assurly.

UVSQ université PARIS-SACLAY

UNIVERSITÉ PARIS SACLAY

MASTER DATASCALE

Rapport de stage de fin de cycle

Maître de stage :
Mr. Mickael BENHASSEN

Superviseur :
Dr. Yehia TAHER

Dieu Merci KIMPOLO NKOKOLO
dieu-merci.kimpolo-nkokolo@ens.uvsq.fr

29 août 2021

Declaration of Authorship

I, Mr. Mickael BENHASSEN, declare that this thesis titled, « Rapport de stage de fin de cycle » and the work presented in it are my own. I confirm that :

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed :

Date :

« Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism. »

Dave Barry

UNIVERSITÉ PARIS SACLAY

Résumé

Faculty Name
dieu-merci.kimpolo-nkokolo@ens.uvsq.fr

Doctor of Philosophy

Rapport de stage de fin de cycle

by Mr. Mickael BENHASSEN

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Acknowledgements

The acknowledgments and the people to thank go here, don't forget to include your project advisor…

Table des matières

Table des figures

Liste des tableaux

List of Abbreviations

LAH	**L**ist **A**bbreviations **H**ere
WSF	**W**hat (it) **S**tands **F**or

Physical Constants

Speed of Light $\quad c_0 = 2.997\,924\,58 \times 10^8\,\mathrm{m\,s^{-1}}$ (exact)

List of Symbols

a	distance	m
P	power	W ($\mathrm{J\,s^{-1}}$)
ω	angular frequency	rad

For/Dedicated to/To my...

Chapitre 1

Contexte du stage

Dans cette section je vais faire une présentation générale de Gigamesh, ses missions, et ses projets.

1.1 Présentation de Gigames

Fondée en 2017 par Toufik Gozim et Mickaeel Benhassen, Gigamesh est une InsurTech française visant à se lancer dans l'assurance emprunteur. Pour ce faire, l'entreprise travaille depuis sa creéation à obtenir l'agreément deélivré par l'ACPR (Autorité de Contrôle Prudentiel et de Résolution), nécessaire pour proposer une assurance. Cette institution est chargée de surveiller les banques et assurances en France.

Gigamesh devrait obtenir l'agrément en fin d'année 2021 et pourra donc officiellement lancer son produit d'assurance. Le choix de se positionner dans le domaine de l'assurance emprunteur n'est pas anodin. En effet, ce secteur est un oligopole détenu par des géants avec peu de place à la concurrence. Nous verrons par la suite sa complexité mais aussi l'intérêt qu'il suscite.

La loi Bourquin de février 2017, ouvre le marché de l'assurance emprunteur en permettant aux assurés de pouvoir changer leur contract d'assurance chaque année à la date anniversaire. De plus, le préavis pour effectuer ce changement est passé de 15 jours(Loi Hamon) à 2 mois. L'idée prend forme, puis le projet devient concret en octobre 2017 avec le création de la société GigaMesh SAS par Toufik Et Michael.
Gigamesh souhaite révolutionner le secteur de l'assurance emprunteur sur tous les fronts. Redonner du pouvoir d'achat, apporter une approche technologique et développer de nouvelles relations clients sont au cœur de leur vision.

1.2 L'organigramme de Gigames :

A compléter

1.3 Localisation de Gigames :

Localisé à Paris dans un espace de travail : **platform58**, **Gigamesh** profite d'un environnement de coworking et de bureaux privés. Un espace de coworking est un espace de travail partage mettant en avant l'échange et l'ouverture aux autres. Située au **58, rue de la Victoire 75009 Paris**, la **platform58** est un lieu d'innovation de la Banque Postale qui repose sur deux briques :

— Un programme d'accompagnement de startups
— Un lieu physique proposant la location d'espaces de travail

1.4 Domaine d'activité de Gigamesh

Nous allons voir dans cette partie le domaine dans lequel évolue **Gigamesh** et la réponse apportée pour le faire progresser.

1.4.1 Assurance emprunteur

L'assurance de prêt généralement désignée par assurance emprunteur (article L. 313-29 du code de la consommation) est une garantie demandée par les prêteurs (les banques) lors d'une demande de prêt. Bien que ce ne soit pas une obligation légale, elle est exigée dans la quasi-totalité des cas. Cette assurance permet de couvrir les risques de défaut de paiement quelles que soient leurs causes, ce qui explique qu'elle soit ainsi exigée. Elle comporte des garanties couvrant les risques :

— D'incapacité
— D'invalidité
— Voire de perte d'emploi

1.4.2 Assurly

Assurly est le nom donne au produit d'assurance de Gigamesh et represente tout le travail effectue par l'entreprise depuis sa création. Assurly est donc le résultat du projet de Gigamesh Assurly est une assurance emprunteur qui bouleverse le secteur de l'assurance en créant un produit clair, simple et au juste prix.

Tout est parti du constat que les assurances emprunteur traditionnelles ont construit leur analyse des risques sur une photographie figée de l'assuré, en décalage avec les nombreux changements que la vie réserve. Ils ont donc perdu de vue leur mission de départ : protéger en cas de problème.

Suite à 3 ans de recherche et développement, Gigamesh a réussi à associer les apports de l'innovation technologique et leur expertise assurantielle pour créer Assurly.

Assurly est née avec l'ambition de redonner le pouvoir aux assurés en proposant un produit clair avec des garanties all-inclusive au meilleur tarif. Assurly transforme l'assurance emprunteur grâce à une expérience client simplifiée et 100% digitale : une souscription en moins de 10 minutes depuis son portable avec zéro paperasse, zéro rendez-vous, zéro stress ! Assurly propose de couvrir ces risques avec les garanties suivantes :

1. La garantie perte totale et irréversible d'autonomie (PTIA).

   Elle intervient lorsque l'assuré se trouve dans un état particulièrement grave, nécessitant le recours permanent à une tierce personne pour exercer les actes ordinaires de la vie.

   La couverture Assurly : Gigamesh sera là pour vous épauler et rembourser 100% des mensualités du reste de votre prêt. La garantie PTIA cesse au 71ème anniversaire de l'assuré.

2. La garantie incapacité temporaire totale (ITT) dénommée incapacité temporaire totale dans le contrat.

   Elle intervient lorsque la personne assurée est temporairement inapte à exercer son activité professionnelle ou, si il n'en a pas, d'observer un repos complet l'obligeant

à interrompre toutes ses occupations de la vie quotidienne

La couverture Assurly : Gigamesh seralà pour vous épauler et payer 100% vos mensualités de remboursement de prêt durant votre temps d'impossibilité de travailler avec un plafond de 7500 euros par mois, quelle que soit votre perte de revenu. La garantie ITT cesse au plus tard au jour du 65ème anniversaire de l'assuré. Les affections dorsales, psychiques et psychiatriques causant l'ITT sont couvertes sans condition d'hospitalisation ou d'intervention chirurgicale. spacing

label La garantie Invalidité prend deux formes :

(a) La garantie invalidité permanente partielle (IPP) :
Elle intervient lorsque la personne assurée est, de façon définitive, incapable d'exercer strictement son activité après la reconnaissance de l'état d'invalidité estimé avec un taux entre 33% et 66%.

La couverture Assurly : Gigamesh sera là pour vous épauler et payer 50% de l'indemnité garantie en cas d'ITT, quelle que soit votre perte de revenu, avec un plafond de 7500€ par mois. Le délai de franchise maximale est de 90 jours après l'interruption de l'activité. Les affections dorsales, psychiques et psychiatriques causant l'IPP sont couvertes sans condition d'hospitalisation ou d'intervention chirurgicale

(b) La garantie Invalidité Permanente Totale (I.P.T.)
Elle intervient lorsque la personne assurée est, de façon définitive, incapable d'exercer strictement son activité professionnelle après la reconnaissance de l'état d'invalidité estimé avec un taux supérieur à 65%

La couverture Assurly : Gigamesh sera là pour vous épauler et rembourser 100% des mensualités du reste de votre prêt avec un plafond de 3 000 000 euros, quelle que soit votre perte de revenu. La garantie invalidité cesse au jour du 65ème anniversaire de l'assuré.

3. La garantie décès
Elle intervient en cas de décès de la personne assurée. Nous serons là pour épauler votre famille et payer 100% de vos mensualités de remboursement de votre prêt.

1.5 Objectif du stage

Cette partie fera l'objet de la présentation du projet sur lequel j'avais travaillé et de mes propres mission dans ce celui ci : Assurly est une plateforme digitale et innovante, qui simplifie le système d'assurance d'emprunt. Elle est constituée des modules suivante :

— Une application mobile B2C, pour la simulation de l'assurance et l'espace client,
— Une application B2B, pour la simulation de l'assurance et de la création d'un assure par des partenaires,
— Un ensemble d'application de gestion, paiement en ligne et service client,
— Une application backend pour la gestion et données et la fourniture des APIs

Les applications frontend communiquent avec le backend via une API de type Rest. Dans le cadre de mon stage j'intervenais plus souvent dans le backend et sur les missions suivantes :

- Le déploiement et l'intégration d'un service web (Reflex) dans un environnement AWS

Intégration des données en format txt dans une base de données sql dans l'environnement AWS

- Etude de faisabilité sur le déploiement d'une application Front-End dans un environnement AWS
- Sécurisation d'une API Rest développée avec la technologie Serveless d'AWS (Lambada + python).
- Le déploiement d'un système de web scraping dans un environnement AWS.

Chapitre 2

Etat de l'art

Dans cette partie nous présenterons le positionnement du travail demandé par rapport à l'état de l'art scientifique ou technique.

2.1 Les APIs(Application programming interface)

une API est une façade clairement délimitée par laquelle une application offre des services à d'autres applications. Cette façade peut comporter des classes, des méthodes ou des fonctions, des types de données et des constantes. Les API peuvent être publiques, partenaires ou privées

2.1.1 Objectif d'une API

Tout comme une interface utilisateur graphique facilite l'utilisation des programmes pour les profanes, une API a pour objectif de fournir une porte d'accès à une ou plusieurs fonctionnalités d'un système en exposant uniquement les objets ou les actions dont le développeur a besoin et en cachant les détails de la mise en œuvre. Elles permettent ainsi aux développeurs d'utiliser plus facilement certaines technologies pour créer des applications. Une API simplifie la programmation.

2.1.2 Analogie d'une API avec la vie courante

Prenons l'exemple d'un appel téléphonique, la plupart des gens ne savent pas comment se déroule un appel téléphonique : le mécanisme électronique qui régit un appel téléphonique. Pourtant tout le monde peut effectuer un appel juste en composant le numéro de son correspondant et en appuyant simplement sur un bouton (envoyer ou ok) du clavier. Le clavier et l'écran du téléphone sont l'équivalent de l'API ; ils représentent l'interface du système électronique qui régit un appel téléphonique

2.2 Le modèle ou l'architecture Rest

Les APIs peuvent être intégrées suivant plusieurs modèles ou styles d'architecture : le modèle à base des simples fonctions, le modèle RPC, le modèle SOAP, le modèle Rest puis le modèle GraphQL. Le modèle Rest, est un style orienté ressources et utilisant principalement le protocole HTTP pour la communication. Avec le modèle les données entre le client et le fournisseur sont transmises en format JSON ou XML. Ainsi, Rest utilise des notions qu'il faut obligatoirement comprendre : les ressources, les identifiants, les méthodes HTTP, et les formats de données.

2.2.1 Les principes de l'architecture REST

L'architecture Rest est régit par six principes suivants :

1. La séparation entre client et serveur
   les responsabilités du côté serveur et du côté client sont séparées, si bien que chaque côté peut être implémenté indépendamment de l'autre. Le code côté serveur (l'API) et celui côté client peuvent chacun être modifiés sans affecter l'autre, tant que tous deux continuent de communiquer dans le même format. Dans une architecture REST, différents clients envoient des requêtes sur les mêmes endpoints, effectuent les mêmes actions et obtiennent les mêmes réponses.
2. . L'absence d'état de sessions (stateless)
   la communication entre client et serveur ne conserve pas l'état des sessions d'une requête à l'autre. Autrement dit, l'état d'une session est inclus dans chaque requête, ce qui signifie que ni le client ni le serveur n'a besoin de connaître l'état de l'autre pour communiquer. Chaque requête est complète et se suffit à elle-même : pas besoin de maintenir une connexion continue entre client et serveur, ce qui implique une plus grande tolérance à l'échec. De plus, cela permet aux APIs REST de répondre aux requêtes de plusieurs clients différents sans saturer les ports du serveur. L'exception à cette règle est l'authentification, pour que le client n'ait pas à préciser ses informations d'authentification à chaque requête.
3. L'uniformité de l'interface
   les différentes actions et/ou ressources disponibles avec leurs endpointset leurs paramètres spécifiques doivent être décidés et respectés religieusement, de façon uniforme par le client et le serveur. Chaque réponse doit contenir suffisamment d'informations pour être interprétée sans que le client n'ait besoin d'autres informations au préalable. Les réponses ne doivent pas être trop longues et doivent contenir, si nécessaire, des liens vers d'autres endpoints.
4. La mise en cache
   les réponses peuvent être mises en cache pour éviter de surcharger inutilement le serveur. La mise en cache doit être bien gérée : l'API REST doit préciser si telle ou telle réponse peut être mise en cache et pour combien de temps pour éviter que le client ne reçoive des informations obsolètes.
5. L'architecture en couches
   un client connecté à une API REST ne peut en général pas distinguer s'il est en communication avec le serveur final ou un serveur intermédiaire. Une architecture REST permet par exemple de recevoir les requêtes sur un serveur A, de stocker ses données sur un serveur B et de gérer les authentifications sur un serveur C.
6. Le code à la demande Cette contrainte est optionnelle. Elle signifie qu'une API peut retourner du code exécutable au lieu d'une réponse en **JSON** ou en XML par exemple. Cela signifie qu'une API **RESTful** peut étendre le code du client tout en lui simplifiant la vie en lui fournissant du code exécutable tel qu'un script **JavaScript**.

Une API REST ne peut être qualifiée de **RESTful** si elle ne respecte pas les six contraintes, mais on peut tout de même la qualifier d'API REST si elle n'enfreint que deux ou trois principes. REST est sans doute le standard le plus utilisé pour concevoir des architectures d'API, mais il en existe bien d'autres qui pourraient le complémenter, voire un jour le détrôner

2.3 L'architecture serverless

C'est une architecture où l'utilisateur n'a pas à gérer la moindre infrastructure. L'architecture serverless ne signifie pas pour autant qu'il n'y a pas de serveurs : cela veut dire qu'ils sont invisibles pour l'utilisateur et sont gérés par les fournisseurs et non par les consommateurs. Sans trop penser à leur maintenance, les ressources informatiques sont utilisées comme des services.

L'architecture serverless fait changer la façon dont on conçoit et maintient les applications. Elle permet aux consommateurs de bâtir des plateformes en utilisant exclusivement des services managés, de se concentrer sur les aspects liés à la logique business, et non sur les contraintes de déploiement ou de scalabilité.

- Exemples fournisseur des architectures Serverless
  AWS
  —* Amazon S3
  Amazon Dynamodb
  * Amazon Lambda
  — Google :
  * Google Cloud Functions
  — Microsoft
  * Azure Functions

2.4 Le protocole OAuth2 et la délégation d'autorisation

Le Oauth2 est la seconde version du protocole Oauth(Open Autorization). Connu comme protocole de délégation d'autorisations, le Oauth2 est un protocole qui permet à une application d'obtenir une autorisation d'accès limitée aux ressources disponibles sur un serveur accessible via HTTP. Si ces ressources n'appartiennent pas à l'application, l'autorisation d'accès lui est déléguée par le détenteur des ressources ; au cas contraire l'application obtient l'autorisation en s'authentifiant avec ses identifiants. Avec le protocole Oauth2 le propriétaire de ressources ne partage pas ses identifiants avec l'application qui sollicite ses ressources. Le protocole Oauth2 fournit un modèle dans lequel le détenteur de ressource peut accorder un accès limité à ses données en émettant simplement un jeton temporaire, qui sera utilisé par cette application sollicitant ses données pour s'identifier auprès du serveur de ressources. Le protocole Oauth2 met le propriétaire de ressources au cœur du système d'octroi d'autorisation. C'est le propriétaire de ressources qui fait le lien entre ses comptes sur différentes applications sans que des administrateurs de la sécurité aient besoin d'intervenir directement sur chaque application.

2.4.1 Exemple d'implémentation du protocole Oauth2

Le protocole Oauth2 est utilisé par plusieurs entreprises comme Facebook, Google et Twitter. L'exemple que nous proposons ici est celui qui permet d'afficher instantanément les tweets sur Facebook sans avoir besoin de votre mot de passe Facebook

2.4.2 Le vocabulaire du protocole Oauth2

Les rôles : Le protocole Oauth2 identifie quatre rôles : Client, Propriétaire de ressources, Serveur d'autorisation et Serveur de ressources.

- Le détenteur des ressources(Resource Owner) :
  Le détenteur ou le propriétaire de ressources est une entité capable d'accorder l'accès à une ressource protégée. Lorsque le propriétaire de la ressource est une personne, on parle d' utilisateur final.
  Le serveur de ressources (Resource Server) :
  Le serveur de ressources est le serveur qui héberge les ressources protégées ou à accès limité. Un exemple du serveur des ressources peut être Facebook et Google qui hébergent les informations des profils des utilisateurs.
- Le client (Client Application) :
  C'est une application qui sollicite les ressources du serveur de ressources, celle-ci peut être une application PHP, une application mobile, une application Javascrip.
- Le serveur d'autorisation (Authorization Server) :
  Le serveur d'autorisation représente le serveur qui délivre des jetons au client. Ces jetons seront utilisés lors des requêtes du client vers le serveur de ressources. Ce serveur peut être le même que le serveur de ressources (physiquement et applicativement), et c'est souvent le cas.

Les jetons : Un jeton est une chaîne des caractères, générée par le serveur d'autorisation au client une fois qu' une autorisation lui est offerte.

- Le jeton d'accès(Access token) :
  est un jeton nécessaire pour accéder aux ressources partagées par OAuth2. Il a une durée de vie qui est généralement assez courte.
- Le jeton de renouvellement (Refresh Token) :
  est un jeton que le serveur d'autorisation peut émettre aux clients et il peut être échangé contre un nouveau jeton d'accès, sans répéter le processus d'autorisation. Il n'a pas de temps d'expiration.

Les types d'autorisations(Grant types)

- L'autorisation implicite (Implicit Grant) :
  Elle doit être utilisée quand l'application se trouve côté client(typiquement une application Javascript). Il ne permet pas d'obtenir de token de renouvellement.

  L'autorisation via un code (Authorization Code Grant) :
  Ce type d'autorisation permet d'obtenir deux jetons : le jeton d'accès et le jeton de renouvellement. Le client dans ce cas de figure interagit avec le propriétaire des ressources via un client web généralement un navigateur
- L'autorisation serveur à serveur (Client Credentials Grant) :
  Elle doit être utilisée lorsque le client est lui-même le détenteur des données. Il n'y a pas d'autorisation à obtenir de la part de l'utilisateur
- L'autorisation via mot de passe (Resource Owner Password Credentials Grant) :
  Avec ce type d'autorisation, les identifiants sont envoyés au client et ensuite au serveur d'autorisation. Il est donc impératif qu'il y ait une confiance absolue entre ces 2 entités. Ce type d'autorisation est principalement utilisé lorsque le client a été développé par la même autorité que celle fournissant le serveur d'autorisation final.

2.5 Intégration de données

L'intégration des données est le processus qui consiste à combiner des données provenant de différentes sources dans une vue unifiée : de l'importation au nettoyage en passant par le mapping et la transformation dans un gisement cible, pour finalement rendre les données plus exploitables et plus utiles pour les utilisateurs qui les consultent.

2.5.1 Les avantages

- L'intégration des données améliore l'unification des systèmes et la collaboration globale

  L'intégration des données fait gagner du temps
- L'intégration des données réduit les erreurs (et les besoins de modifications)

2.5.2 Opérations ETL et intégration des données

Les opérations d'extraction, de transformation et de chargement (ETL) forment un processus d'intégration à part entière dans lequel les données sont extraites du système source et livrées au data warehouse. Il s'agit d'un processus continu que le data warehousing exécute pour transformer plusieurs sources de données en informations cohérentes et utiles destinées à la Business Intelligence et aux analyses.

2.6 Containerisation

Il s'agit d'un type de virtualisation utilisé au niveau des applications. Le principe repose sur la création de plusieurs espaces utilisateurs isolés les uns des autres sur un noyau commun. On utilise alors le terme de « conteneur » pour désigner une telle instance. Cette séparation repose sur un concept similaire à celui des modules applicatifs cloisonnés, communiquant à l'aide de services et applications web. Les conteneurs, bien qu' indépendants, partagent un noyau commun (donc un ou plusieurs systèmes d'exploitation) et un même espace mémoire. La conteneurisation permet de packager tous les services, scripts, API, librairies dont une application a besoin. L'objectif : en permettre l'exécution sur n'importe quel noyau compatible

2.6.1 Avantages

- Elle évite de se soucier d'interactions ou d'incompatibilités avec les conteneurs déjà présents ou à venir sur cette machine.
- Elle permet de ne pas occuper autant de ressources que réclamerait une machine virtuelle (ou virtual machine, VM), qui emporte son propre système d'exploitation et bloque des ressources à son lancement.

2.6.2 Exemple des systèmes de conteneurisation

- Docker

  Kubernetes
- CoreOs rkt
- OpenV

Chapitre 3

Outils et technologies utilisés

Dans cette section je vais présenter les outils et les technologies que j'avais utilisé durant mon stage.

3.1 Amazon Lambda

AWS Lambda est un service de calcul Serverless qui vous permet d'exécuter du code sans :

- provisionner ou gérer des serveurs,
  créer une logique de dimensionnement de cluster prenant en charge la charge de travail,
- maintenir les intégrations d'événements ou gérer les environnements d'exécution.

Avec Lambda, vous pouvez exécuter du code pour pratiquement n'importe quel type d'application ou service backend , sans aucune tâche administrative. Il suffit de télécharger votre code sous forme de fichier ZIP ou d'image de conteneur, et Lambda alloue automatiquement et précisément la puissance d'exécution de calcul et exécute votre code en fonction de la demande ou de l'événement entrant, pour n'importe quelle échelle de trafic. Vous pouvez configurer votre code de sorte qu'il se déclenche automatiquement depuis plus de 200 applications SaaS et services AWS, ou l'appeler directement à partir de n'importe quelle application web ou mobile. Vous pouvez écrire des fonctions Lambda dans votre langage préféré (Node.js, Python, Go, Java, etc.)

3.1.1 Avantages

- Aucun serveur à gérer : AWS Lambda exécute automatiquement votre code, sans que vous ayez à mettre en service ou à gérer des serveurs

  Dimensionnement continu : AWS Lambda dimensionne automatiquement votre application en exécutant le code en réponse à chaque déclencheur. Votre code s'exécute en parallèle et traite chaque déclencheur indépendamment. La charge de travail est ainsi mise à l'échelle de façon précis
- Optimisation des coûts grâce au comptage en millisecondes : Avec AWS Lambda, vous ne payez que pour le temps de calcul que vous consommez
- Performances constantes à n'importe quelle échelle : Avec AWS Lambda, vous pouvez optimiser le temps d'exécution de votre code en choisissant la bonne taille de mémoire pour votre fonction

3.2 Docker

Docker est le système de containerisation le plus utilisé. Il permet d'embarquer une application dans un ou plusieurs containers logiciels qui pourra s'exécuter sur n'importe quel serveur machine, qu'il soit physique ou virtuel. Docker fonctionne sous Linux comme Windows Server. C'est une technologie qui a pour but de faciliter les déploiements d'application, et la gestion du dimensionnement de l'infrastructure sous-jacente. Docker possède deux type de fichier Dockerfile et docker-compose.yml :

- Dockerfile contient des instructions permettant de créer un container docker

  Docker-compose contient des instructions qui permettent de composer des containers

3.3 LAmplify

AWS Amplify est un ensemble d'outils et de services qui peuvent être utilisés ensemble ou un par un, pour aider les développeurs web mobile et frontal à créer des applications évolutives et intégrales à technologie AWS. Avec Amplify, vous pouvez configurer les backends d'application et connecter votre application en quelques minutes, déployer des applications Web statiques en quelques clics et facilement gérer le contenu des applications en dehors de la console AWS. Amplify prend en charge les frameworks Web populaires, tels que JavaScript, React, Angular, Vue, Next.js, et les plateformes mobiles, telles qu'Android, iOS, React Native, Ionic, Flutter.

3.3.1 Fonctionnements

Developpement

A compléter

Hebergement

A compléter

3.4 Amazon Simple Storage Service (Amazon S3)

S3 est un service de stockage d'objet offrant une évolutivité, une disponibilité des données, une sécurité et des performances de pointe. Les clients de toutes tailles et de tous secteurs peuvent ainsi utiliser ce service afin de stocker et protéger n'importe quelle quantité de données pour un large éventail de cas d'utilisation comme des lacs de données, des sites web, des applications mobiles, la sauvegarde et la restauration, l'archivage, des applications d'entreprise, des appareils IoT et des analyses du Big Data. Amazon S3 fournit des fonctions de gestion faciles à utiliser pour vous permettre d'organiser vos données et de configurer des contrôles d'accès affinés pour vos exigences métier, d'organisation et de conformité spécifiques. Amazon S3 est conçu pour offrir 99,999999999 % de durabilité et stocker les données de millions d'applications pour des entreprises du monde entier

3.5 Aws Aurora

Amazon Aurora est un moteur de base de données relationnelle qui associe la vitesse et la fiabilité des bases de données commerciales haut de gamme à la simplicité et la rentabilité des bases de données open source. Amazon Aurora MySQL offre des performances jusqu'à cinq fois supérieures à celles de MySQL sans nécessiter de modifications de la plupart des applications MySQL. De la même manière, Amazon Aurora PostgreSQL offre des performances jusqu'à trois fois supérieures à celles de PostgreSQL. Amazon RDS gère vos bases de données Amazon Aurora en prenant en charge les tâches chronophages telles que la mise en service, l'application des correctifs, la sauvegarde, la récupération, la détection des pannes, ainsi que les réparations. Vous payez un forfait mensuel pour chaque instance de base de données Amazon Aurora utilisée. Aucun coût initial ou engagement à long terme n'est requis.

3.5.1 Amazon Cognito

Amazon Cognito permet d'ajouter facilement et rapidement l'inscription et la connexion des utilisateurs ainsi que le contrôle d'accès aux applications Web et mobiles. Amazon Cognito s'adapte à des millions d'utilisateurs et prend en charge la connexion avec les fournisseurs d'identité sociale tels qu'Apple, Facebook, Google et Amazon, et les fournisseurs d'identité d'entreprise via SAML 2.0 et OpenID Connect

3.5.2 Fonctionnalités

Répertoire d'utilisateurs sécuritaire et se mettant à l'échelle

Les groupes d'utilisateurs d'Amazon Cognito fournissent un répertoire d'utilisateurs sécurisé qui s'étend à des centaines de millions d'utilisateurs. En tant que service entièrement géré, les groupes d'utilisateurs sont faciles à configurer sans avoir à s'inquiéter de la mise en place d'une infrastructure serveur.

Fédération d'identité sociale et d'entreprise

Avec Amazon Cognito, vos utilisateurs peuvent se connecter via des fournisseurs d'identité sociale tels que Apple, Google, Facebook et Amazon, et via des fournisseurs d'identité d'entreprise tels que SAML et OpenID Connect.

Authentification basée sur les standards

Amazon Cognito User Pools est un fournisseur d'identités normalisé et prend en charge les normes de gestion des identités et des accès, telles que Oauth 2.0, SAML 2.0 et OpenID Connect.

Sécurité pour vos applications et vos utilisateurs

Amazon Cognito prend en charge l'authentification multi-facteurs et le chiffrement des données au repos et en transit. Amazon Cognito est éligible HIPAA et conforme aux normes PCI DSS, SOC, ISO/IEC 27001, ISO/IEC 27017, ISO/IEC 27018, et ISO 9001.

Contrôle d'accès pour les ressources AWS

Amazon Cognito fournit des solutions pour contrôler l'accès aux ressources AWS à partir de votre application. Vous pouvez définir des rôles et associer des utilisateurs à des rôles différents afin que votre application puisse accéder uniquement aux ressources autorisées pour chaque utilisateur. Autre possibilité : vous pouvez également utiliser les attributs des fournisseurs d'identité dans les stratégies d'autorisation AWS Identity and Access Management. Cela vous permettra de contrôler l'accès à des ressources pour les utilisateurs qui remplissent des conditions d'attributs spécifiques

Intégration facile avec votre application

Amazon Cognito fournit des solutions pour contrôler l'accès aux ressources AWS à partir de votre application. Vous pouvez définir des rôles et associer des utilisateurs à des rôles différents afin que votre application puisse accéder uniquement aux ressources autorisées pour chaque utilisateur. Autre possibilité : vous pouvez également utiliser les attributs des fournisseurs d'identité dans les stratégies d'autorisation AWS Identity and Access Management. Cela vous permettra de contrôler l'accès à des ressources pour les utilisateurs qui remplissent des conditions d'attributs spécifiques.

3.6 API Gateway

Amazon API Gateway est un service entièrement opéré, qui permet aux développeurs de créer, publier, gérer, surveiller et sécuriser facilement des API à n'importe quelle échelle. Les API servent de « porte d'entrée » pour que les applications puissent accéder aux données, à la logique métier ou aux fonctionnalités de vos services backend. À l'aide d'API Gateway, vous pouvez créer des API RESTful et des API WebSocket qui permettent de concevoir des applications de communication bidirectionnelle en temps réel. API Gateway prend en charge les charges de travail conteneurisées et sans serveur, ainsi que les applications web.
API Gateway gère toutes les tâches liées à l'acceptation et au traitement de plusieurs centaines de milliers d'appels d'API simultanés, notamment la gestion du trafic, la prise en charge de CORS, le contrôle des autorisations et des accès, la limitation, la surveillance et la gestion de la version de l'API. Aucuns frais minimum ou coûts initiaux ne s'appliquent à API Gateway. Vous payez pour les appels d'API que vous recevez et la quantité de données transférées et, avec le modèle de tarification par paliers de l'API Gateway, vous pouvez réduire vos coûts en fonction de l'utilisation de votre API

3.7 Amazon Dynamodb

Amazon DynamoDB est une base de données NoSql de type clé-valeur et de documents, offrant des performances de latence de l'ordre de quelques millisecondes, quelle que soit l'échelle. Il s'agit d'une base de données multi-région, multi-active et durable entièrement gérée, avec des systèmes intégrés de sécurité, de sauvegarde, de restauration et de mise en cache en mémoire pour les applications à l'échelle d'Internet. DynamoDB peut traiter plus de 10 mille milliards de demandes par jour et supporte des pics de 20 millions de demandes par seconde.

La plupart des entreprises du monde qui connaissent la croissance la plus rapide, comme Lyft, Airbnb et Redfin, ainsi que Samsung, Toyota et Capital One s'appuient sur la

mise à l'échelle et les performances de DynamoDB pour prendre en charge leurs charges de travail stratégiques. Des centaines de milliers de clients AWS ont choisi DynamoDB comme base de données de clés-valeurs et de documents pour leurs applications mobiles, Web, de jeux, de technologie publicitaire, IoT, etc. nécessitant un accès à faible latence aux données, quelle que soit l'échelle

3.8 CloudWatch

A compléter

3.9 Reflex

A compléter

Chapitre 4

Approche suivie et solution proposée

Dans cette section nous ferons une présentation détaillée des solutions que nous avions proposées par rapport à nos missions.

4.1 Méthodologie de travail : Scrum avec Agile

A compléter

4.2 Les différentes missions et solutions

4.2.1 Mission 1

Etude de faisabilité sur le déploiement d'une application Front-End dans un environnement AWS. L'application frontend(web app flutter) d'Assurly est hébergée sur Firebase, tandis que l'essentiel de ses ressources se trouve dans le cloud d'Amazon. C'est dans le but d'unifier nos environnements cloud que cette mission m'avait été confiée. Pour cela j'avais exploré deux possibilités :

Solutions

L'utilisation de S3 comme repos de déploiement La solution ici consiste à créer un repos S3 de le rendre public, afin de pouvoir accepter tous les trafics. De configurer Code Pipeline pour les besoins de CI/CD avec Github, De configurer Cloudfront pour la gestion des trafic et Route 54 pour le routage.

L'utilisation d'Amplify comme comme repos de déploiement Cette solution consiste à utiliser le service d'hébergement qu'offre Amplify et son système de CI/CD en l'associant à système de versioning : Github. CloudFront était utilisé pour la gestion du trafic et Route 54 pour le routage. NB : C'est la solution 2(l'utilisation d'Amplify comme comme repos de déploiement) qui avait été retenue. Celle-ci est simple et plus adaptée.

4.2.2 Mission 2

Le déploiement et l'intégration d'un service web (Reflex) dans un environnement AWS. La souscription au produit d'assurance Assurly est subdivisée en trois parcours : P1, P2, et P3. Un utilisateur ne peut se retrouver dans un seul parcours en fonction des données fournies. Si un utilisateur se retrouve dans le parcours P3 un questionnaire plus complexe est nécessaire, c'est à cet instant qu'intervient Reflex.

Solutions

Reflex est composé de trois modules de base : CEF pour frontend, DOCS pour la gestion des rapports et RAS qui constitue le cœur du système Reflex. La procédure de déploiement consiste mettre à jour le repos Github qui contient les fichiers nécessaires à la construction de notre container, à se connecter à notre EC2, à puller les fichiers du repos Github depuis un répertoire de notre EC2 puis à construire et déployer notre container avec la commande docker-compose

4.2.3 Mission 3

Sécurisation d'une API Rest développée avec la technologie Serveless d'AWS (Lambada + python)

4.2.4 Mission 4

Le déploiement d'un système de web scraping dans un environnement AWS

Annexe A

Frequently Asked Questions

A.1 How do I change the colors of links ?

The color of links can be changed to your liking using :
`\hypersetup{urlcolor=red}`, or
`\hypersetup{citecolor=green}`, or
`\hypersetup{allcolor=blue}`.

If you want to completely hide the links, you can use :
`\hypersetup{allcolors=.}`, or even better :
`\hypersetup{hidelinks}`.

If you want to have obvious links in the PDF but not the printed text, use :
`\hypersetup{colorlinks=false}`.

Bibliographie

ARNOLD, A. S. et al. (mars 1998). « A Simple Extended-Cavity Diode Laser ». In : *Review of Scientific Instruments* 69.3, p. 1236-1239. URL : http://link.aip.org/link/?RSI/69/1236/1.

HAWTHORN, C. J., K. P. WEBER et R. E. SCHOLTEN (déc. 2001). « Littrow Configuration Tunable External Cavity Diode Laser with Fixed Direction Output Beam ». In : *Review of Scientific Instruments* 72.12, p. 4477-4479. URL : http://link.aip.org/link/?RSI/72/4477/1.

WIEMAN, Carl E. et Leo HOLLBERG (jan. 1991). « Using Diode Lasers for Atomic Physics ». In : *Review of Scientific Instruments* 62.1, p. 1-20. URL : http://link.aip.org/link/?RSI/62/1/1.