



*Ecole Nationale
Supérieure d'Electronique,
Informatique,
Télécommunications,
Mathématique et
Mécanique de Bordeaux*

RAPPORT DE STAGE

NOM : Juillard
PRENOM : Sandrine
FILLIERE : Télécommunication

DATE DU STAGE : du 29/06 au 31/08 , soit 9 semaines

DENOMINATION DE L'ORGANISME D'ACCUEIL : BLU AGETM

ADRESSE : 32 av Léonard de Vinci, 33600 PESSAC

PAYS : FRANCE

EXPERIENCE INTERNATIONALE : NON

CONVENTION DE STAGE : OUI

Responsable stage 2A : Astien Eric / eric.astien@bordeaux-inp.fr

Avant propos

Ce rapport fait office de compte-rendu de mon stage de 2eme année à Blu Age, dans le cadre de l'obtention de mon diplôme à l'ENSEIRB MATMECA. Ce stage de deux mois c'est dérouler d'une manière assez particulière, étant donnée les circonstances de la crise du COVID19 de 2020. Il c'est déroulé des manière hybride : en présentiel lorsque c'était nécessaire, et en télétravail la plus part du temps. Malgré le contexte exceptionnelle, ce stage à été, pour ma part, une très bonne expérience.

Remerciements

Merci à mon maître de stage, Alexis Henry pour m'avoir offert ce stage, et pour m'avoir accompagner tout au long de celui-ci.

Merci à Clement DHIVER de m'avoir accorder du temps pour répondre à mes questions lorsque j'en avais besoin.

Merci a tous les collaborateurs de BluAge pessac pour leurs acceuille.

Tables de Matière

I. Blu Age	p.1
1) Netfective Technology, l'organisation mère	p.1
2) Les activités de BluAge : Cobol, langage obsolète	p.1
3) Le centre BluAge Pessac	p.1
II. Contexte Technique	p.1
1) Deployment des produit BluAge	
2) Chaîne d'intégration continu	
3) Outils de développement et d'intégration	
4) Organisation du travail	
III. Création d'une pipeline	p.1
1) Les défauts de l'ancienne pipeline	
2) Nouveau modèle	
IV. Validation du modèle	p.1
1) Outils d'évaluations	
2) Paramétrage de la Pipeline	
V. Anecdotes	p.1
1) Difficulté technique rencontré	
2) Expérience du télétravail	
VI. Bilan et expérience professionnelle	p.1

Listes des Abréviations

AWS	Amazone Web Services
Cobol	Common Business Oriented Language
API	Application Programming Interface (interface de programmation applicative)
DevOps	Software Development (<i>Dev</i>) IT operations (<i>Ops</i>)
VM	Virtual Machin
IDE	Integrated development environment (Environnement de développement intergré)

Introduction

Fin 2018, à l'occasion de la conférence AWS Re:invent à Las Vegas, BluAge présente officiellement "[serverless COBOL for AWS solution](#)", ainsi qu'un compilateur, sous forme d'API, qui génère, en Java, une application développer en COBOL. Une annonce cohérant avec la volonté d'Amazone de banaliser l'utilisation du "Serverless", une offre Cloud émergeante où la gestion des ressources est manager par le fournisseur [1] [2]. Pour suivre cette logique, le compilateur mentionné plus tôt va être mise à disposition des clients à l'aide des micro environnement d'exécution d'amazone, appelé Lambda. Grâce à cette outils, les développeurs COBOL, pourrons configurer leurs IDE Visual Studio Code, et directement faire appelle à une Lambda, pour récupéré à la compilation leur application générée en Java. (Annexe 1)

Pourquoi la nécessiter de compiler en Java un code écrit en COBOL ? Car le COBOL est en langage que l'on pourrais qualifié d'obsolète, pourtant, il est encore très présent dans les entreprises. BluAge à pour but d'accompagner ces entreprises dans leurs transitions digital. La plus par de leurs solutions sont des outils pour assister les développeurs dans la traduction de leurs applications du COBOL vers le Java ou le .Net. Néanmoins, il est parfois difficile pour des développeur COBOL de se reconvertis vers le Java. Le compilateur proposer par BluAge, permet d'éviter les licenciement, sans restreindre l'entreprise dans leurs conversion digital en permettant au développeur de continuer à écrire en COBOL, tout en générant des applications dans un langage plus moderne, le Java. Ce compilateur est le 1e produit BluAge à traduire du code à 100%, sans nécessité aucune intervention humaine;

A l'heure actuelle, **les produits severless de BluAge sont mise à disposition du client de manière manuelle**. A partir de la console graphique de Amazone, chaque lambda est déployer manuellement dans les régions souhaiter et tester une à une. L'opération peut prendre une demi-journée, voir une journée complète pour chaque nouvelle version publiée. Il semble judicieux de réfléchir à l'automatiser du déploiement de ces produits.

Introduction

Objectif : Crée une pipeline

L'objectif va donc de crée une nouvelle pipeline pour remplacer celle déjà existante, en se concentrant tous particulièrement à résoudre ces précédent défauts.

La chaîne de deployment va devoir être contruite à partir de zeros, aucun outils ou méthodes n'est déinit. Plusieur solution de prototpe sont envisagable, utilisant des outils classique de DevOps tel que Jenkins, mais aussi des outils très spéciique à Amazone tel que AWS Step function.

Il eiste beacoup de chemains possible, la difficulter sera de choisir celui qui correspondra le mieux au besoin de BluAge. On veux par exemple que la pipeline soit la foix robuste aux erreurs, et flexible, mais aussi qu'elle ne laisse passer aucune informations sensible en claire, en d'autre termes qu'elle soit sécurisé.

Ce rapport présentera la solution définitive pour la pipeline, et mettra en lumière les choix et les compromis qui ont du etre fait.

Blu Age

Avant de pouvoir présenter les activités qui m'ont été conférés, il est important de mettre en place le contexte de ce stage. Cela passe par la présentation de la société dans laquelle j'ai évolué pendant 2 mois : Blue Age

Netfective Technology, l'organisation mère

La révolution numérique a bouleversé notre société et en particulier dans le monde des entreprises. Depuis l'essor des nouveaux outils et possibilités qu'offre le digital, de nouveau besoin, émergeant des entreprises, aussi bien privé que public, ont vu le jour. En particulier, les évolutions très rapides de ces technologies demandent de se renouveler constamment. C'est pour répondre à cette demande qu'a été créée Netfective Technology, la corporation mère de Blu Age. Crée en 2000 et dirigée par Christian Champagne, la firme organise son activité autour de l'accompagnement des grandes entreprises et organismes publics vers le digital. Malgré seulement vingt ans d'ancienneté, la firme compte déjà plus de 160 collaborateurs, dont 80% d'ingénieurs. Implanté dans 3 pays : La France, le Maroc et les États-Unis; On retrouve parmi ces clients et partenaires de grande entreprise tel que Amazon, BNP PARIBAS, Orange, Spora Steria, Accenture.. Mais aussi des administrations gouvernementales tel que L'Administration de la sécurité sociale des États-Unis, Le département du Travail et des Retraites britannique, ou encore La direction des finances publique française.

Une organisation jeune et flexible

Netfective mais aussi par extension Blu Age est une jeune entreprise et ce, non seulement du fait de l'ancienneté de la firme mais aussi par l'âge de ces collaborateurs : 32 ans en moyenne.

Et cette jeunesse se retrouve aussi dans les coutumes managériales de la firme. Dans les locaux de Blu Age, les employés s'organisent selon la méthode Agile : Le scrum master réunit tous les collaborateurs et met en place le plan d'action chaque semaine.

L'ambiance général au sein de l'organisation se veux également détendue : Le tutoiement est obligatoire, et le bien-être des employées est au cœur des préoccupations.

Les activités de Blu Age : Cobol, langage obsolète

Le Cobol est un langage informatique créé en 1959 pour la programmation d'applications de gestion. Malgré son ancienneté, ce langage est encore très répandu dans les entreprises à travers le monde. Dans une optique de modernisation et de mise à niveau des technologies, il est alors intéressant de réécrire ces applications dans des langages plus modernes tel que le Java et le .NET. Ainsi, dans l'optique de réduire le coût de la transition, Blu Age propose des outils capables d'effectuer automatiquement cette traduction.

Images

Année de Cr éation 2000	Dirigeant Christian Champagne
Age moyen 32 ans	Nombre de Collaborateurs 160
Pays implanté France (Pessac, Surenne) Maroc (Casablanca, Rabat) Etat-Unis (Dallas)	TurnOver 5% Chiffre d' affaire 17M€

Figure 1,Table informatives de l'ensemble des sociétés Blu Age



Figure 2, organigram de Netfective technologie et ces filiales

Blu Age

La réécriture d'application

Blu Age base ces techniques de transcription sur une approche MDA : Comme son nom l'indique, cette technique se base sur la récupération d'un modèle. La suite logiciel « Blu Age Classic », va automatiquement extraire la logique de l'application, et facilitera le travail du programmeur chargé de la réécrire [2]. Cette suite fournira en plus des outils de générations automatique de code des outils de visualisation du code et des outils d'analyse fig.[2].

Ce logiciel, et tous les autres logiciels de la suite Blu Age sont régulièrement mise à jour et déployer sur les serveur AWS.

Le centre Blu Age Pessac

- Le centre BluAge de pessac est un des 5 point d'implenation de l'entreprise. Elle compte dans son service une soixantaine d'employé séparé dans 3 domaine d'activité distincts :

Le partie R&D : Service dans lequelle s'integre mon stage. Le pôle R&D dirigé par Alexis Henry est charger du developpement des nouveaux produit BluAge.

La partie BluSight : Les outils de traduction que propose BluAge ne traduise pas une application à 100%. L'intervention d'un developpeur est nécessaire. Bien que les outils sont consus pour faciliter le travail, il peux arriver que des entreprises ne souhaitent pas donner cette tâche à un developpeur en interne. Le pôle BluSight, dirigé par Youssef Iraoui, est donc en charge d'utilisé les outils BluAge, afin de livré directement à l'entreprise qui en à fait la demande, un application traduite à 100%.

Le partie e-commerce : Les activité du pôle e-commerce sont excentré des deux autre pôle. Ce services dirigé par Sébastien PRADET n'a qu'un seul client, un seul projet : Intermarché. Son but est de maintenir et amélioré le site de e-commerce de la grande enseigne.

Images

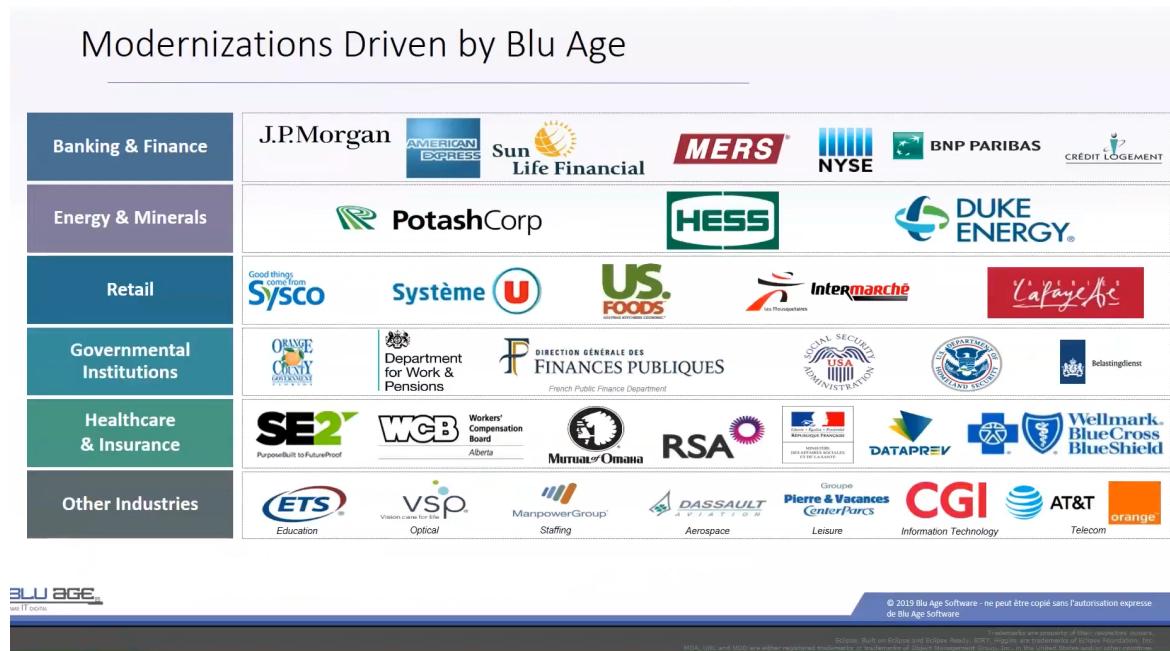


Figure 1,, Capture d'écran du logiciel BluAge Analyser

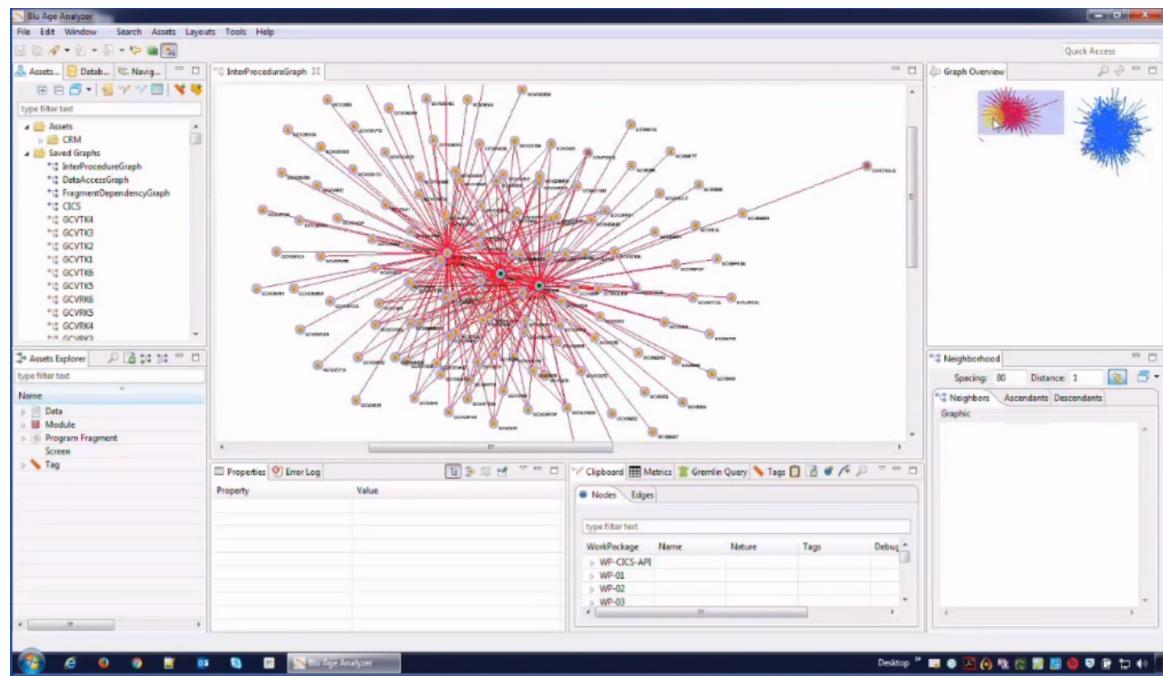


Figure 2, Capture d'écran du logiciel BluAge Analyser

Contexte technique

La réalisation d'une pipeline ne peut se réaliser sans outils adéquat. Le premier objectifs à mon arrivé chez Blu Age à été de me familiariser avec les différents logiciels que l'on m'a préconisé. Cette partie à pour but de vous les présenter brièvement, afin de rendre la partie qui suit plus accessible.

Deployment des produit bluAge

Pour mettre à disposition à disposition ces produits, bluAge utilise les serveurs d'AWS, et notamment Amazon Simple Storage Service, autrement appelé [Amazon S3](#).

Lorsque l'on veux déployer des données sur Amazon S3, on commence par créer un "[Bucket](#)". Ce bucket est une conteneur pour les données, autrement dit un répertoire. À l'intérieur, on place des objets. Un objet est l'ensemble données + metadata. Ce bucket pourra être déployé selon la/les régions souhaitées, et selon le type de stockage voulu, parmi les services proposés par Amazon (Services tarifés différemment, et conçus pour différents usages). Parmi les services proposés, deux nous intéresseront dans le cadre du stage :

Les nouvelles outils de déploiement : Cloud Serverless

Une architecture de cloud dites "serverless" désigne un cloud dans lequel le fournisseur de serveur gère dynamiquement les ressources allouées au service client. Ainsi, un client est débarrassé des tâches de gestion de serveur; AWS propose un service serverless basé sur la mise à disposition d'environnement 'stateless' (sans état) et éphémère : [Les lambda](#).

Une lambda est un environnement d'exécution ayant la particularité d'être "stateless". Cela signifie qu'il est l'inverse d'un environnement d'exécution classique, comme un conteneur ou une VM, il n'est pas dépendant de son historique d'exécution. Ce n'est pas dire pour autant qu'il est contraint à des opérations basiques : Toute la donnée dont la lambda a besoin pour effectuer une opération est stockée dans une base de données annexe.

L'avantage de ce type d'environnement est sa grande flexibilité, et sa résistance au panne : Tous d'abord, l'espace mémoire nécessaire à l'alloué, étant annexé à la lambda, est adapté au client. Mais surtout, si la lambda échoue, pour une raison quelconque, une autre lambda parmi celles disponibles pourra directement reprendre l'exécution, simplement grâce à la base de données allouée au client.

Du point de vue du client, la continuité de services est garantie, et la facturation du service est réduite strictement à ce qu'il a utilisé.

Images

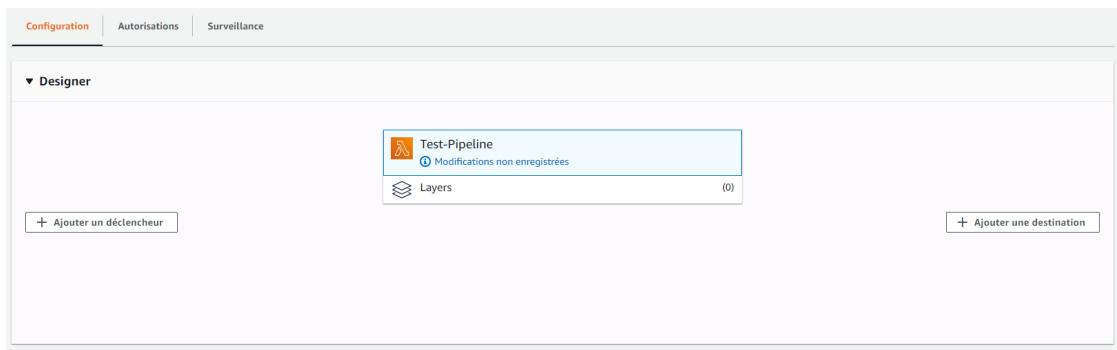


Figure 5, Console graphique d'AWS, Configuration d'une Lambda

Description de la Pipeline

Pour une meilleur compréhension de la pipeline, nous allons la subdivisé en 3 parties : La chaîne d'intégration, et la chaîne de diplôment et la chaîne ACL (Access Control List)

La chaîne de déploiemnt

Cette chaîne decrit les etapes nécessaire pour deployer un produit sur le cloud de AWS Serverless.

Etape 1 : Deploiment du produit sur s3 (sur une seule région)

Etape 2 : Transformation en Layer (sur une seule région)

Etape 3 : Deployer et tester une Lambda. (sur une seule région)

Etape 4 : Diffuser sur le reste de la liste des régions

A l'issue de la chaîne de déploement, le produit est bien stocker les serveurs AWS, mais il n'est pas encore à disposition des clients. A ce sae, seul le compte AWS qui à deployer le produit à access

Livrables

Nous avons vu dans la partie précédente, la chaîne de production qui doit être réalisé. Pour la concréteriser, une multitude d'options s'offre à nous. Une des tâches de ce stage a été de faire des choix, parmis tous les outils, et toutes les méthodes possibles celles qui seraient les plus adéquates. Pour cela, il est important de bien avoir en tête ce que on désir.

Une solution de pipeline devra respecter les points suivants :

- Doit être paramétrables, et ces paramètres doivent être accessibles sans avoir à modifier le code.
- Doit être sécurisé, les informations sensibles (clef d'accès etc) ne doivent pas être accessibles.
- Doit être peu coûteuse.
- Ne doit pas nécessiter d'intervention humaines, on cherche à construire une chaîne d'automatisation, elle doit s'adapter à tous les cas de figure possibles

Réalisation Technique

Solutions Envisagées :

Proposition #1 : Jenkins

La chaîne d'intégration est la partie en charge de la compilation de l'application. Toutes les opérations réalisées se vont de manière "local", sur les serveurs de BluAge.

Proposition #2 : Step Function

AWS Step Functions est un service web qui permet de coordonner les différents composants d'une chaîne d'opération ("Qu'on appelle généralement workflow ou flux d'opérations"). Cette outil permet de séparer simplement chaque opération et de pouvoir visualiser et suivre ce flux de manière graphique (cf figure).

Le langage pour programmer cette chaîne est le "Amazon States Language", basé sur le JSON. Il permet d'exécuter toutes les opérations sur les services AWS réalisables via la console AWS, ou par les SDK d'Amazon. Ce service permet de faciliter la transition entre les états mais ne réalise aucune exécution. En revanche, il permet de manager des environnements d'exécution AWS qui pourront être effectués du code, comme les Lambda.

Les avantages de cet outil sont sa facilité de compréhension et son suivi graphique qui le rend très simple à débugger lorsqu'il y a des erreurs. Amazon facture ce service par le nombre de transitions effectuées à 0,025 USD, soit 0,022 Euro pour 1 000 transitions d'état. Ajouter à cela, le prix d'exécution des Lambdas.

Proposition #3 : Pipeline exécuter à partir de Lambda

Il est possible de faire exécuter la pipeline uniquement par les Lambda. AWS fournit des SDK (bibliothèques) qui contiennent des fonctions permettant d'effectuer toutes les opérations sur tous les services proposés par Amazon. Parmi les langages de programmation disponibles, le Python a été choisi pour ce prototype, avec la bibliothèque développée par Amazon, Boto3.

Deux Lambda ont été utilisés pour cette pipeline de déploiement. (cf. fig) La première Lambda va être chargée de récupérer le fichier

Images

Stage View



Figure 7, Console graphique de Jenkins pour la visualisation des différentes étapes d'une pipeline

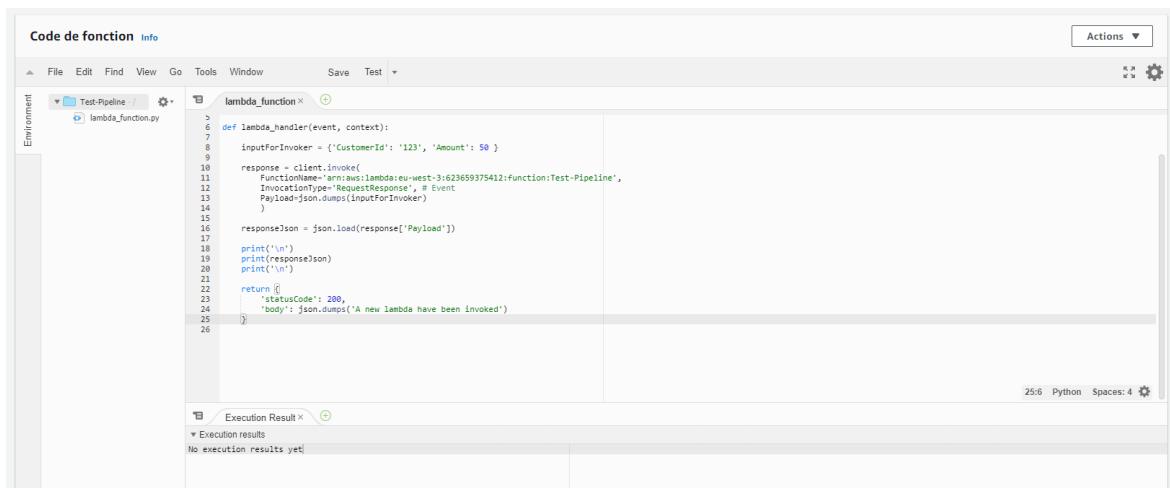


Figure 8, IDE intégré à AWS, pour programmer une Lambda en python

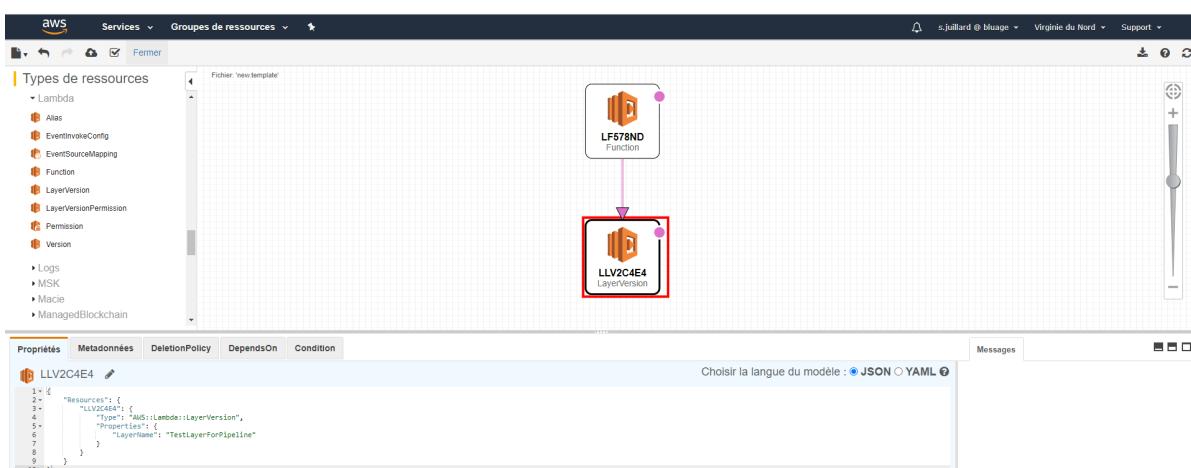


Figure 9, Interface graphique AWS pour créer des templates avec Cloud Formation

Solution retenue

La solution choisie est la proposition #3, celle qui code la pipeline en Python, avec la SDK d'Amazon et qui la fait exécuter sur des Lambda.

Pourquoi choisir cette solution ?

Le choix de cette solution est motivé d'une part, par sa flexibilité, mais également pour son coût très faible comparé à une autre solution proposée. Enfin, l'avantage de cette méthode par rapport à l'utilisation de Jenkins par exemple est que l'on utilise quand même des outils Amazon, et les opérations sont effectuées de manière interne au compte Amazon de l'entreprise. En d'autre terme, on élimine la nécessité de devoir se logger et donc de transmettre des identifiants ni en clair. Cette solution propose le meilleur compromis entre sécurité, flexibilité et coût.

Aperçu global du prototype

Lorsque les développeurs publient une nouvelle version, ils la pushent avec un outil de type Git. S'en suit alors un script Jenkins. C'est la chaîne d'intégration. Cette chaîne va compiler et tester le code. À l'issue de cette chaîne, la nouvelle version du framework va être chargé dans le Bucket Source (cf fig). Ce fichier, c'est celui indiqué sur le schéma sous le nom de "Layer File". C'est là que la chaîne de déploiement commence. L'action de déposer un fichier dans le bucket va alors déclencher le script de la Pipeline sur une Lambda. Cette fonction que l'on nomme "MainPipeline", va récupérer le fichier, publié la Layer, la tester, et ajouter les permissions sur le compte AWS qui doivent avoir accès. Toutes ces opérations sont effectuées dans la région par défaut (conventionnellement, en Virginie du Nord us-east-1). Puis avant de terminer, le script va déclencher de manière asynchrone (c'est à dire simultanément, comme des threads) N fonctions lambda "Deployer". Chacune avec comme argument une région différente. Leur rôle, effectuer le déploiement de la layer dans la région qui leur a été attribuée. Elles vont donc à partir du bucket source, créer une copie de ce bucket dans un bucket qui sera localisé dans leurs régions, pour pouvoir publier le framework sous forme d'une Layer dans leurs régions.

Optimisation des performances :

La chaîne présente à la partie précédente démarre en série (de manière synchrone) une d'autres Lambdas effectuant la fonction "Déployer". La fonction principale de la pipeline "ServerlessPipeline" va attendre qu'un déployer s'achève, soit qu'il récupère une réponse, pour démarrer le déployer suivant. Hors, AWS facture l'utilisation de la lambda selon son temps d'utilisation. Il semble alors judicieux de réfléchir à comment minimiser les temps d'attente. Pour évaluer les performances de cette méthode d'organisation de l'exécution des lambdas, nous allons observer le temps moyen d'exécution de la lambda en fonction du nombre de régions dans lesquelles on va la déployer. On cherchera à réduire au maximum cette durée afin de rendre notre solution plus économique.

Images

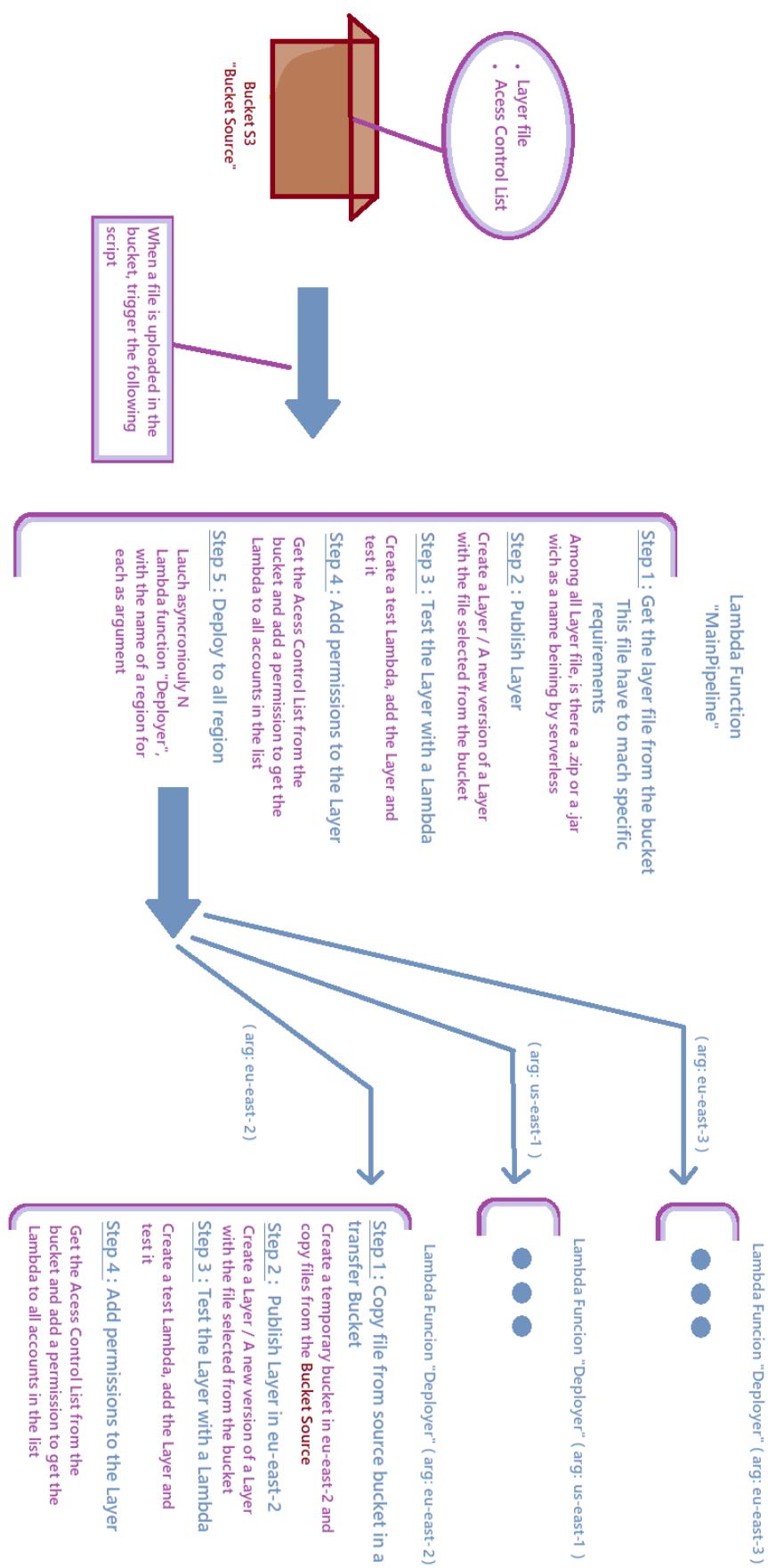


Figure 01, Schema détaillé du protocole suivit par la chaîne de déploiement.

Solution retenu

Optimisation des performances :

Performance des deux Lambda de la chaîne :

- Durée de ServerlessPipeline seule (deployment seulement sur la régions par défaut) : ~7s
- Durée du Deployer seule (temps de déploiement pour 1 régions) : ~14s

Nous allons construire un modèle afin de pouvoir comparer différente organisation dans le lancement des Lambda : durée Théorique de Serverless Pipeline : $7s + 14s \times nb_regions$

Vérification du modèle : Durée empirique de serverlessPipeline pour 2 région en plus de celle par default : $\sim 36s / 7 + 14 \times 2 = 35s$

Notre modèle n'est pas parfaitement exacte mais il sera satisfaisant pour notre approximation.

Soit est facturé durée de serverlessPipeline 35s (en comptant le temps d'attente) + durée des Deployer $14 \times 2 = 28s \implies$ durée facturé : 1min 3s

La première solution qui semble valable pour résoudre ce problème est de lancer les lambda

"Deployer" de manière asynchrone (soi en parallèle) comme des Thread.

Images

Test-Pipeline

Résultat de l'exécution: réussite (journaux)

Détails

La zone ci-dessous affiche le résultat renvoyé par l'exécution de votre fonction. [Découvrez-en davantage](#) sur le renvoi des résultats à partir de votre fonction.

```
{ "statusCode": 200, "body": "\"It works !\""} 
```

Récapitulatif

Code SHA-256	ID de demande
D3ONu5prNbYiu7WVT2LqQclryWnlbRtE9vkYko0bc=	e6a848d3-c995-44d0-a73c-8c220cdc58c8

Durée	Durée facturée
1304.18 ms	1400 ms

Ressources configurées	Mémoire max utilisée
128 MB	65 MB Init Duration: 251.19 ms

Sortie de journal

La section ci-dessous contient les appels de journalisation dans votre code. Ces derniers correspondent à une seule ligne dans le groupe de journaux CloudWatch correspondant à cette fonction Lambda. [Cliquez ici](#) pour afficher le groupe de journaux CloudWatch.

```
START RequestId: e6a848d3-c995-44d0-a73c-8c220cdc58c8 Version: $LATEST
END RequestId: e6a848d3-c995-44d0-a73c-8c220cdc58c8
REPORT RequestId: e6a848d3-c995-44d0-a73c-8c220cdc58c8 Duration: 1304.18 ms    Billed Duration: 1400 ms    Memory Size: 128 MB    Max Memory Used: 65 MB  Init Duration: 251.19 ms 
```

Configuration Autorisations Surveillance

Définitions

TurnOver

Taux de renouvellement du personnel d'une entreprise.

Métadata

Donnée servant à définir ou décrire un fichier. Dans le cas des « objects » que l'on met dans un bucket Amazon S3, on qualifie de métadata le chemin absolu de la donnée

Région

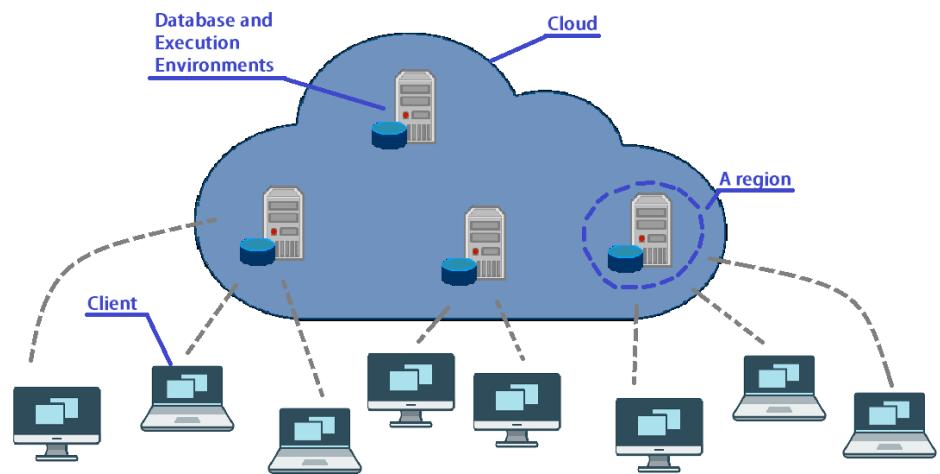
En Cloud computing, on appelle régions, une portions du Cloud, déterminer par l'emplacement physique des ressources qui hébergé le services. (Exemple de région AWS : us-east-3)

Cloud / Cloud Computing

Cloud storage is a model of computer data storage in which the digital data is stored in logical pools. The physical storage spans multiple servers (sometimes in multiple locations), and the physical environment is typically owned and managed by a hosting company. These cloud storage providers are responsible for keeping the data available and accessible, and the physical environment protected and running. People and organizations buy or lease storage capacity from the providers to store user, organization, or application data.

Cloud storage services may be accessed through a colocated cloud computing service, a web service application programming interface (API) or by applications that utilize the API, such as cloud desktop storage, a cloud storage gateway or Web-based content management systems.

Images



Conclusion

Références

- [1] « Ce qu'il faut retenir d'AWS re:Invent 2018», ITForBuisness, Laurent Delattre , 2018
Lien : <https://www.itforbusiness.fr/ce-qu-il-faut-retenir-d-aws-re-invent-2018-18837>
- [2] « Atout et limites du serverless computing », ITForBuisness, Laurent Delattre , 2018
Lien : <https://www.itforbusiness.fr/atouts-et-limites-du-serverless-computing-18219>
- [3] « Démo - Modernisation d'application avec Blu Age », BluAge, 2017
Lien : <https://www.youtube.com/channel/UCREdw7o0hKmqWFt1BjUmTuQ>
- [4] Serverless COBOL - Quickstart,, BluAge
Lien : <https://www.bluage.com/products/serverless-cobol-quickstart>
- [5] « A short story of serverless COBOL for AWS », BluAge, 2019
Lien : <https://github.com/BluAge/ServerlessCOBOLforAWS>
- [6] « A Streamlined Journey from Legacy to Microservices with Blu Age », BluAge, Avril 2019
Lien : <https://www.youtube.com/watch?v=jhB39NIgGl4&feature=youtu.be&t=2806>
- [7] « AWS Lambda Language Comparison : Pros and Cons », Yan Cui, Octobre 2018
Lien : <https://epsagon.com/development/aws-lambda-programming-language-comparison/>
- [8] Invocation Asynchrone et Invocation Syncrone , Documentation Amazon Web Services, 2020
Liens : <https://docs.aws.amazon.com/lambda/latest/dg/invocation-async.html>
<https://docs.aws.amazon.com/lambda/latest/dg/invocation-sync.html>
- [9] « Managing AWS Lamnda fucntion Concurrency», Chris Munns , Decembre 2017
Lien : [https://aws.amazon.com/fr/blogs/compute/managing-aws-lambda-function-concurrency/#:](https://aws.amazon.com/fr/blogs/compute/managing-aws-lambda-function-concurrency/#:~:text=AWS%20Lambda%20functions%20are%20designed%20to%20execute%20asynchronous%20code%20in%20a%20serverless%20environment%20and%20scale%20automatically%20based%20on%20the%20volume%20of%20events%20they%20receive.)

Annexes