. Lancez le travail d'entraînement : c'est là que nous lui transmettons l'emplacement de votre jeu de données dans S3. L'entraînement est réalisé sur l'infrastructure, créée et provisionnée automatiquement en fonction de vos besoins. Une fois l'entraînement terminé, l'artefact du modèle est enregistré dans S3. L'infrastructure est interrompue automatiquement, et vous ne payez que pour ce que vous avez réellement utilisé.
4. Déployer le modèle : vous pouvez déployer un modèle soit sur un point de terminaison HTTPS pour la prédiction en temps réel, soit pour la transformation par lots. Encore une fois, il vous suffit de définir les exigences en matière d'infrastructure.
5. Prédire les données : appel d'un point de terminaison en temps réel ou d'un transformateur de lots. Comme vous vous y attendez, l'infrastructure est gérée ici aussi. Pour la production, vous devez également surveiller la qualité des données et des prévisions.
6. Nettoyez ! : Cela implique de réduire le point de terminaison, afin d'éviter des frais inutiles.

Comprendre ce flux de travail est essentiel pour être productif avec SageMaker. Heureusement, le SDK SageMaker possède des API simples qui correspondent à ces étapes.

Avant toute chose, nous devons créer un utilisateur nous permettant de nous connecter à SageMaker Studio. Pour cela, il suffit de suivre la procédure suivante. Elle ne prend que quelques minutes.

Créer un utilisateur pour SageMaker Studio

1. Ouvrez la console SageMaker ;
2. Choisissez SageMaker Studio en haut à gauche de la page ;
3. Sur la page SageMaker Studio sous « Mise en route », choisissez « Démarrage rapide » ;
4. Pour « Nom d'utilisateur », conservez le nom par défaut ou créez un nouveau nom. La longueur maximale du nom est de 63 caractères. Caractères valides : - A à Z, a à z, 0 à 9, et - (trait d'union) ;
5. Pour « Rôle d'exécution », choisissez « Créer un rôle », la boîte de dialogue « Créer un rôle IAM » s'ouvre :
   a - Pour « Compartiments S3 que vous spécifiez », spécifiez des compartiments S3 supplémentaires auxquels les utilisateurs de vos blocs-notes peuvent accéder.
   b - Si vous ne souhaitez pas ajouter d'accès à d'autres compartiments, choisissez « Aucun ».
6. Choisissez « Créer un rôle ». SageMaker crée un rôle IAM nommé « AmazonSageMaker-ExecutionPolicy » ;
7. Choisissez « Soumettre » ;
8. Dans le panneau de configuration, attendez que « Statut » passe à « Ready » (Prêt) ;
9. Choisissez « Ouvrir Studio ». La page de chargement Amazon SageMaker Studio s'affiche.

Lorsque Studio s'ouvre, nous pouvons commencer à l'utiliser, en créant un nouveau carnet Jupyter dans le menu « File / New / Notebook ».

Installation du SDK SageMaker

Une fois notre carnet démarré, nous pouvons installer la dernière version du SDK SageMaker comme suit :

```
%%sh
pip install -q sagemaker --upgrade
```

Puis, il suffit de l'importer :

```
import sagemaker
print(sagemaker.__version__)
2.19.0
```

Chargement et préparation du jeu de données

Téléchargeons le jeu de données à partir d'un de mes dépôts GitHub :

```
%%sh
wget https://raw.githubusercontent.com/PacktPublishing/Learn-Amazon-SageMaker/master/sdkv2/ch4/housing.csv
```

En utilisant pandas, nous chargeons le jeu de données CSV :

```
import pandas as pd
dataset = pd.read_csv('housing.csv')
```

Ensuite, nous imprimons la forme du jeu de données :

```
print(dataset.shape)
dataset[:5]
```

Il contient 506 lignes et 13 colonnes : (506, 13)

Maintenant, nous affichons les 5 premières lignes du jeu de données. Cela imprime le tableau visible dans l'image suivante. Pour chaque maison, nous voyons 12 caractéristiques, et un attribut cible (medv) défini sur la valeur médiane de la maison en milliers de dollars. **Figure 1**

En lisant la documentation de l'algorithme https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html, nous voyons que SageMaker exige qu'un fichier CSV ne comporte pas d'entête et que la variable cible se trouve dans la première colonne. En conséquence, nous déplaçons la colonne medv vers l'avant du dataframe.

[4]:

	crim	zn	indus	chas	nox	age	rm	dis	rad	tax	ptratio	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	5.33	36.2

Figure 1

```
dataset = pd.concat([dataset['medv'], dataset.drop(['medv'], axis=1)], axis=1)
```

Un peu de magie scikit-learn aide à diviser les données en deux parties : 90 % pour la formation, 10 % pour la validation.

```
from sklearn.model_selection import train_test_split
training_dataset, validation_dataset = train_test_split(dataset, test_size=0.1)
print(training_dataset.shape)
print(validation_dataset.shape)

(455, 13)
(51, 13)
```

Nous enregistrons ces deux jeux de données dans des fichiers CSV individuels, sans index ni en-tête.

```
training_dataset.to_csv('training_dataset.csv', index=False, header=False)
validation_dataset.to_csv('validation_dataset.csv', index=False, header=False)
```

Chargement des jeux de données dans S3

Nous devons maintenant télécharger ces deux fichiers sur S3. Nous pourrions utiliser n'importe quel compartiment, et ici nous utiliserons le compartiment par défaut facilement créé par SageMaker dans la région dans laquelle nous fonctionnons. Nous pouvons trouver son nom avec l'API Sagemaker.session.default_bucket () :

```
sess = sagemaker.Session()
bucket = sess.default_bucket()
prefix = 'boston-housing'
```

Enfin, nous utilisons l'API SageMaker.session.upload_data () pour télécharger les deux fichiers CSV dans le compartiment par défaut. Ici, les jeux de données de formation et de validation sont constitués d'un fichier chacun, mais nous pourrions télécharger plusieurs fichiers si nécessaire. Pour cette raison, nous devons télécharger les jeux de données sous différents préfixes S3, afin que leurs fichiers ne soient pas mélangés.

```
training_data_path = sess.upload_data(path='training_dataset.csv', key_prefix=prefix + '/input/training')
validation_data_path = sess.upload_data(path='validation_dataset.csv', key_prefix=prefix + '/input/validation')
print(training_data_path)
print(validation_data_path)
```

Les deux chemins S3 ressemblent à ceci. Bien sûr, le numéro de compte dans le nom du compartiment par défaut sera différent.

```
s3://sagemaker-eu-west-1-123456789012/boston-housing/input/training/training_dataset.csv
s3://sagemaker-eu-west-1-123456789012/boston-housing/input/validation/validation_dataset.csv
```

Maintenant que les données sont prêtes dans S3, nous pouvons configurer le travail d'entraînement.

Entraînement du modèle

L'objet Estimator (Sagemaker.Estimator.Estimator) est la pierre angulaire de l'entraînement du modèle. Il vous permet de sélectionner l'algorithme approprié, de définir vos exigences en matière d'infrastructure de formation et plus encore. Les algorithmes SageMaker sont emballés dans des conteneurs Docker. En utilisant boto3 et l'API image_uris.retrieve (), nous pouvons facilement trouver le nom de l'algorithme LinearLearner dans notre région.

Exemple de code :

```
import boto3
from sagemaker import image_uris

region = boto3.Session().region_name
container = image_uris.retrieve('xgboost', region, version='latest')

from sagemaker.estimator import Estimator

role = sagemaker.get_execution_role()
```

Maintenant que nous connaissons le nom du conteneur, nous pouvons configurer notre travail d'entraînement avec l'objet Estimator. Outre le nom du conteneur, nous transmettons également le rôle IAM que les instances SageMaker utiliseront, le type d'instance et le nombre d'instances à utiliser pour la formation, ainsi que l'emplacement de sortie du modèle.

```
xgb_estimator = Estimator(container,
  role=role,
  instance_count=1,
  instance_type='ml.m5.large',
  output_path='s3://{}/{}/output'.format(bucket, prefix)
)
```

En regardant les hyperparamètres https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost_hyperparameters.html nous voyons que le seul requis est num_round. Comme il n'est pas évident de définir quelle valeur, nous allons opter pour une grande valeur, et allons également définir le paramètre early_stopping_rounds afin d'éviter le surajustement. Bien sûr, nous devons fixer l'objectif d'un problème de régression.

```
xgb_estimator.set_hyperparameters(objective='reg:linear',
                                  num_round=200,
                                  early_stopping_rounds=10)
```

Puis, nous définissons les canaux de données : un canal est une source nommée de données transmises à un estimateur SageMaker. Tous les algorithmes intégrés nécessitent au moins un canal d'entraînement, et beaucoup acceptent également des canaux supplémentaires pour la validation et le test. Ici, nous avons deux canaux, qui fournissent tous les deux des données au format CSV. L'API TrainingInput () nous permet de définir leur emplacement, leur format, qu'ils soient compressés ou non, etc.

```
training_data_channel  = sagemaker.TrainingInput(s3_data=training
_data_path, content_type='text/csv')
validation_data_channel = sagemaker.TrainingInput(s3_data=validation
_data_path, content_type='text/csv')
```

Nous lançons ensuite le travail d'entraînement :

```
xgb_data = {'train': training_data_channel, 'validation': validation_data
_channel}

xgb_estimator.fit(xgb_data)
```

Le travail a duré 22 tours, ce qui signifie que l'arrêt anticipé a été déclenché. En regardant le journal d'entraînement, nous voyons que le round #12 était en fait le meilleur, avec une erreur (rmse) de 2,43126.

```
[12]#011train-rmse:1.25702#011validation-rmse:2.43126
<output removed>
[22]#011train-rmse:0.722193#011validation-rmse:2.43355
```

Nous créons un nom unique pour notre point de terminaison :

```
from time import strftime, gmtime
timestamp = strftime('%d-%H-%M-%S', gmtime())
endpoint_name = 'xgb-demo-'+timestamp
print(endpoint_name)

xgb-demo-14-08-23-23
```

Le déploiement prend une ligne de code.

```
xgb_predictor = xgb_estimator.deploy(
  endpoint_name=endpoint_name,
  initial_instance_count=1,
  instance_type='ml.t2.medium')
```

Une fois le modèle déployé, nous utilisons l'API predict () pour lui envoyer un échantillon CSV.

```
test_sample = '0.00632,18.00,2.310,0,0.5380,6.5750,65.20,4.0900,1,296.0,
15.30,4.98'

xgb_predictor.serializer = sagemaker.serializers.CSVSerializer()
xgb_predictor.deserializer = sagemaker.deserializers.CSVDeserializer()

response = xgb_predictor.predict(test_sample)
print(response)
```

Le résultat nous dit que cette maison devrait coûter $23,754.

```
[['23.73023223876953']]


response = runtime.invoke_endpoint(EndpointName=endpoint_name,
                                   ContentType='text/csv',
                                   Body=test_sample)

print(response['Body'].read())
```

Enfin, nous supprimons le point de terminaison lorsque nous avons terminé.

```
xgb_predictor.delete_endpoint()
```

Conclusion

Dans cette prise en main, vous avez appris sur le flux de travail SageMaker et comment l'implémenter avec l'API du SDK SageMaker, sans jamais vous soucier de l'infrastructure.

Vous trouverez plus d'informations ici :
https://aws.amazon.com/sagemaker
https://github.com/aws/amazon-sagemaker-examples
https://www.amazon.fr/Learn-Amazon-SageMaker-developers-scientists/dp/180020891X/

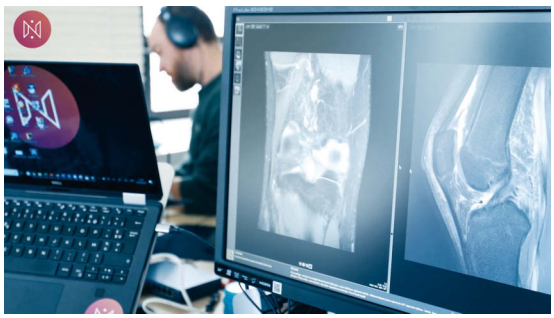
Incepto Medical : simplifier l'IA

Dans le domaine médical, la masse d'informations et de données structurées et non structurées est énorme et de natures très diverses : IRM, scanneurs, comptes-rendus, analyses médicales, dossiers patients, etc. L'imagerie médicale est un défi en soi car elle génère beaucoup d'images et de données qu'il faut stocker et analyser pour fournir les bons diagnostics. L'interprétation par les médecins reste essentielle mais il est possible de les aider pour se focaliser sur le patient.

François Tonic

Incepto Medical est une jeune startup française. Une dizaine de data scientists aide à concevoir les modèles d'analyses et de machine learning. Le monde médical est un domaine particulier où les réglementations sont fortes et la certification des logiciels prend du temps. Comme nous l'a indiqué Alexandre Lemaresquier (responsable SecDevOps), il faut environ 1 an de travail pour développer, concevoir les algorithmes et sortir le service. Il y a un important travail pour définir les modèles et les entraîner sur un volume de données conséquent pour affiner les résultats. Puis les algorithmes passent les validations.

« Nous essayons jusqu'à obtenir les bons résultats. L'image médicale est très lourde, avec beaucoup de données. Il faut réaliser un important travail de prétraitement pour nettoyer la donnée brute. Puis, il faut identifier les parties intéressantes des images, ce qui nécessite beaucoup d'entraînement pour que les modèles de Machine Learning reconnaissent ces zones. » explique Alexandre.



Décharger les équipes de la plomberie !

Le métier d'Incepto Medical n'est pas de gérer ou de créer des services de machine learning ni les infrastructures qui vont avec. C'est là que les services dédiés d'AWS prennent tout leur sens.

Incepto Medical utilise principalement 2 flux : un workflow d'apprentissage et un workflow pour la production. L'apprentissage est très consommateur de puissance GPU. N'oublions pas que pour entraîner les modèles, il faut beaucoup de puissance et les GPU sont idéaux pour cela. Les instances GPU d'AWS sont intensivement utilisées. A chaque ajustement des modèles et des algorithmes, il faut relancer des batteries de tests. Parmi les services utilisés, on retrouve Amazon FSx. FSx gère les systèmes de fichiers. Ce service supporte Windows File Server et Lustre.

Les autres piles techniques sont classiques : Argo et Kubernetes. Les algorithmes sont développés en Python. Pour le stockage, la startup utilise à la fois SFx et S3. S3 est dévolu aux images. Pour pouvoir monter en charger, notamment durant l'analyse des images, Kubernetes joue pleinement son rôle d'orchestrateur pour ajuster le cluster de conteneurs.

« Nous devons exécuter beaucoup de services, de codes et d'algorithmes en peu de temps. Pour apporter la puissance nécessaire, nous utilisons des instances P2 » indique Alexandre. Actuellement, les P3 (NVidia V100) sont utilisés avec les mécanismes d'auto-scaling et en spot instances pour les entraînements des algos. Les P2 (NVidia K800), ou des P3, sont utilisés en production, en fonction des rendements de l'algorithme et du ratio coût/bénéfice.

Les data scientists d'Incepto Medical sont tous développeurs Python. « Nous faisons aussi du RUST et de l'Electron. » poursuit Alexandre.

Des services en ligne

« Nos services sont directement disponibles en mode SaaS. Ils sont déployés sur AWS. Nous avions envisagé de les monter en interne mais ce n'est pas le mode de fonctionnement d'une startup, ni notre métier. On préfère investir dans les compétences et non dans l'infrastructure. » précise Alexandre.

La startup laisse le choix aux utilisateurs :

- Déploiement sur site : typiquement dans les serveurs des hôpitaux ;
- En mode SaaS.

En on-premise, Incepto Medical déploie les connexions avec les services AWS pour pouvoir traiter les images et les données.

La startup propose deux solutions :

- Développer et proposer nos propres algorithmes. Ils sont toujours conçus en collaboration avec des groupes médicaux ;
- Proposer une plateforme de distribution d'algorithmes de partenaires que l'on met à disposition.

« Créer des algorithmes et des modèles de machine learning pour toutes les maladies est un défi. Cela demande beaucoup de temps. Il existe déjà de nombreux algorithmes pour de nombreuses pathologies. » explique Alexandre.

AWS – TENSORFLOW – CUDA

Dans le cadre du développement des modèles mathématiques d'apprentissage profond (deep learning) en imagerie médicale, nous sommes convaincus de la pertinence des produits AWS P3 pour le calcul GPU et AWS FSX pour le stockage de la donnée.

Nous avons constaté des gains entre la génération P2 (NVidia K80) et la génération P3 (NVidia V100) de l'ordre de X4 durant les entraînements avec nos jeux de données et modèles dédiés à l'imagerie médicale. Cela nous a permis de réduire nos temps d'expérimentations de la journée à quelques heures et par conséquent nous avons plus de retours d'expériences rapidement et cela sans avoir à modifier une ligne de code. D'ailleurs la relation entre le coût et le gain de performances étant linéaire cela est très avantageux.

De plus, l'exploitation des P3 grâce à la technologie SPOT nous permet d'obtenir des réductions très significatives sur notre dépense GPU si nous exploitons tout en offre à la demande. Les instances spots ont la caractéristique de pouvoir être arrêtées par AWS à n'importe quel moment en contrepartie de ce tarif très attractif. D'un point de vue logiciel, cette caractéristique demande de coder selon les bonnes pratiques issues du monde du calcul à haute performance (HPC), à savoir notamment de mettre en place les sauvegardes des états d'expériences de manière très fréquentes (mécanisme des checkpoints en Tensorflow) afin de pouvoir redémarrer les expériences à tout moment sur un état déjà calculé.

Grâce à Tensorflow et surtout son API de haut-niveau Keras, nous pouvons nous concentrer sur la création des modèles sans devoir penser pour l'instant à des couches de bas niveau comme CUDA. La complexité des problèmes et des modèles en imagerie médicale étant telle que, pour l'instant, coder un kernel CUDA dédié à une tâche est de l'ordre de l'optimisation.

Il faut aussi savoir que nous exploitons intensivement une technologie comme Argo qui nous permet de retrouver des fonctionnalités disponibles en général sur les clusters HPC plus traditionnels comme la possibilité de planifier avec PBS ou Slurm ; l'articulation de cette technologie avec les autres (P3, FSX, Spot, Tensorflow, Keras) nous permet d'entraîner de manière très confortable nos modèles.

Entraîner et déployer un modèle en utilisant Tensorflow 2 et Object Detection API avec Amazon SageMaker

Othmane Hamzaoui

Consultant Data Science au sein des équipes AWS Professional Services France. Othmane accompagne les clients AWS afin de développer et mettre en production des solutions Machine Learning.

Sofian Hamiti

Architecte de solutions AWS spécialisé en AI/ML. Il aide les entreprises à construire des capacités de Machine Learning servant à résoudre leurs défis commerciaux.

Avec la croissance rapide des techniques de détection d'objets, plusieurs librairies, avec des modèles pré-entrainés, ont été développées pour accélérer le développement de modèle Machine Learning. GluonCV, Detectron2 et TensorFlow Object Detection API sont des librairies de vision par ordinateur avec des modèles pré-entrainés. Dans cet article, nous utilisons Amazon SageMaker [1] créer, entraîner et déployer un modèle EfficientDet [2] à l'aide de l'API TensorFlow Object Detection [3]. Ce modèle est construit en utilisant TensorFlow 2 et facilite la création, l'entraînement et le déploiement de modèles de détection d'objets. De plus, nous utiliserons TensorFlow 2 Detection Model Zoo. Il s'agit d'une collection de modèles de détection pré-entraînés pour accélérer nos efforts.

SageMaker est un service managé permettant aux développeurs et aux scientifiques des données de créer, de former et de déployer rapidement et facilement des modèles de machine learning (ML). SageMaker facilite chaque étape du processus de machine learning.

Laissez-vous guider !

Cet article vous montrera comment effectuer les opérations suivantes :

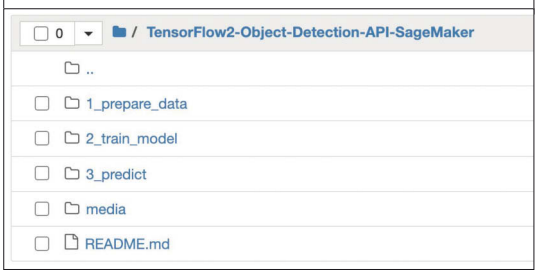
- Annoter des images à l'aide de SageMaker Ground Truth ;
- Générer les fichiers TFRecords à partir de notre dataset en utilisant SageMaker Processing ;
- Entraîner un modèle EfficientDET avec TF2 sur Amazon SageMaker ;
- Surveiller l'entraînement du modèle avec Tensorboard et SageMaker Debugger ;
- Déployer le modèle dans un SageMaker endpoint et visualiser les prédictions.

POUR ALLER PLUS VITE

Si vous souhaitez essayer chaque étape vous-même, assurez-vous que vous avez les éléments suivants en place :

- Un compte AWS [4] ;
- Un bucket Amazon S3 [5] ;
- Une Amazon SageMaker Notebook Instance [6] ;
- Ce répertoire GitHub [7] cloné dans l'instance notebook dans Amazon SageMaker.

Le répertoire de code contient les dossiers suivants avec un guide pas à pas via des notebook Jupyter :



ÉTAPE 1 :
Préparer les données

Vous pouvez suivre cette section en exécutant les cellules de ce notebook [8].

Le jeu de données

Nous utiliserons un jeu de données de http://inaturalist.org/ et entraînerons un modèle pour reconnaître des abeilles à partir d'images RGB. Ce dataset contient 500 images d'abeilles qui ont été publiées par des utilisateurs de Inaturalist dans le but d'enregistrer l'observation et l'identification. Nous n'avons utilisé que des images que leurs utilisateurs ont placées sous la licence CC0. **Figure 1**

Figure 1

Nous avons placé le dataset dans une archive zip que vous pouvez télécharger à partir de ce bucket S3 [9] ou en suivant les instructions du notebook `prepare_data.ipnb` dans votre instance. L'archive contient 500 images `.jpg` et un fichier `output.manifest` que nous vous expliquerons plus tard dans le blog. Nous avons également 10 images de test dans le dossier `3_predict/test_images` qui seront utilisées pour visualiser nos prédictions de modèle plus tard.

Annoter les images

Pour entraîner un modèle ML, vous avez besoin de datasets volumineux, de hautes qualités, et annotés. L'annotation de milliers d'images peut devenir fastidieuse et prendre beaucoup de temps. Heureusement, Amazon SageMaker Ground Truth [10] facilite le crowdsourcing de cette tâche. Ground Truth offre un accès facile à des annotations humaines publiques et privées pour vos données. Il fournit des tâches

d'étiquetage des interfaces pour les tâches d'annotation courantes, y compris pour la détection d'objets.

Vous pouvez maintenant passer à la création d'une tâche d'étiquetage dans SageMaker Ground Truth. Ici nous ne couvrons pas chaque étape de cette procédure. Il est déjà abordé en détail par cet excellent article [11]. Pour notre dataset, nous avons également suivi cet article [12] pour créer des instructions d'étiquetage de qualité. **Figure 2**
À la fin d'une tâche d'étiquetage, SageMaker Ground Truth enregistre un fichier output.manifest dans S3 où chaque ligne correspond à une image, ses annotations, ainsi que quelques métadonnées.

Par exemple :

```
{"source-ref":"s3://sagemaker-remars/datasets/na-bees/500/10006450.jpg"
,"bees-500":{"annotations":[{"class_id":0,"width":95.39999999999998,
"top":256.2,"height":86.80000000000001,"left":177}],"image_size":[{"
width":500,"depth":3,"height":500}]},"bees-500-metadata":{…class-map":
{"0":"bee"},"human-annotated":"yes","objects":[{"confidence":0.75}],
"creation-date":…}
```

Pour aller plus vite, nous avons déjà effectué un travail d'étiquetage appelé bees-500 et avons inclus le fichier output.manifest dans l'archive dataset.zip. Dans le notebook fourni, nous chargeons ce dataset dans le bucket S3 par défaut avant la préparation des données.

Générer les fichiers TFRecords et le dictionnaire des classes à partir du dataset

Pour utiliser notre dataset dans l'API TensorFlow Object Detection, nous devons d'abord combiner ses images et annotations et les convertir au format de fichier TFRecord. Le format TFRecord [13] est un format simple pour stocker une séquence d'enregistrements binaires, ce qui aide à la lecture des données et à l'efficacité de leur traitement. Nous devons également générer un dictionnaire des classes [14], qui définit la correspondance entre l'identifiant numérique d'une classe et son nom.

SageMaker Processing permet d'exécuter les tâches de traitement de données précédant la phase d'entraînement de modèles de manière managé et scalable. Dans le notebook fourni, nous définissons un job SageMaker Processing utilisant notre propre conteneur docker pour le traitement des données [15]. Nous construisons d'abord un conteneur docker avec l'image Tensorflow nécessaire, les librairies Python et le code pour exécuter cette étape et le publier dans un répertoire Amazon ECR. Nous lançons ensuite un job SageMaker Processing qui exécute le conteneur et prépare les données pour l'entraînement du modèle.

Le Processing job prend les images .jpg, output.manifest et le dictionnaire des classes comme entrées depuis Amazon S3. Il sépare le dataset en deux parties : les données d'entraînement et de validation. Ensuite il génère les fichiers TFRecord et label_map.pbtxt, et les envoie dans le chemin S3 de notre choix.

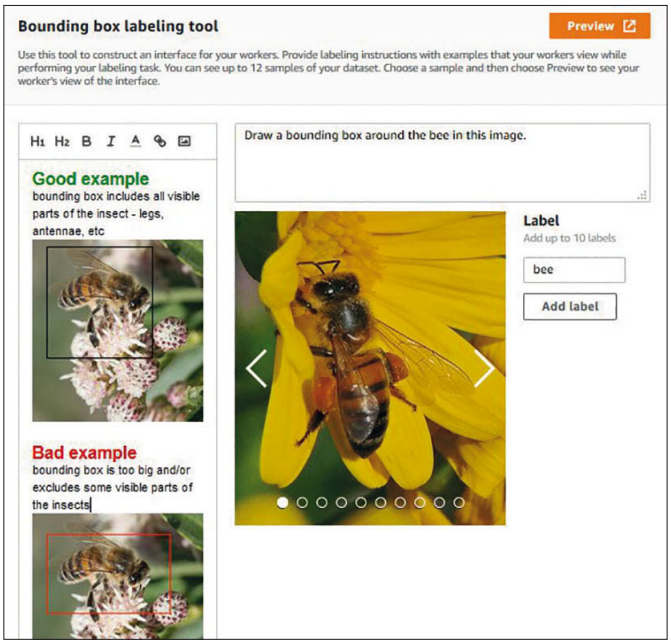


Figure 2

```
data_processor = Processor(role=role,
                           image_uri=container,
                           instance_count=1,
                           instance_type='ml.m5.xlarge',
                           volume_size_in_gb=30,
                           max_runtime_in_seconds=1200,
                           base_job_name='tf2-object-detection')
```

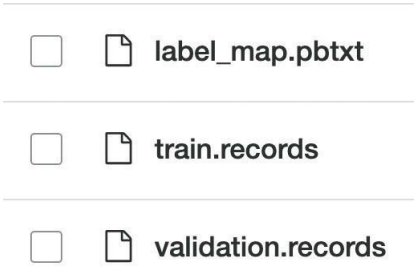
```
input_folder = '/opt/ml/processing/input'
ground_truth_manifest = '/opt/ml/processing/input/output.manifest'
label_map = '{"0": "bee"}' # chaque classe ID doit correspondre à l'équivalent
 lisible par l'homme
output_folder = '/opt/ml/processing/output'
```

```
data_processor.run(
  arguments= [
    f'--input={input_folder}',
    f'--ground_truth_manifest={ground_truth_manifest}',
    f'--label_map={label_map}',
    f'--output={output_folder}'
  ],
  inputs = [
    ProcessingInput(
      input_name='input',
      source=s3_input,
      destination=input_folder
    )
  ],
  outputs= [
    ProcessingOutput(
      output_name='tfrecords',
      source=output_folder,
      destination=f's3://{bucket}/data/bees/tfrecords'
    )
  ]
```

Sur un total de 500 images, nous allons en utiliser 450 pour l'entraînement et 50 pour la validation. Pendant l'entraîne-

ment, l'algorithme utilisera le premier ensemble pour entraîner le modèle et le second pour l'évaluation.

Vous devriez vous retrouver avec trois fichiers nommés label_map.pbtxt, train.records et validation.records dans la destination S3 que vous avez définie (c'est-à-dire : 's3 ://{bucket}/data/bees/tfrecords').



Nous pouvons maintenant passer à la section d'entraînement du modèle !

ÉTAPE 2 :
Entraîner un modèle EfficientDet avec TF2 sur SageMaker

Vous pouvez suivre cette section en se référant au bloc-notes Jupyter disponible sur GitHub [16].

Créer un conteneur docker contenant TensorFlow 2 Object Detection

Il existe plusieurs possibilités pour utiliser Amazon SageMaker avec Tensorflow, dans notre cas nous allons utiliser la fonctionnalité script mode. Cette fonctionnalité nous permet de spécifier un conteneur Docker ainsi qu'un fichier "entry_point" qui sera exécuté par SageMaker pendant la phase d'entraînement.

Vous construisez d'abord le conteneur Docker personnalisé qui sera basé sur l'image docker fournie par Tensorflow [17] à savoir le l'image Docker "tensorflow/tensorflow:2.2.0-gpu". Dans ce dernier vous installez la librairie "TensorFlow Object Detection API" [18] ainsi que toute autre librairie Python dont vous aurez besoin lors de l'entraînement.

Voici un extrait du Dockerfile utilisé :

```
FROM tensorflow/tensorflow:2.2.0-gpu
...
# Install apt dependencies
RUN apt-get update && apt-get install -y \
  git \
  gpg-agent \
  python3-cairocffi \
  protobuf-compiler \
  python3-lxml \
  wget

# COPY the "Tensorflow Object Detection API" library
COPY models/research/object_detection /home/tensorflow/models/
research/object_detection
...
```

Afin que le conteneur Docker soit accessible depuis SageMaker, il faut utiliser Amazon Elastic Container Registry (ECR). ECR est un registre de conteneurs Docker permettant de stocker, gérer et déployer facilement des images de conteneur. Vous pouvez effectuer cela en utilisant les commandes suivantes :

```
image= "nom_du_conteneur"
aws_account = ...
aws_region = ...
fullname="${aws_account}.dkr.ecr.${aws_region}.amazonaws.com/${image}
:latest"

#Go to the docker directory in the git repository
cd ../TensorFlow2-Object-Detection-API-SageMaker/2_train_model/docker/

#Create the docker image
docker build --no-cache -t ${image} -f Dockerfile .

# Get the login command from ECR and execute it directly
aws ecr get-login --region ${region} --no-include-email

# Pushing the image to ECR
docker push ${fullname}
```

Configurer le monitoring en utilisant Tensorboard et SageMaker Debugger

Avant de procéder à l'entraînement, vous allez mettre en place des mesures d'observabilité reliées aux entraînements des modèles. Vous utilisez Tensorboard, le kit de visualisation de Tensorflow ainsi que Amazon SageMaker Debugger, une fonctionnalité permettant d'assurer la visibilité en temps réel et de manière scalable des entraînements SageMaker.

Vous commencez par l'instanciation de l'objet "TensorBoardOutputConfig" disponible à travers le SDK SageMaker [19]. Deux paramètres sont précisés, le premier (c.-à-d.: "s3_output_path") étant le chemin vers S3 dans lequel vous avez enregistré les points de contrôle Tensorflow. Le deuxième paramètre (c.-à-d. "container_local_output_path") est le chemin local du conteneur Docker dans lequel Tensorflow sera configuré.

Le code suivant illustre cela :

```
from sagemaker.debugger import TensorBoardOutputConfig

tensorboard_output_config = TensorBoardOutputConfig(
  s3_output_path=tensorboard_s3_prefix,
  container_local_output_path='/opt/training/'
)
```

Pendant la phase d'entraînement, vous allez pouvoir consulter en temps réel les performances du modèle, et cela en lisant depuis S3 et en les affichant directement sur Tensorboard.

Entraîner un modèle Tensorflow 2 Object Detection en utilisant Sagemaker

Tensorflow propose une collection de modèles pré-entraînés sur le dataset COCO 17 [20] qui permettent d'effectuer la classification d'images "out-of-the-box". Dans cet article nous avons choisi le modèle EfficientDET comme exemple et procéderons à son re-entraînement en utilisant le dataset de la section 1.

Il nous faut :

- Télécharger le fichier .tar.gz correspondant au modèle depuis le repository de Tensorflow. Ce fichier contiendra le modèle (sous forme de Tensorflow checkpoints) ainsi qu'un fichier de configuration nommé "pipeline.config".
- Configurer les différents aspects du modèle et son pipeline d'entraînement en modifiant le fichier "pipeline.config". Nous détaillerons quelques composantes dans l'exemple ci-dessous.

L'objet "train_input_reader" :

```
{...
train_input_reader: {
 #Un fichier précisant le mapping entre les identifiant des labels et leur nom
 #Example: {"0":"bee", "1": "dog"}
 label_map_path: "/opt/ml/input/data/train/label_map.pbtxt"
 tf_record_input_reader {
  #L'endroit contenant les données d'entraînement
  input_path: "/opt/ml/input/data/train/train.records"
 }
}
...}
```

L'objet "train_config" :

```
train_config {
 #la taille du batch d'entraînement
 batch_size: 64
 #le nombre d'itérations d'entraînement
 num_steps: 300000
 ...
 #quel optimiseur nous utilisons et sa configuration
 optimizer {
  momentum_optimizer {
   learning_rate {
    cosine_decay_learning_rate {
     learning_rate_base: 0.07999999821186066
     total_steps: 1000
     warmup_learning_rate: 0.0010000000474974513
     warmup_steps: 100
    }
   }
   momentum_optimizer_value: 0.8999999761581421
  }
  use_moving_average: false
 }
```

Une fois le fichier de configuration complet, vous procédez à l'intégration du modèle avec SageMaker. Vous pouvez utiliser le script bash "run_training.sh" comme point d'entrée d'exécution pour les jobs d'entraînements d'Amazon Sagemaker.
Il s'agit du script principal qui sera exécuté par SageMaker pendant l'entraînement du modèle et qui effectuera les opérations suivantes :

- Lancer l'entraînement du modèle en fonction des hyper paramètres spécifiés ;
- Lancer l'évaluation du modèle en fonction du dernier "tensorflow checkpoint" pendant l'entraînement ;
- Préparation du modèle entraîné pour l'inférence à l'aide du script "export_main_v2.py" de Tensorflow.

Vous êtes prêt à lancer l'entraînement du modèle en utilisant le code suivant :

```
hyperparameters = {
  "model_dir":"/opt/training",
  "pipeline_config_path": "pipeline.config",
  "num_train_steps": 1000,
  "sample_1_of_n_eval_examples": 1
}

estimator = CustomFramework(image_name=container,
                    role=role,
                    entry_point='run_training.sh',
                    source_dir='source_dir/',
                    train_instance_count=1,
                    train_instance_type='ml.p3.8xlarge',
                    hyperparameters=hyperparameters,
                tensorboard_output_config=tensorboard_output_config,
                    base_job_name='tf2-object-detection')

estimator.fit(inputs, wait=False)
```

Au cours de l'exécution du job d'entraînement par SageMaker, vous allez utiliser Tensorboard afin d'évaluer 1/la performance du modèle pendant l'entraînement 2/les résultats du modèle pendant la phase d'évaluation.
Commençons par les mesures de performance émises par Tensorflow pendant la phase d'entraînement: Étant donné que vous avez configuré SageMaker Debugger, la variable valeur "tensorboard_s3_output_path" fera référence au chemin S3 qui a été configuré ci-dessus, et le préfix "/train" contiendra les mesures de performance d'entraînement. Vous pouvez aussi récupérer cette valeur à l'aide de l'object Estimator en utilisant la commande suivante :

```
job_artifacts_path = estimator.latest_job_tensorboard_artifacts_path()
tensorboard_s3_output_path = f'{job_artifacts_path}/train'
```

Afin de lancer le serveur Tensorboard en local, vous utilisez la commande suivante :

```
!F_CPP_MIN_LOG_LEVEL=3 AWS_REGION=<ADD YOUR REGION HERE>
 tensorboard --logdir=$tensorboard_s3_output_path
```

Enfin, vous pouvez ainsi ouvrir Tensorboard en visitant cette URL :

```
https://le-nom-de-votre-instance-sagemaker.notebook.your-region.
sagemaker.aws/proxy/6006/
```

Figure 3

Vous pouvez également consulter les résultats d'évaluation générés par Tensorflow après l'entraînement et qui sont accessibles sous le préfix "/eval" :

```
tensorboard_s3_output_path = f'{job_artifacts_path}/eval'
region_name = 'eu-west-1'

!F_CPP_MIN_LOG_LEVEL=3 AWS_REGION=$region_name tensorboard
 —logdir=$tensorboard_s3_output_path
```

Les résultats d'évaluation générés par Tensorflow permettent de comparer, pour un échantillon, les ground-truth labels (image de droite) et prédictions du modèle (image de gauche). **Figure 4**

Figure 3

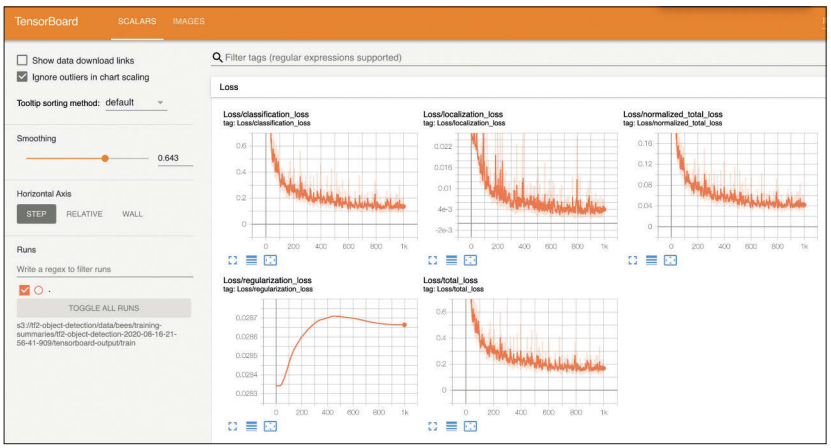


Figure 4

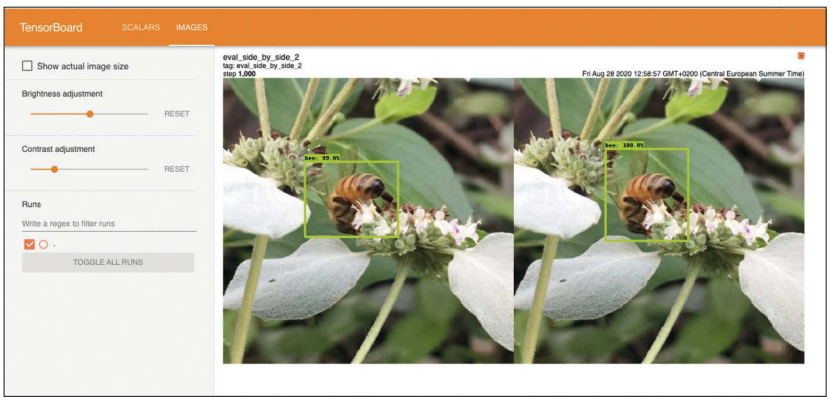
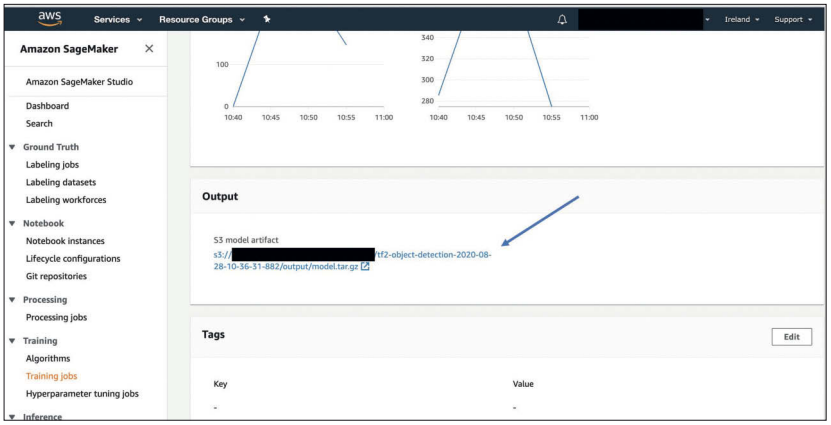


Figure 5



ÉTAPE 3 :
Object Detection en utilisant un endpoint SageMaker

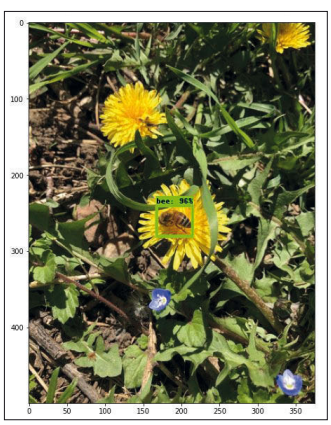
Une fois l'entraînement terminé, le modèle est transformé vers un format protobuf (.pb) puis compressé en ".tar.gz" par SageMaker et directement copié sur S3. Par ailleurs, SageMaker fournit plusieurs conteneurs Docker Tensorflow optimisés pour l'inférence [21], ce qui vous permet de facilement déployer le modèle entraîné sur une API http hébergé par SageMaker. Ce dernier vous permettra d'effectuer des prédictions en temps réel (les tâches de transformation par lots, ou batch processing, sont également disponibles pour les prédictions asynchrones et hors ligne).
Tout d'abord vous récupérez le chemin S3 dans lequel le modèle a été sauvegardé. Cela peut se faire à travers l'api boto3 ou aussi depuis la console AWS. Il suffit d'ouvrir la section "Training jobs" et sélectionner le job d'entraînement correspondant : **Figure 5**

Une fois que vous avez le chemin S3, vous pouvez déployer le endpoint en utilisant le code suivant :

```
from sagemaker.tensorflow.serving import Model

model_artefact = '<your-model-s3-path>'

model = Model(model_data=model_artefact,
        name=name_from_base('tf2-object-detection'),
        role=role,
        framework_version='2.2')

predictor = model.deploy(initial_instance_count=1, instance_type='ml.m5.xlarge')
```

Une fois que l'API de prédiction SageMaker est opérationnelle, vous pouvez effectuer des requêtes HTTPS permettant d'envoyer des images de test et par la suite évaluer les résultats à l'aide de la bibliothèque Matplotlib.

```
img = image_file_to_tensor('test_images/22673445.jpg')

input = {
 'instances': [img.tolist()]
}

detections = predictor.predict(input)['predictions'][0]
```



Nettoyage

Supprimons maintenant les ressources créées au cours de cet exemple. Il est conseillé de supprimer les ressources qui ne sont pas utilisées de façon active, afin de réduire les coûts (NDLR : nous sommes dans un paiement à l'usage). Des ressources non supprimées peuvent augmenter votre facture. La première étape consiste à supprimer l'endpoint SageMaker :

1. Sur la console SageMaker et sous la section « Inference/Prédiction », cliquez sur « Endpoints/Points de terminaison » ;
2. Sélectionner l'endpoint SageMaker que vous avez déployé ;
3. Cliquez sur Actions et sélectionnez Supprimer, en haut de la page.

La dernière étape sera de supprimer l'ensemble des données d'entraînement dans votre compartiment S3 :

1. Accédez à la console S3 en utilisant le menu Services en haut de votre console AWS et cliquez sur Compartiments dans le menu à votre gauche. Vous pourrez voir à votre droite la liste de tous les compartiments S3 existant dans votre compte ;
2. Trouvez et sélectionnez le compartiment S3 que nous avons créé précédemment dans ce tutoriel ;
3. Une fois votre compartiment sélectionné, cliquez sur le bouton Vider en haut de la liste des compartiments et confirmez votre choix en suivant les instructions de la page suivante ;
4. Revenez à la liste des compartiments, sélectionnez à nouveau le compartiment S3 que vous venez de vider et cliquez sur le bouton Supprimer en haut de la liste. Confirmez votre choix en suivant les instructions de la page suivante.

Conclusion

Dans cet article, nous avons abordé le processus de collecte et d'annotation des données à l'aide de SageMaker GroundTruth, la préparation et la conversion des données en TFRecords et enfin l'entraînement et le déploiement d'un modèle de détection d'objets personnalisé à l'aide de l'API de détection d'objets Tensorflow et Amazon SageMaker..

A vous de détecter !

Références

1. https://aws.amazon.com/sagemaker
2. https://arxiv.org/abs/1911.09070
3. https://github.com/tensorflow/models/tree/master/research/object_detection
4. https://portal.aws.amazon.com/billing/signup
5. https://aws.amazon.com/s3
6. https://docs.aws.amazon.com/sagemaker/latest/dg/gs-setup-working-env.html
7. https://github.com/aws-samples/amazon-sagemaker-tensorflow-object-detection-api
8. https://github.com/aws-samples/amazon-sagemaker-tensorflow-object-detection-api/blob/main/1_prepare_data/prepare_data.ipynb
9. https://tf2-object-detection.s3-eu-west-1.amazonaws.com/data/bees/input/dataset.zip
10. https://aws.amazon.com/sagemaker/groundtruth
11. https://aws.amazon.com/blogs/aws/amazon-sagemaker-ground-truth-build-highly-accurate-datasets-and-reduce-labeling-costs-by-up-to-70
12. https://aws.amazon.com/blogs/machine-learning/create-high-quality-instructions-for-amazon-sagemaker-ground-truth-labeling-jobs
13. https://www.tensorflow.org/tutorials/load_data/tfrecord
14. https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/using_your_own_dataset.md)
15. https://docs.aws.amazon.com/sagemaker/latest/dg/build-your-own-processing-container.html
16. https://github.com/aws-samples/amazon-sagemaker-tensorflow-object-detection-api/blob/main/2_train_model/train_model.ipynb
17. https://hub.docker.com/r/tensorflow/tensorflow/
18. https://github.com/tensorflow/models/tree/master/research/object_detection
19. https://sagemaker.readthedocs.io/en/stable/index.html
20. http://cocodataset.org/
21. https://github.com/aws/sagemaker-tensorflow-serving-container

Créer automatiquement des modèles de machine learning avec une visibilité totale (Python)

Bruno Medeiros de Barros

Solutions Architect au sein des équipes AWS France où il aide les clients français à innover à travers l'adoption des technologies du cloud en assurant la sécurité de leur infrastructure et de leurs données.

David Gallitelli

Solutions Architect au sein des équipes AWS France et assistant les clients français dans l'adoption des solutions basées sur l'intelligence artificielle & le Machine Learning.

En 1959, Arthur Samuel a défini le terme "Machine Learning" comme la capacité des ordinateurs à apprendre sans être explicitement programmés. En pratique, cela signifie identifier un algorithme capable d'identifier des schémas récurrents, ou patterns, d'un ensemble de données existant et utiliser ces modèles pour fournir un modèle prédictif capable de se généraliser à de nouvelles données. Depuis lors, de nombreux algorithmes de Machine Learning ont été créés, offrant aux développeurs et aux scientifiques un large choix d'options leur permettant de construire de nouvelles applications.

Cependant, cette abondance d'algorithmes crée également une difficulté : lequel choisir ? Comment pouvez-vous déterminer de manière fiable quel algorithme sera le plus performant pour votre problématique métier spécifique ? En outre, les algorithmes de Machine Learning ont généralement une longue liste de paramètres d'entraînement (également appelés hyperparamètres) qui doivent être configurés de façon optimale si vous voulez obtenir le maximum de précision de vos modèles. Pour augmenter la complexité du sujet, les algorithmes exigent également que les données soient préparées et transformées de manière spécifique (un processus appelé extraction de caractéristiques/features) pour un apprentissage optimal. Finalement, vous devez également déterminer quelle est la meilleure plateforme de calcul pour entraîner votre modèle afin d'optimiser l'exécution des différentes tâches.

Nous vous présentons Amazon SageMaker Autopilot [1], qui est un service managé qui vous permet de résoudre facilement toutes ces problématiques. En utilisant un simple appel d'API, ou quelques clics sur SageMaker Studio [2], il examine d'abord votre ensemble de données, et exécute un certain nombre de modèles candidats pour déterminer la combinaison optimale des étapes de prétraitement des données, la sélection de l'algorithme de Machine Learning et la configuration des hyperparamètres. Ensuite, il utilise cette combinaison pour entraîner un pipeline d'inférence, que vous pouvez facilement déployer sur un point de terminaison (endpoint) soit pour effectuer des prédictions en temps réel, soit pour obtenir des inférences sur un jeu de données entier à travers d'une transformation par lots (mode batch). Finalement, SageMaker Autopilot génère également un code Python qui vous montre exactement comment les données ont été prétraitées : non seulement vous pouvez comprendre ce qu'a fait SageMaker Autopilot, mais vous pouvez également réutiliser ce code pour effectuer d'autres réglages et modifications manuelles si vous le souhaitez.

Dans cet article, vous allez apprendre à :

1. Configurer SageMaker Studio de façon à avoir accès à SageMaker Autopilot
2. Télécharger un ensemble de données public avec S3 [3] et SageMaker Studio
3. Créer une expérience d'entraînement avec SageMaker Autopilot
4. Explorer les différents stades de l'expérience d'entraînement
5. Identifier et déployer le meilleur modèle à partir de l'expérience d'entraînement
6. Effectuer des prédictions avec le modèle déployé

Vous allez jouer le rôle d'un développeur travaillant dans une banque. Mission vous été confiée de développer un modèle de machine learning pour prédire si un client va s'inscrire pour un certificat de dépôt (CD). Le modèle sera entraîné à partir d'un ensemble de données marketing qui contiennent des informations sur les caractéristiques sociodémographiques des clients, les réactions aux événements marketing et les facteurs externes. Les données ont été étiquetées pour plus de commodité et une colonne dans l'ensemble de données indique si le client est inscrit pour un produit offert par la banque. Une version de cet ensemble de données est accessible au public sur le référentiel de ML de l'Université de Californie à Irvine [4].

Créer un modèle de ML

Nous utiliserons SageMaker Studio pour créer et déployer notre modèle en utilisant SageMaker Autopilot dans un environnement de développement machine learning entièrement intégré. SageMaker Studio fournit une interface web visuelle unique. Elle vous permettra de mettre en œuvre toutes les étapes du développement de machine learning. Avec SageMaker Studio, vous avez un accès, un contrôle et une visibilité complets sur chaque étape nécessaire à la création, l'entraînement et le déploiement de modèles.

Configurez votre environnement AWS

1. Afin de pouvoir exécuter ce tutoriel, vous devez d'abord vous connecter à votre compte AWS. Si vous n'avez pas encore de compte AWS, vous pouvez en créer un gratuitement à partir de la page d'inscription AWS [5].

Remarque : le coût d'exécution de ce tutoriel est inférieur à 10 $. Pour obtenir plus d'informations sur les coûts de ce tutoriel, vous pouvez consulter la page de tarification de SageMaker Studio [6]. À la fin de ce tutoriel, vous disposerez d'une procédure de nettoyage qui vous aidera à supprimer l'environnement que vous avez créé (au cas où vous le souhaiteriez).

2 Une fois que vous êtes connecté à votre console AWS, vous pouvez configurer votre environnement SageMaker Studio en accédant à la console SageMaker. Vous pouvez y accéder en sélectionnant SageMaker dans le menu Services en haut de la page ou en saisissant SageMaker dans la barre de recherche des Services AWS.

Remarque : dans le coin supérieur droit, assurez-vous de sélectionner une région AWS dans laquelle SageMaker Studio est disponible. Pour nos exemples, nous allons utiliser la région de Virginie du Nord (us-east-1). Pour avoir une liste des régions, consultez la page d'intégration pour SageMaker Studio [7].

3 Dans le volet de navigation de SageMaker, sélectionnez Amazon SageMaker Studio.

Remarque : si vous utilisez SageMaker Studio pour la première fois, vous devez passer par le processus d'inscription [7]. Au moment de l'inscription, vous avez le choix entre AWS Single Sign-On (AWS SSO) et AWS Identity and Access Management (IAM) comme méthode d'authentification. Si vous optez pour l'authentification IAM, vous avez le choix entre le démarrage rapide et la procédure de configuration standard. Pour plus de simplicité, ce tutoriel utilise la procédure Démarrage rapide.

4 Dans la fenêtre Mise en route, sélectionnez Démarrage rapide et indiquez un nom d'utilisateur. Pour ce tutoriel, il n'est pas nécessaire de modifier le nom d'utilisateur par défaut fourni par SageMaker Studio. **Figure 1**

5 Pour Rôle d'exécution, sélectionnez Créer un rôle IAM. Dans la boîte de dialogue qui s'ouvre, sélectionnez Tout compartiment S3 et ensuite sélectionnez Créer un rôle. SageMaker crée un rôle avec les autorisations nécessaires et l'affecte à votre instance. **Figure 2**

6 Une fois votre rôle créé, cliquez sur le bouton Soumettre dans la page de configuration de SageMaker Studio.

Remarque: Si vous avez plus d'un Amazon Virtual Public Cloud (VPC) disponible dans votre région, il vous sera peut-être demandé de sélectionner le VPC que vous souhaitez voir utiliser par SageMaker Studio. Pour les besoins de ce tutoriel, sélectionnez le VPC par défaut disponible dans votre région ainsi que l'un de ses sous-réseaux.

Télécharger l'ensemble de données

Vous n'avez pas besoin d'être un scientifique des données ou un spécialiste du Machine Learning pour créer un modèle prédictif performant pour vos applications. Il suffit de charger des données tabulaires dans un fichier dont les valeurs sont séparées par des virgules (par exemple, à partir d'un classeur ou d'une base de données) et de choisir la colonne cible pour la prédiction. À partir de cela, SageMaker Autopilot prend en charge tous les aspects importants pour construire un modèle prédictif pour vous.

1 Pour commencer, vous pouvez télécharger le fichier ZIP de l'ensemble de données à partir de l'URL https://sagemaker-sample-data-us-west-2.s3-us-west-2.amazonaws.com/autopilot/direct_marketing/bank-additional.zip et le décompresser dans un dossier local dans votre ordinateur.

2 Ouvrez la console Amazon S3 en utilisant le menu Services en haut de votre console AWS et cliquez sur le bouton Créer un compartiment. **Figure 3**

3 Dans la section Configuration générale, entrez un nom de compartiment de votre choix. Rappelez-vous que dans AWS, les noms des compartiments doivent être uniques au niveau global. N'hésitez pas à ajouter votre prénom ou toute autre chaîne de caractères mémorisables de votre choix à la fin du nom de votre compartiment au cas où vous recevriez une erreur indiquant que le nom du compartiment existe déjà. Dans mon cas, je vais utiliser le nom de compartiment disponible tutoriel-sagemaker-autopilot-programmez.

4 Pour la Région, sélectionnez exactement la même région que celle que vous avez utilisée à l'étape précédente pour configurer votre environnement SageMaker Studio. Dans notre cas, cette région sera Virginie du Nord (us-east-1). **Figure 4**

Figure 1

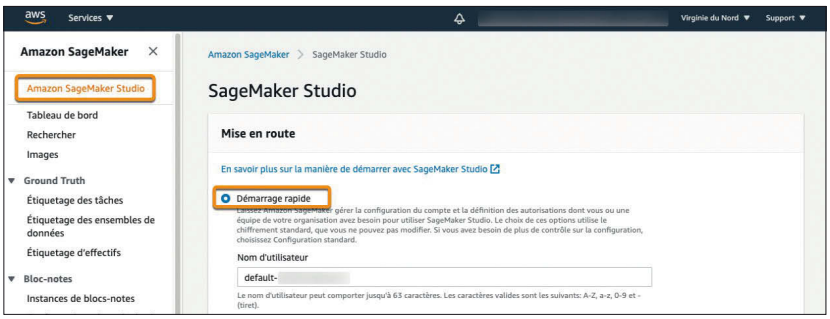


Figure 2

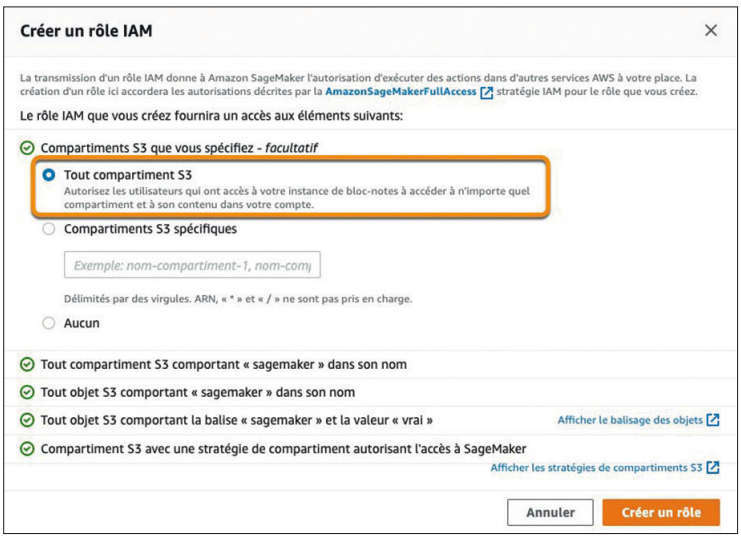


Figure 3

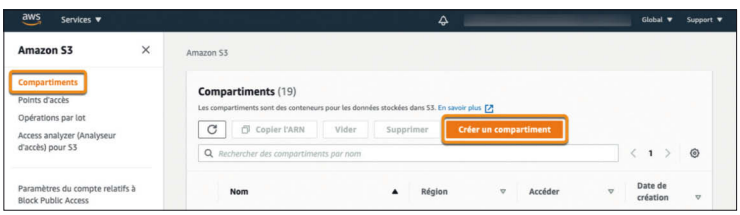


Figure 4

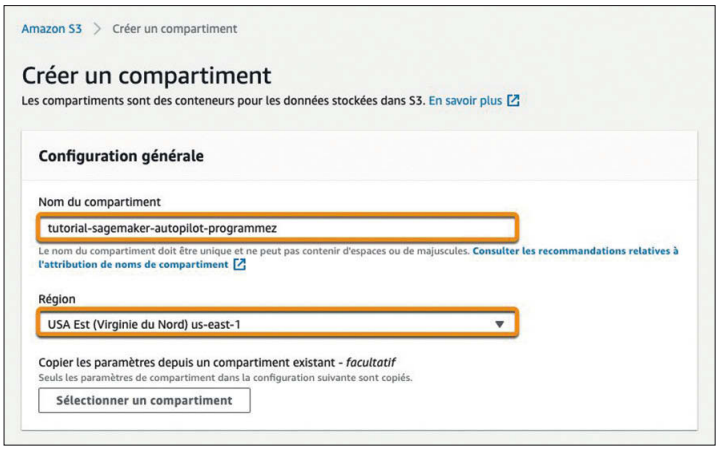


Figure 5

Figure 6

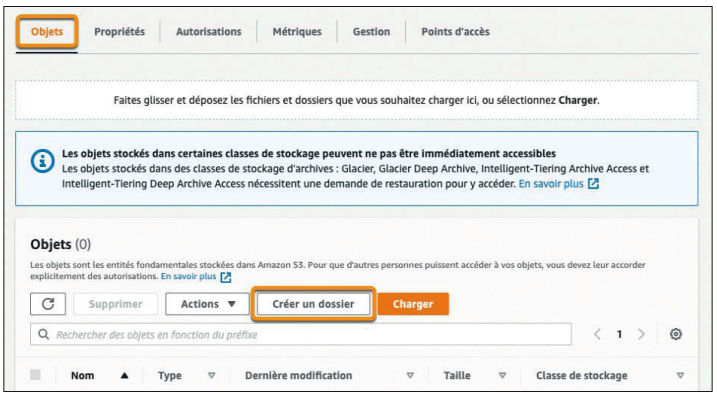


Figure 7

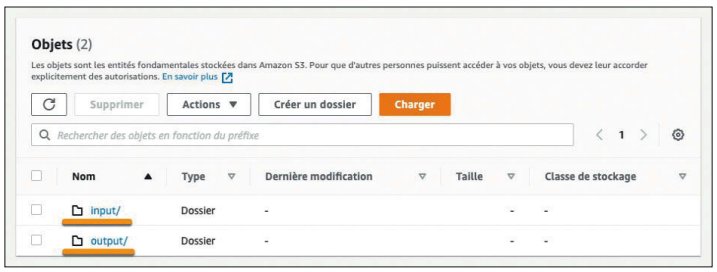


Figure 8

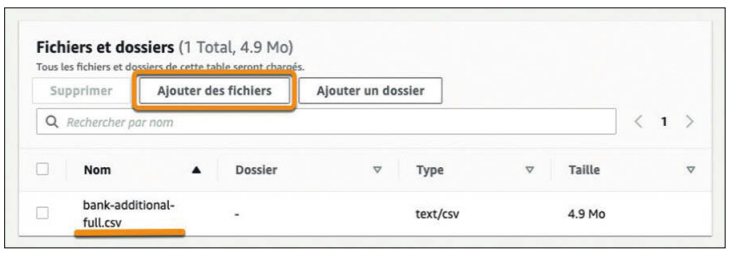
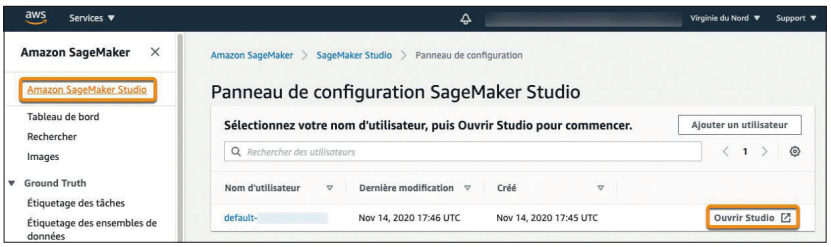


Figure 9



5 Laissez tous les autres paramètres de configuration par défaut et cliquez sur le bouton Créer un compartiment au bas de la page.

6 Trouvez votre compartiment récemment créé dans votre liste de compartiments dans la console S3 et cliquez sur son nom. Vous allez voir une page contenant plusieurs onglets contenant différentes propriétés de votre compartiment S3. **Figure 5**

7 Dans l'onglet Objets, cliquez sur le bouton Créer un dossier. Pour le nom du dossier, entrez "input". Laissez les autres paramètres par défaut et cliquez sur "Créer un dossier". **Figure 6**

8 Répétez l'étape précédente et créez un dossier nommé "output". Vous devriez alors pouvoir voir les deux dossiers créés dans l'onglet Objets de votre compartiment. **Figure 7**

9 Nous allons maintenant télécharger notre fichier de données dans le dossier input de notre compartiment S3. Vous devez pour cela localiser votre dossier input dans l'onglet Objets de votre compartiment et cliquer sur son nom. Vous pouvez alors voir la liste des objets à l'intérieur de votre dossier qui devrait être vide à ce stade.

10 Dans la section Objets de votre dossier, cliquez sur Charger.

11 Dans la section Fichiers et dossiers, cliquez sur Ajouter des fichiers, allez dans le dossier local où vous avez décompressé votre ensemble de données et sélectionnez le fichier "bank-additional-full.csv" à charger. Laissez les autres paramètres par défaut et cliquez sur Charger. **Figure 8**

Maintenant que nous avons notre compartiment S3 contenant nos données d'entrée, ainsi qu'un dossier de sortie qui sera utilisé par SageMaker Autopilot, nous sommes prêts à revenir à notre service et à créer une expérience qui permettra à SageMaker Autopilot de créer un modèle prédictif pour nous.

Créer une expérience SageMaker Autopilot

Une expérience est un recueil de tâches de traitement et d'entraînement en lien avec un même projet de machine learning. Dans les prochaines étapes, nous allons créer une nouvelle expérience sur SageMaker Studio. Pour obtenir plus d'informations sur les expériences SageMaker, consultez la page Créez un SageMaker expérience dans SageMaker Studio [8] de la documentation d'Amazon SageMaker.

1 Accédez à la console SageMaker en utilisant le menu Services en haut de votre console AWS et cliquez sur SageMaker Studio dans le menu à votre gauche. Sur le Panneau de configuration de SageMaker Studio, sélectionnez Ouvrir Studio. **Figure 9**

2 Sur le volet de navigation de gauche d'Amazon SageMaker Studio, sélectionnez l'onglet Composants et registres SageMaker (SageMaker Components and registries, en anglais), sélectionner Expériences et essais (Experiments and trials, en anglais) dans le menu composants et registres, puis cliquer sur le bouton Créer une expérience (Create Experiment, en anglais). **Figure 10**

3 Pour le nom de l'expérience, utilisez tutoriel-autopilot-programmez.

4 Pour l'emplacement des données d'entrée, choisissez l'option Trouver compartiment S3 (Find S3 bucket, en anglais) et trouvez le compartiment S3 que nous avons créé dans les étapes précédentes dans la liste des noms des compartiments S3 (S3 bucket name, en anglais). Pour le nom du fichier de l'ensemble de données (Dataset file name, en anglais), sélectionnez input/bank-additional-full.csv.

5 Pour le nom de l'attribut cible (Target, en anglais), qui correspond à l'attribut que nous voulons prédire, sélectionnez y. Cet attribut est lisible à partir des données d'entrée que nous avons chargées précédemment et nous indique si un client de la banque a souscrit à un produit bancaire qui a été proposé dans le cadre d'une campagne de marketing.

6 Pour l'emplacement des données de sortie, retrouvez à nouveau le compartiment S3 que nous avons créé dans les étapes précédentes dans la liste des noms des compartiments S3 (S3 bucket name, en anglais), et pour le nom du fichier de l'ensemble de données (Dataset file name, en anglais), sélectionnez le dossier output/.

7 Laissez tous les autres paramètres avec leurs valeurs par défaut, puis sélectionnez Créer une expérience (Create Experiment, en anglais).

Explorer les stades d'une expérience SageMaker Autopilot

Pendant que votre expérience s'exécute, vous pouvez découvrir et explorer les différents stades de l'expérience SageMaker Autopilot. Cette section donne davantage de détails sur les stades de l'expérience SageMaker Autopilot. Pour plus de détails, vous pouvez consulter la page SageMaker Autopilot sortie de bloc-notes [9].

1 Le stade de prétraitement des données et de définition des candidats identifie le type de problème à résoudre (régression linéaire, classification binaire, classification multi-classe). Il donne ensuite dix pipelines candidats. Un pipeline combine une étape de prétraitement des données (gestion des valeurs manquantes, ingénierie de nouvelles fonctionnalités, etc.) à une étape d'entraînement de modèle avec un algorithme de ML correspondant au type de problème. Une fois cette étape terminée, l'expérience passe à l'ingénierie de fonctionnalités. **Figure 11**

2 Au stade Ingénierie de fonctionnalités, l'expérience crée des ensembles de données d'entraînement et de validation pour chaque pipeline candidat, et stocke tous les artefacts dans votre compartiment S3. Pendant le stade Ingénierie de fonctionnalités, vous pouvez consulter deux blocs-notes générés automatiquement par SageMaker Autopilot:

- Le bloc-note d'exploration des données, qui contient des informations et des statistiques sur les ensembles de données analysées par SageMaker Autopilot;
- Et le bloc-note de génération de candidat, qui contient la définition de dix pipelines. En fait, il s'agit d'un bloc-note exécutable : vous pouvez reproduire exactement ce que fait la tâche de SageMaker Autopilot, comprendre la conception des différents modèles et même les optimiser à votre convenance.

Grâce à ces deux blocs-notes, vous pouvez comprendre en détail le prétraitement des données, ainsi que la conception et l'optimisation des modèles. **Figure 12**

3 Au stade Optimisation du modèle, pour chaque pipeline candidat et son ensemble de données pré-traité, SageMaker Autopilot lance une tâche de réglage de modèle automatique [10]. Les tâches d'entraînement associées explorent un large choix de valeurs des hyperparamètres et convergent rapidement vers des modèles aux performances accrues.

Une fois au bout de ce stade, la tâche est terminée. Vous pouvez voir et explorer toutes les tâches dans le Studio.

Déployer et tester votre modèle

Maintenant que votre expérience est terminée, vous pouvez choisir la meilleure optimisation de modèle et déployer le modèle vers un point de terminaison géré par SageMaker.

Déployer le meilleur modèle

1 Dans la liste Essais de votre expérience, sélectionnez la flèche en regard de la section Objectif pour trier les tâches d'optimisation dans l'ordre décroissant. La meilleure tâche d'optimisation est mise en avant par une étoile.

2 Sélectionnez la meilleure tâche d'optimisation (marquée par une étoile) et cliquez sur Déployer un modèle (Deploy Model). **Figure 13**

3 Dans la section Options de déploiement (Deployment options, en anglais), sélectionnez Prédictions en temps réel (Real Time Predictions, en anglais) et cliquez sur les

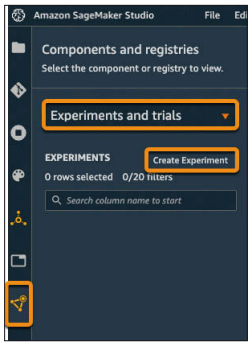


Figure 10

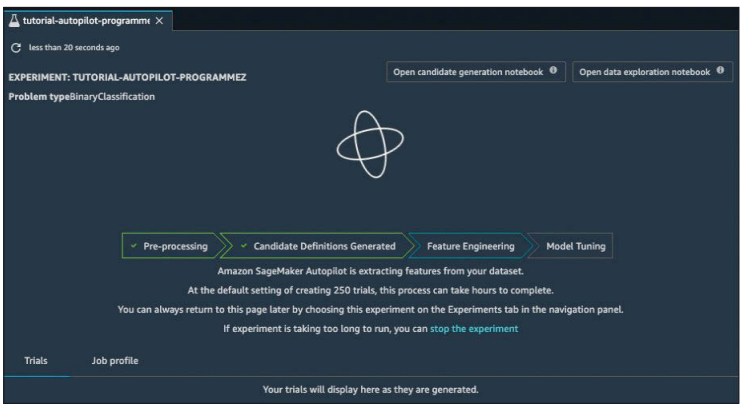


Figure 11

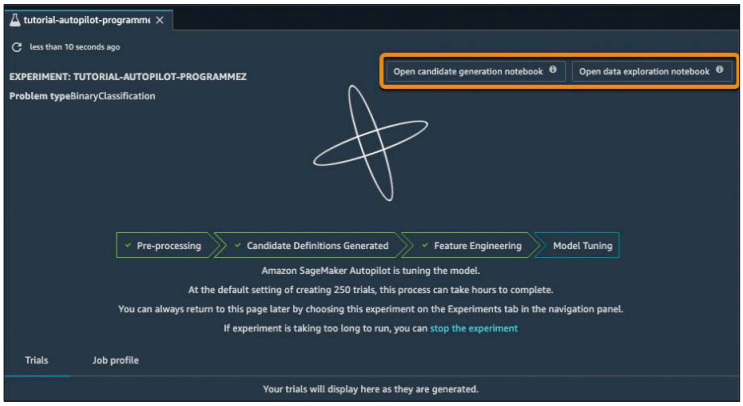


Figure 12

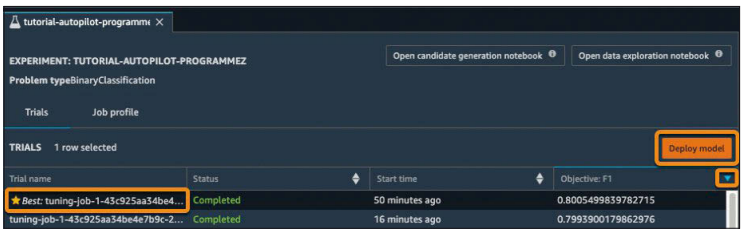


Figure 13

configurations de déploiement en temps réel (Real Time Deployment Settings, en anglais). Cela vous permettra de déployer votre modèle pour faire des prédictions en temps réel via un point de terminaison Amazon SageMaker.

4. Dans la fenêtre Déployer un modèle, nommez votre point de terminaison (Endpoint name, en anglais) tutoriel-autopilot-best-model et laissez tous les paramètres avec leurs valeurs par défaut. Cliquez sur Déployer modèle (Deploy model, en anglais). Votre modèle est déployé vers un point de terminaison HTTPS géré par Amazon SageMaker. **Figure 14**
5. Vous pouvez suivre la progression du déploiement de votre modèle en cliquant sur l'onglet Composants et registres SageMaker (SageMaker Components and registries, en anglais) sur le volet de navigation de gauche du Studio, et en sélectionnant Points de terminaison (Endpoints, en anglais) dans le menu composants et registres.
6. Dans la liste des points de terminaison, vous pouvez consulter le statut du point de terminaison que vous venez de créer dans la colonne Endpoint status. Une fois le statut du point de terminaison défini comme En service (In Service, en anglais), vous pouvez envoyer des données et recevoir des prédictions de votre point de terminaison. Vous pouvez également appliquer des filtres pour visualiser le statut des points de terminaison spécifiques dans votre environnement. **Figure 15**

Effectuer des prédictions avec votre modèle

Maintenant que le modèle est déployé, vous pouvez prédire les 2000 premiers échantillons de l'ensemble de données en utilisant l'appel InvokeEndpoint depuis l'API SageMaker. Pour effectuer des appels API vers SageMaker ainsi que vers tout autre service AWS depuis votre application, vous pouvez utiliser le SDK AWS disponible pour votre langage de programmation préféré. Au moment où cet article est rédigé, AWS fournit des SDK pour les langages C++, Go, Java, JavaScript, .NET, Node.js, PHP, Python et Ruby. Pour plus d'informations sur les SDK et autres outils de développement fournis par AWS, consultez la page Outils pour créer sur AWS [11].

Dans l'exemple qui suit, nous utilisons le AWS SDK pour Python (Boto3) pour implémenter un code qui va collecter les prédictions pour les 2000 premiers échantillons de données de notre ensemble de données et produire des métriques qui nous permettent de comparer les résultats prédits par notre modèle avec les résultats réels dans l'ensemble de données. Pour rappel, les résultats obtenus à partir de notre modèle correspondent aux prédictions concernant la souscription d'un client à un certain service bancaire, et sont représentés par les valeurs "yes" (si le modèle prédit une souscription) ou "no" (si le modèle prédit une non souscription).

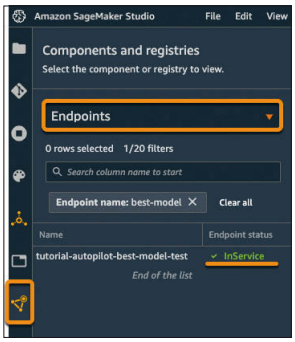


Figure 15

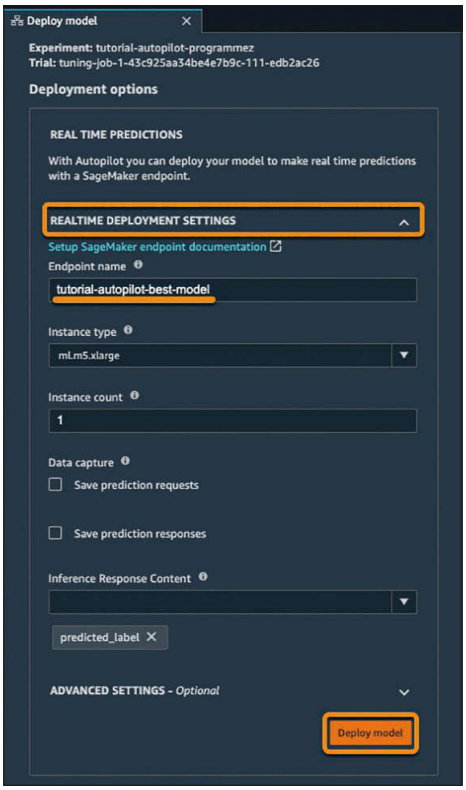


Figure 14

```
import boto3, sys

ep_name = 'tutorial-autopilot-best-model' # Si vous avez fourni un nom
 différent pour votre point de terminaison, modifiez cette variable en lui
donnant la valeur appropriée
sm_rt = boto3.Session().client('runtime.sagemaker')

tn=tp=fn=fp=count=0

with open('bank-additional-full.csv') as f:
  lines = f.readlines()
  for l in lines[1:2000]:  # Sauter l'en-tête
    l = l.split(',')     # Diviser la ligne CSV en caractéristiques
    label = l[-1]        # Stocker les labels 'yes' et 'no'
    l = l[:-1]           # Enlever les labels des données de test
    l = ','.join(l)      # Recréer les lignes CSV sans label

    response = sm_rt.invoke_endpoint(EndpointName=ep_name,
                        ContentType='text/csv',
                        Accept='text/csv', Body=l)

    response = response['Body'].read().decode("utf-8")

    if 'yes' in label:
      # L'échantillon est positif
      if 'yes' in response:
        # Vrai positif
        tp=tp+1
      else:
        # Faux négatif
        fn=fn+1
    else:
      # L'échantillon est négatif
      if 'no' in response:
        # Vrai négatif
        tn=tn+1
      else:
        # Faux positif
        fp=fp+1
    count = count+1
    if (count % 100 == 0):
```

```
        sys.stdout.write(str(count)+' ')

print ("Done")

accuracy = (tp+tn)/(tp+tn+fp+fn)
precision = tp/(tp+fp)

print ("Exactitude: %.4f, Précision: %.4f" % (accuracy, precision))
```

Remarque: Le code présenté ici peut être exécuté dans n'importe quel environnement Python 3 configuré avec les informations d'identification et les autorisations appropriées. Pour savoir comment configurer votre environnement d'exécution, consultez la documentation du SDK AWS pour le langage que vous avez choisi. Par exemple, les instructions pour configurer votre environnement afin d'utiliser le AWS SDK pour Python peuvent être consultées dans la page Boto3 Quickstart [12].

Pour des raisons pratiques, vous pouvez exécuter et tester votre code dans le Studio en créant un environnement Python 3 dans un bloc-notes. Une fois que nous avons déjà configuré SageMaker Studio avec toutes les autorisations nécessaires pour exécuter ce tutoriel dans les premières étapes, nous n'avons pas besoin de nous occuper de configurer des autorisations supplémentaires pour notre environnement d'exécution.

1. Dans SageMaker Studio, sur le menu Fichier, sélectionnez Nouveau, puis Bloc-notes. Dans la fenêtre Sélection du noyau, sélectionnez Python 3 (Data Science). **Figure 16**
2. Remarquez que le code présenté précédemment utilise un fichier CSV local pour charger les 2000 premiers échantillons de données pour lesquels nous voulons obtenir des prédictions à partir de notre modèle. Pour rendre ce fichier disponible pour notre code, dans l'onglet Navigateur de fichiers (File Brower, en anglais), cliquez sur Charger un fichier et sélectionnez le fichier "bank-additional-full.csv" dans l'ensemble de données que vous avez téléchargé précédemment pour le charger dans SageMaker Studio. **Figure 17**
3. Dans le bloc-notes que nous avons créé, copiez et collez notre code Python dans une cellule de code et cliquez sur Exécuter. Vous devriez obtenir le résultat suivant contenant la progression du nombre d'échantillons de données prévus ainsi que les mesures d'exactitude et de précision générées par votre code.

Remarque: Si vous recevez une note vous informant que le noyau de votre bloc-notes est toujours en cours de démarrage, attendez quelques minutes que le noyau soit démarré et essayez d'exécuter à nouveau la cellule.

Nettoyage

Nous allons supprimer les ressources que vous avez créées au cours de ce tutoriel. Il est conseillé de supprimer les ressources qui ne sont pas utilisées de façon active, afin de réduire les coûts. Des ressources non supprimées peuvent entraîner des frais sur votre compte.

1. Tout d'abord, nous allons supprimer le point d'extrémité où notre modèle prédictif est déployé. Retournez dans la console SageMaker, cliquez sur Points de terminaison sous la section Prédiction du menu à gauche. Vous verrez à droite la liste des points de terminaison SageMaker déployés dans votre compte dans la région en cours.
2. Sélectionnez le point de terminaison que nous avons déployé à l'aide de SageMaker Studio, cliquez sur le bouton Actions et sélectionnez Supprimer, en confirmant votre choix si nécessaire. **Figure 18**

La dernière étape sera de supprimer tous les artefacts utilisés et produits par SageMaker Autopilot dans votre S3, y compris l'ensemble des données d'entraînement et les modèles de prédiction créés.

3. Accédez à la console S3 en utilisant le menu Services en haut de votre console AWS et cliquez sur Compartiments dans le menu à votre gauche. Vous pourrez voir à votre droite la liste de tous les compartiments S3 existant dans votre compte
4. Trouvez et sélectionnez le compartiment S3 que nous avons créé précédemment dans ce tutoriel. Si nécessaire, tapez "tutorial-sagemaker-autopilot" dans la barre de recherche en haut de la page.
5. Une fois votre compartiment sélectionné, cliquez sur le bouton Vider en haut de la liste des compartiments et confirmez votre choix en suivant les instructions de la page suivante. **Figure 19**
6. Revenez à la liste des compartiments, sélectionnez à nouveau le compartiment S3 que vous venez de vider et cliquez sur le bouton Supprimer en haut de la liste. Confirmez votre choix en suivant les instructions de la page suivante.

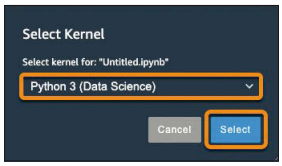


Figure 16

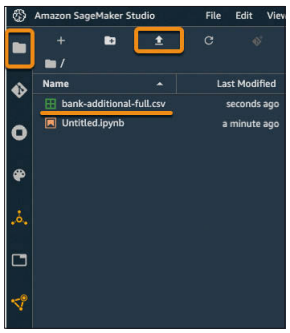


Figure 17

Figure 18

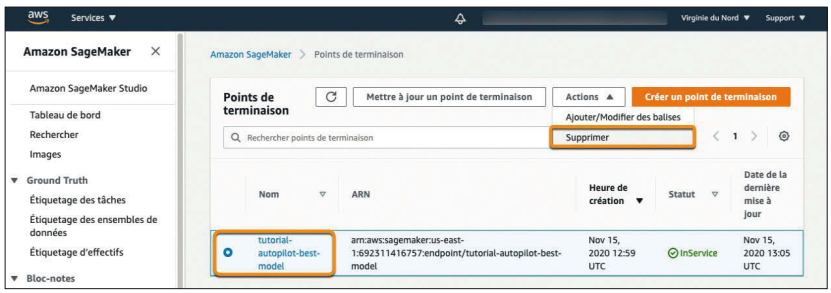
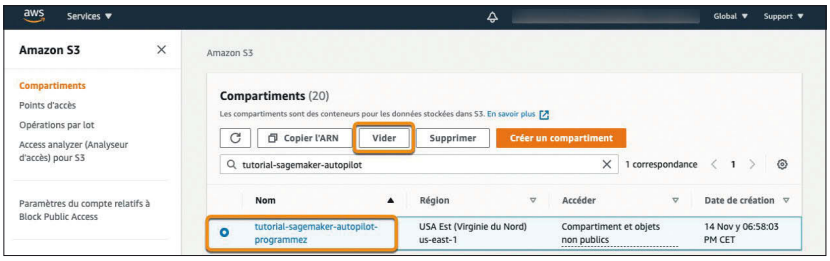


Figure 19



Liens

[1] https://aws.amazon.com/fr/sagemaker/autopilot/
[2] https://docs.aws.amazon.com/fr_fr/sagemaker/latest/dg/gs.html
[3] https://aws.amazon.com/fr/s3/
[4] https://archive.ics.uci.edu/ml/datasets/bank+marketing
[5] https://portal.aws.amazon.com/billing/signup?language=fr_fr
[6] https://docs.aws.amazon.com/fr_fr/sagemaker/latest/dg/studio-pricing.html
[7] https://docs.aws.amazon.com/fr_fr/sagemaker/latest/dg/gs-studio-onboard.html
[8] https://docs.aws.amazon.com/fr_fr/sagemaker/latest/dg/autopilot-automate-model-development-create-experiment.html
[9] https://docs.aws.amazon.com/fr_fr/sagemaker/latest/dg/autopilot-automate-model-development-notebook-output.html
[10] https://docs.aws.amazon.com/fr_fr/sagemaker/latest/dg/automatic-model-tuning.html
[11] https://aws.amazon.com/fr/tools/
[12] https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html#configuration

Olivier Cruchant
Solution Architect Spécialiste en Machine Learning (ML) chez AWS. Il travaille chez Amazon depuis près de 7 ans et est basé à Lyon en France. Olivier aide les clients d'AWS à développer et déployer des applications utilisant du ML, dans tous les secteurs, de l'agriculture à la finance.

Optimiser les modèles de Machine Learning pour l'IoT et le déploiement embarqué

Rapidité, disponibilité, confidentialité des données : nombreuses sont les raisons de déployer ses modèles de Machine Learning (ML) en embarqué, au plus près de leur point de consommation. Néanmoins, ce choix s'accompagne de contraintes : smartphones, caméras intelligentes et autres objets connectés ont des capacités de calcul limitées qui doivent être utilisées de manière frugale.

Inférence ML embarquée : objectifs et contraintes

Un modèle de ML est une transformation de données apprise sur des exemples réels par mimétisme (apprentissage supervisé), par interaction (apprentissage par renforcement) ou par recherche de structure sous-jacente (apprentissage non-supervisé). Par exemple, la classification d'image est un algorithme qui prend en entrée les pixels d'une image et apprend à produire une catégorisation. La détection est une transformation plus avancée qui, outre une catégorie, apprend également à localiser les catégories dans l'image. Autre exemple, la prévision de séries temporelles est un algorithme qui prend en entrée des occurrences passées d'une ou plusieurs séries de valeurs et essaye d'en prédire les futures valeurs.

Figures 1 et 2

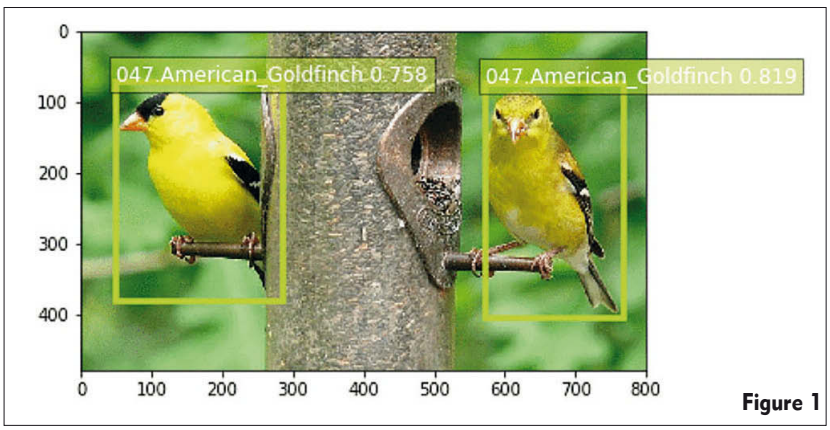


Figure 1

Exemple : Algorithme de détection (classification et localisation simultanées)
https://aws.amazon.com/blogs/machine-learning/identifying-bird-species-on-the-edge-using-the-amazon-sagemaker-built-in-object-detection-algorithm-and-aws-deeplens/

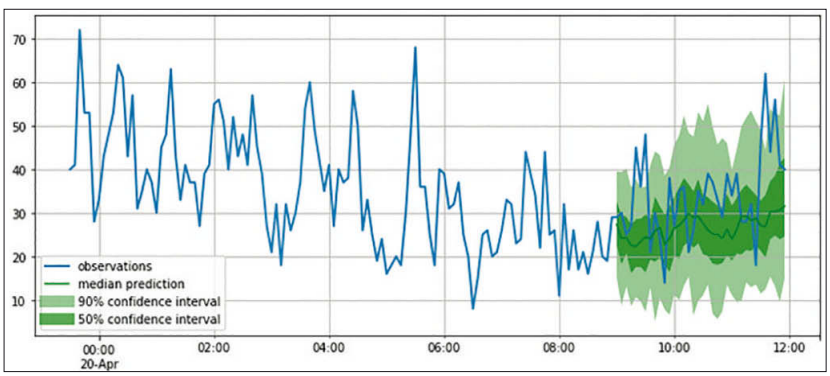


Figure 2

Exemple : Algorithme de prévision de série temporelle
https://aws.amazon.com/blogs/machine-learning/creating-neural-time-series-models-with-gluon-time-series/

Le ML est particulièrement utile pour apprendre des analyses qui seraient compliquées à coder à la main. Par exemple des analyses de données de grande dimensionnalité – comme le langage naturel, parfois constitué de plusieurs dizaines de milliers de mots différents - ou des analyses de données très abstraites, par exemple les milliers voire millions de pixels d'une image, dont la valeur individuelle contient peu d'information mais qui sont porteurs de sens lorsqu'on étudie leur géométrie collective. La phase de réglage de l'algorithme est appelée entraînement, et prend un temps variable selon la taille des données et la complexité du modèle – de quelques secondes à quelques jours. La richesse des dépendances logicielles ainsi que le fort besoin en calcul rendent le cloud AWS particulièrement propice à l'entraînement d'algorithmes de ML : les nombreux environnements prêts à l'emploi et le vaste choix d'infrastructures de calcul permettent de supporter les projets les plus exigeants. La phase durant laquelle un modèle entraîné est utilisé pour réaliser des prédictions ou transformations sur des nouvelles données est appelée *inférence*.

L'inférence peut être réalisée dans le cloud comme en embarqué. L'inférence embarquée sur objet connecté a plusieurs avantages : (1) les données brutes utilisées dans l'inférence ne quittent pas l'objet, (2) le délai d'obtention de prédiction pourra être plus faible (pas besoin de faire un aller-retour réseau vers le cloud) et (3) le modèle ML reste accessible en cas de perte de connectivité. Par exemple, un modèle de vision artificielle pourrait être utilisé dans une mangeoire intelligente qui délivre des doses personnalisées de nourriture aux animaux grâce à leur identification. Si cet objet est déployé dans des régions à faible connectivité, il n'y a d'autre choix que d'effectuer l'inférence ML à bord de l'objet. Autre exemple : un modèle de réglage et d'optimisation des cordages déployé sur un voilier de course. Le besoin de faible latence et de haute disponibilité rend l'inférence embarquée plus pertinente que l'inférence distante dans le cloud. Le déploiement embarqué s'accompagne néanmoins de complexités : à bord d'un objet connecté, la capacité de calcul est

moindre que dans le cloud et la gestion des dépendances logicielles et des mise-à-jour plus difficiles. Dans cet article nous présentons deux services AWS facilitant l'inférence embarquée : le compilateur de modèle SageMaker Neo et AWS IoT Greengrass. **Figure 3**

Optimiser et standardiser le modèle et ses dépendances via SageMaker Neo

Il existe de nombreux outils pour développer des modèles ML. Par exemple, XGBoost est une librairie open-source populaire pour modéliser les données tabulaires. Apache MXNet, TensorFlow et PyTorch sont des frameworks de deep learning (DL) permettant le développement de réseaux de neurones. Les réseaux de neurones sont une famille de modèles constitués d'un enchaînement de cellules élémentaires appelées neurones artificiels et réalisant une somme pondérée de leurs entrées, suivie d'une non-linéarité. En enchaînant un grand nombre de ces transformations, il est possible de d'apprendre des transformations sophistiquées capables de modéliser des données de grande dimension et à faible sémantique, telles qu'image, texte ou son. Dans la plupart des librairies de ML il est possible d'enregistrer le modèle après entraînement, sous la forme d'un fichier contenant son architecture et ses coefficients. Ce fichier est appelé l'artefact du modèle. Les librairies d'entraînement ML sont parfois très volumineuses : en effet, elles contiennent de nombreux modèles pré-écrits et parfois pré-entraînés, mais aussi de vastes collections d'opérateurs et de fonctions permettant d'écrire, orchestrer et optimiser l'entraînement ML.

Les besoins lors de l'inférence ML sont moindres : en théorie, il suffirait d'utiliser uniquement les quelques opérateurs mathématiques élémentaires nécessaires à la traversée du modèle. De plus certaines plateformes matérielles proposent des librairies d'algèbre optimisées qui permettent d'utiliser au mieux l'infrastructure, tel que la bibliothèque Math Kernel Library (MKL) d'Intel. La compilation d'un modèle de ML consiste à produire, à partir de l'artefact d'un modèle entraîné, un artefact de modèle optimisé ainsi qu'une application de prédiction minimale et spécialisée pour l'exécution d'inférence d'un modèle précis sur une plateforme matérielle donnée. SageMaker Neo est un service de compilation intégré facilitant cette optimisation. Etant donné un artefact de modèle, la librairie utilisée pour l'entraînement et une plateforme matérielle cible, SageMaker Neo produit une application de prédiction et un artefact de modèle optimisé. La compilation avec SageMaker Neo a plusieurs bénéfices : (1) l'optimisation de l'application de prédiction et de l'artefact du modèle permettent de réduire la latence et l'empreinte mémoire, tandis que (2) la production d'une application de prédiction permet de simplifier les dépendances nécessaires à l'inférence embarquée. SageMaker Neo supporte les artefacts provenant de TensorFlow, Keras, PyTorch, MXNet, XGBoost, ONNX, DarkNet, TFLite et peut compiler pour de nombreuses plateformes dont NVIDIA, Intel et ARM.

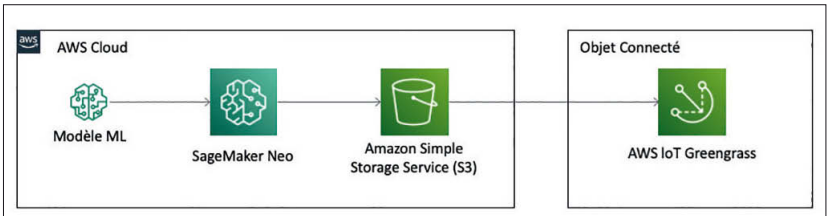


Schéma d'architecture simplifié

Figure 3

La documentation fournit les informations sur la compatibilité https://docs.aws.amazon.com/sagemaker/latest/dg/neo.html. L'exemple ci-dessous montre comment compiler un modèle de classification ResNet provenant de la librairie MXNet avec la *Command Line* AWS (CLI) pour instance cloud EC2 M5. Plus d'information sur l'architecture ResNet peuvent être lues dans l'article de recherche "Deep Residual Learning for Image Recognition » de Kaiming He et ses coauteurs, https://arxiv.org/abs/1512.03385. SageMaker Neo utilise le service de stockage objet Amazon S3 pour lire les modèles en entrée et écrire le résultat de la compilation.

Création d'un fichier de configuration Neo job.json. :

```
{
  "CompilationJobName": "resnet-mxnet",
  "RoleArn": $SAGEMAKER_ROLE_ARN,
  "InputConfig": {
    "S3Uri": "s3://<my S3 bucket>/model.tar.gz",
    "DataInputConfig": "{\"data\": [1, 3, 224, 224]}",
    "Framework": "MXNET"
  },
  "OutputConfig": {
    "S3OutputLocation": "s3:// <my S3 bucket>/output/",
    "TargetPlatform": {"Os": "LINUX", "Arch": "X86_64"},
    "CompilerOptions": "{\"mcpu\": \"skylake-avx512\"}"
  },
  "StoppingCondition": {
    "MaxRuntimeInSeconds": 300
  }
}
```

Détail des champs de configuration :

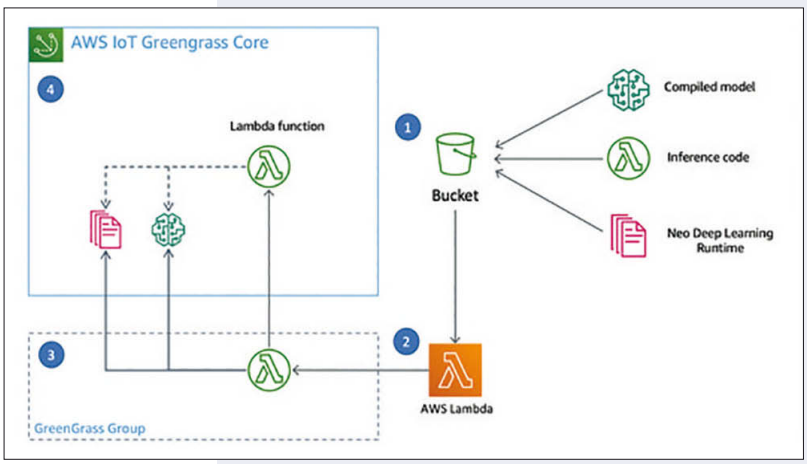
- CompilationJobName : nom de la tâche de compilation
- RoleArn : *Amazon Resource Name* (ARN) d'un role IAM (*Identity and Access Management*) donnant à Amazon SageMaker les permissions nécessaires pour conduire la tâche
- S3Uri : chemin Amazon S3 où l'artefact de modèle est stocké
- DataInputConfig : définit le format d'entrée attendu par le modèle
- Framework : framework de ML avec lequel le modèle a été entraîné
- S3OutputLocation : compartiment Amazon S3 où stocker le résultat de la compilation
- TargetPlatform: infrastructure cible, où le modèle compilé sera déployé. Alternative également possible : TargetDevice
- CompilerOptions : paramètres additionnels éventuels pour la compilation, spécifiques à une TargetPlatform donnée
- MaxRuntimeInSeconds : temps de compilation maximal en secondes

AUTOMATISER ET SÉCURISER LES OPÉRATIONS AVEC AWS GREENGRASS IOT

L'environnement d'inférence DLR étant open-source, il est tout à fait possible de gérer de manière autonome l'inférence embarquée sur l'environnement compatible de son choix. Cependant, pour une inférence intégrée et sécurisée, AWS fournit un service dédié à la gestion d'objets connectés : AWS Greengrass IoT. AWS Greengrass IoT permet notamment de déployer des modèles de ML pour inférence embarquée. La documentation indique les plateformes compatibles avec Greengrass : https://docs.aws.amazon.com/greengrass/latest/developerguide/what-is-gg.html#gg-platforms

AWS Greengrass IoT permet de configurer des Greengrass Group, paramétrage effectué dans le cloud pour régir un ou plusieurs objets connectés. Les objets connectés embarquent un client local appelé Greengrass Core, qui régit l'exécution de calcul local, la connectivité au cloud (éventuellement intermittente) et la connexion à des ressources locales (par exemple accélérateurs ou caméras).

Le déploiement d'un modèle ML sur un objet connecté géré par Greengrass se fait via son Greengrass Group et nécessite 3 composants : (1) une *ML resource* créée à partir d'un artefact de modèle déposé dans Amazon S3 – qui peut être un artefact de compilation Neo ; (2) du code d'inférence mis sous forme d'une fonction AWS Lambda et (3) les dépendances d'inférence, qui se limitent à la librairie DLR dans le cas d'un modèle compilé via Neo. AWS Lambda est un service de micro-machines virtuelles dans le cloud, fournissant des environnements de calcul éphémères limités à 10GB de RAM et 15min d'exécution. AWS IoT Greengrass permet de déployer des fonctions AWS Lambda en embarqué sur des objets connectés. La création de la fonction Lambda peut être faite soit via console, via SDK, ou par langage d'*infrastructure-as-code*. Le schéma ci-dessous résume les étapes nécessaires. Un tutoriel détaillé de déploiement Greengrass est disponible : https://github.com/aws-samples/aws-greengrass-ml-deployment-sample



Source complète : https://docs.aws.amazon.com/sagemaker/latest/API-Reference/API_OutputConfig.html

Lancement de la compilation

```
aws sagemaker create-compilation-job –-cli-input-json file://job.json
```

La compilation est également possible depuis d'autres SDK, par exemple avec boto3 en Python. De nombreux exemples sont disponibles sur GitHub https://github.com/aws/amazon-sagemaker-examples#amazon-sagemaker-neo-compilation-jobs. Les modèles compilés avec SageMaker Neo peuvent être utilisés de plusieurs manières : l'inférence cloud est possible, en utilisant par exemple les containers Docker Amazon SageMaker https://docs.aws.amazon.com/sagemaker/latest/dg/neo-deployment-hosting-services-container-images.html . L'environnement d'inférence est en open-source sous le nom DLR (Deep Learning Runtime, https://github.com/neo-ai/neo-ai-dlr), et peut donc aussi être utilisé hors du cloud, par exemple sur des objets connectés. Dans le prochain paragraphe nous expliquons comment l'utilisation du service AWS Greengrass IoT facilite la gestion d'inférence embarquée sur des objets connectés.

Ci-dessous un exemple d'inférence sur CPU réalisée avec l'API Python de la librairie open-source DLR. Une API C est également disponible. Au préalable, le modèle précédemment compilé a été téléchargé depuis S3 et décompressé dans le dossier 'modeldir'. Le guide d'installation de DLR se trouve ici https://neo-ai-dlr.readthedocs.io/en/latest/install.html#

```
import os
import numpy as np
from dlr import DLRModel

# Load the compiled model
model = DLRModel(model_path='modeldir', dev_type='cpu')

# Load an image in numpy format, with same shape as in compilation configuration
np_image = np.load('pic.npy')

# Predict
out = model.run(np_image)
```

SageMaker Neo supporte une grande variété de frameworks de machine learning. Un exemple exhaustif de compilation de TensorFlow Lite pour Raspberry Pi est disponible ici : https://github.com/neo-ai/neo-ai-dlr/blob/master/sagemaker-neo-notebooks/edge/raspberry-pi/neo-rasb3b-object-detection.ipynb

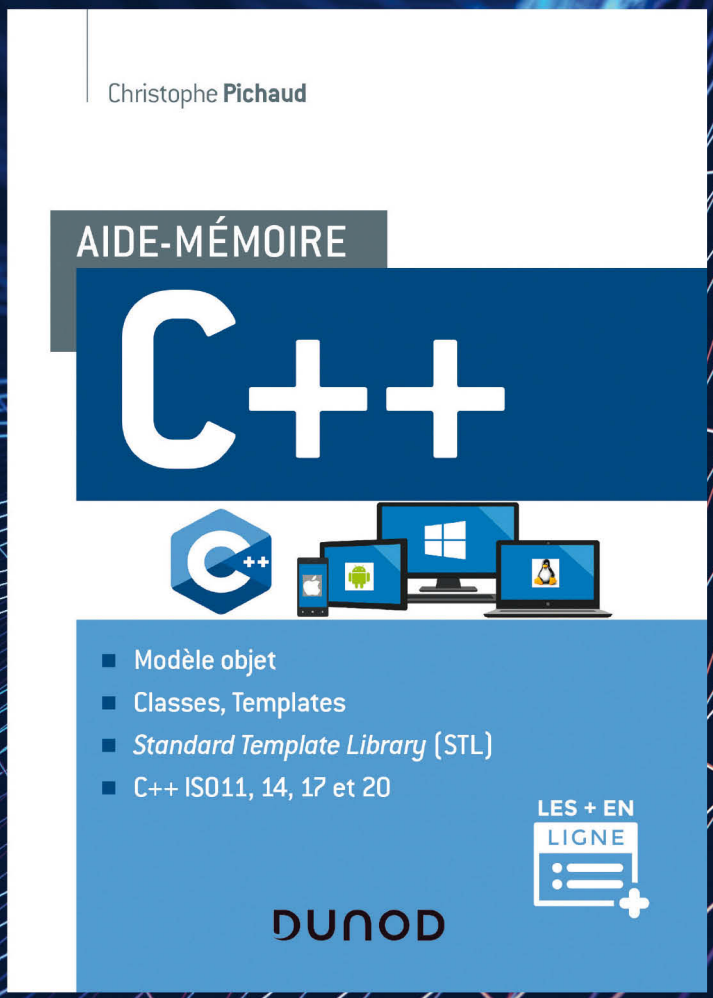
Conclusion

La compilation avec SageMaker Neo a de multiples bénéfices : elle améliore les performances d'inférence, tout en simplifiant la gestion des dépendances logicielles. De nombreuses études de cas et tutoriels sont accessibles via la documentation AWS et le *AWS Machine Learning Blog* : n'hésitez pas à les consulter et à tester ces fonctionnalités !

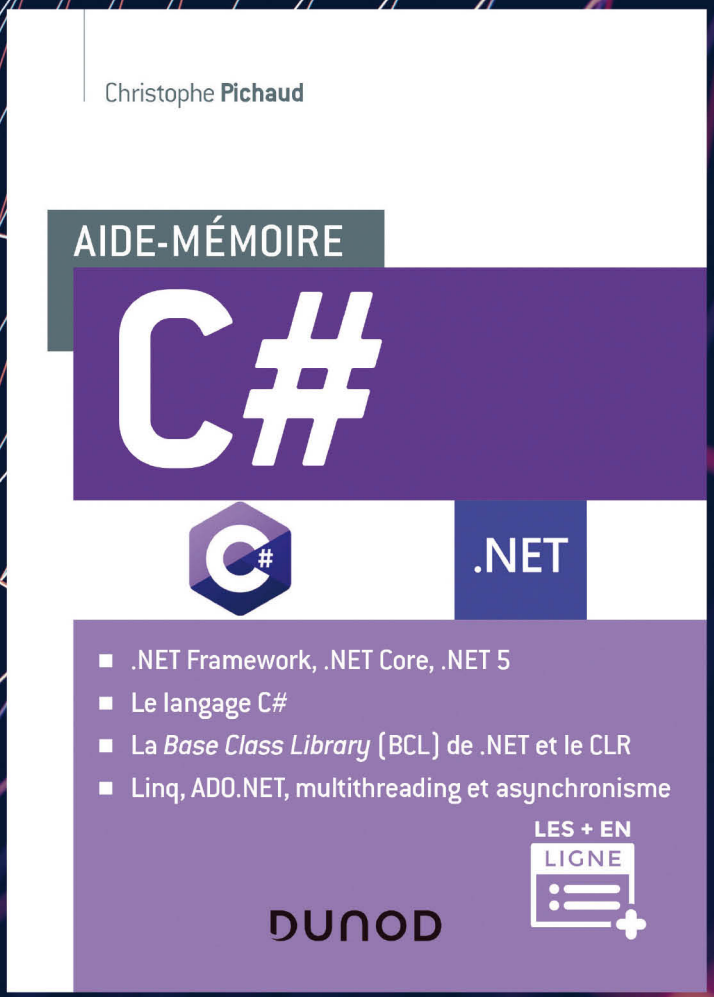
Quelques ressources

AWS ML Blog https://aws.amazon.com/blogs/machine-learning/
SageMaker Neo Documentation :

DÉVELOPPEURS, FORMEZ-VOUS À C++ ET C#



9782100807123 · 21,90€



9782100813223 · 23,90€

Pour les pros et les étudiants

DUNOD
une page d'avance

Bug ou PC ?

LES PROCHAINS NUMÉROS

Programmez! n°245

PHP 8.0 : overview & 1er bilan

Retour sur Drupal 9

Disponible dès le 5 mars

PROGRAMMEZ!

Programmez! hors-série n°3
hiver 2020-2021

Directeur de la publication & rédacteur en chef
François Tonic
ftonic@programmez.com

Secrétaire de rédaction
Olivier Pavie

Contacter la rédaction
redaction@programmez.com

Les contributeurs techniques

Sébastien Stormacq	**Ségolène Dessertine Panhard**
Julien Simon	**Bruno Medeiros de Barros**
Davide Gallitelli	**Othmane Hamzaoui**
Steve Houël	**Safian Hamiti**
Florent Brosse	**Olivier Cruchant**
Dorien Richard	**CommitStrip**

Couverture
Amazon Web Services

Maquette
Pierre Sandré

Marketing – promotion des ventes
Agence BOCONSEIL - Analyse Media Etude
Directeur : Otto BORSCHA
oborscha@boconseilame.fr
Responsable titre : Terry MATTARD
Téléphone : 09 67 32 09 34

Publicité
Nefer-IT
Tél. : 09 86 73 61 08
ftonic@programmez.com

Impression
SIB Imprimerie, France

Dépôt légal
A parution

Commission paritaire
1220K78366

ISSN
2279-5001

Abonnement
Abonnement (tarifs France) : 49 € pour 1 an, 79 € pour 2 ans. Etudiants : 39 €. Europe et Suisse : 55,82 € - Algérie, Maroc, Tunisie : 59,89 € - Canada : 68,36 € - Tom : 83,65 € - Dom : 66,82 €.
Autres pays : consultez les tarifs sur www.programmez.com.
Pour toute question sur l'abonnement :
abonnements@programmez.com

Abonnement PDF
monde entier : 39 € pour 1 an.
Accès aux archives : 19 €.

Nefer-IT
57 rue de Gisors, 95300 Pontoise France
redaction@programmez.com
Tél. : 09 86 73 61 08

tangente l'aventure mathématique

Le seul magazine au monde de culture mathématique.

Abonnez-vous, incitez vos amis à le faire !

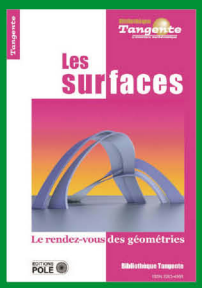
Tangente propose chaque année 10 numéros :

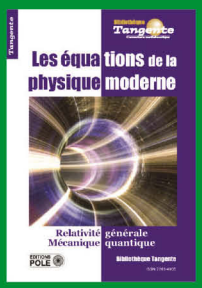
6 numéros « normaux » par an

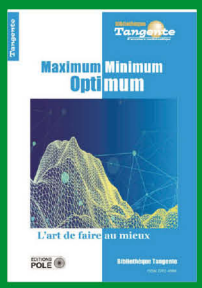
Version numérique sur tangente-mag.com

4 numéros hors série par an

Et aussi, avec l'abonnement SUPERPLUS, la plus belle bibliothèque mathématique au monde







4 livres hors séries Bibliothèque Tangente par an

Retrouvez les numéros de *Tangente* sur

https://www.tangente-mag.com

Commandes et abonnements sur

https://infinimath.com/librairie

Tangente, c'est aussi une version numérique exceptionnelle !

AVEC SA NOUVEAUTÉ : L'ABONNEMENT NUMÉRIQUE INTÉGRAL.

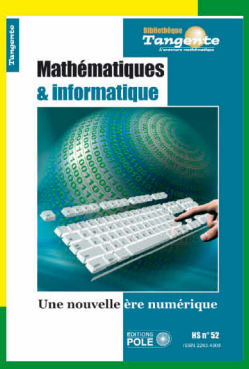
Accédez pour 5 € par mois à la consultation de :

- 28 numéros de *Tangente*
- 17 hors séries
- 53 nos de *Tangente Éducation*
- 400 problèmes du Monde

Et si vous êtes abonné(e) « papier », l'abonnement XXL (papier + num. intégral) ne vous coûte que 8 € de plus par an !

OFFRE SPÉCIALE PROGRAMMEZ !

Maths et informatique
Livre de 160 p. offert
si vous vous abonnez en ligne à *Tangente* Superplus avant le 31/03/21 en précisant dans le commentaire de la commande.
Offre Programmez ! Maths et informatique

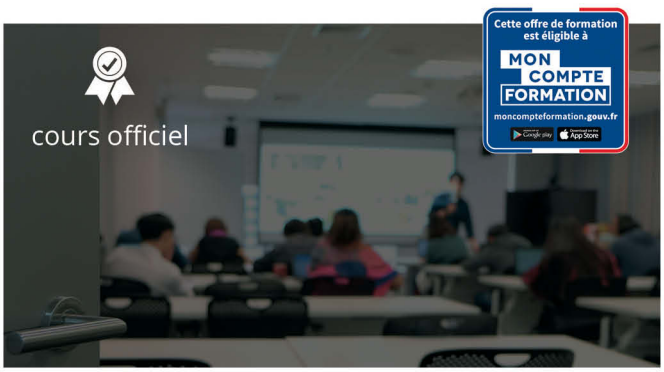


Leader de la formation
à l'informatique

ÉCOLE | FORMATION | ÉDITION

Vous avez un besoin en **FORMATION AWS ?** nous avons **LA SOLUTION !**

Je me forme **en live** avec un formateur



Amazon Web Services – Architecture

Cette formation dispense les fondamentaux de la création d'une infrastructure informatique sur la plateforme AWS. Elle apprend à optimiser le cloud AWS, avec les services AWS et la façon dont ils s'intègrent aux solutions cloud...

Prochaines sessions à distance ou en présentiel :
02/02/2021 - 17/03/2021 - 18/05/2021 - 30/06/2021

Inscrivez-vous ! 02 40 92 45 64

Je me forme **en autonomie**

Avec le livre

AWS
Gérez votre infrastructure sur la plateforme cloud d'Amazon

Par Nicolas DUMINIL

39€

Avec la vidéo

AWS
Développez votre première application web serverless

Par Arnaud JEAN

25€

Et retrouvez l'ensemble de nos formations AWS sur **www.eni-service.fr**

Et de nos livres et vidéos AWS sur **www.editions-eni.fr**

ECOLE IT
1200 diplômés
de BAC+2/+5 par an

FORMATION
+ 700 formations/e-formations
en bureautique et IT

LIVRE & VIDEO
Créateurs de contenu
en bureautique et IT