Notes module 1 :

AWS offre :

* COmpute
* Storage
* Network security
* Blockchain
* Machine learning
* IA
* Robot development
* Video production
* Couverture satellite

EC2 => permet de créer des instaces : des virtual server

AWS : pay only for what U Use

Create and delete instance on a click

Avantage du cloud:

-pay as you go

- econolie d’echelle

- flexibilité

**Module 2:**

EC2

Multitenancy: c’est ce qui permet à AWS de faire plusieurs VM indépendantes sur un seul serveur

EC2 : On peut choisir l’OS (Win ou Linux) et quelles apps faire tourner (webAPPs, database, Internal business App, third partei….). On peut aussi choisir la privacy de nos request (more later)

Type de EC2 :

General purpose : Balance ressouces (memory, CPU, networking) peut être utilisé pour de multiple taches ( web server, Code repo, small-medium database, serveur de jeu)

Compute optimized : Compute des tache intensive (serveur de jeu, High performance computing, modelisation scientique, webapp gourmande, app gourmande, serveur de jeu pour avoir de la haute performace (Pas de lag, pu de latence eetc..))

Memory optimized: pour les tâches demandant beaucoup de mémoire (data pattern matching, hardware accelerator, )pour manipuler des grosses donnée (genre bcp de ram)

Accelerated computing : ideal for workloads such as graphics applications, game streaming, and application streaming.

Storge optimized: pour les taches qui demandes de stocker de grande quantité de données

Plusieurs purchase options :

On-demand (par heure ou par seconde selon le type d’instance) (très populaire, souvent utilisé en début de projet, quand on ne connait pas encore vraient la workload à venir)

Saving plane : On paie moins cher mais on commit à un certain nombre d’usage min pour un engagement de 1 ou 3 ans. Jusqu’à 72% d’économie mais on paie pas upfront (en gros on s’enage à au min 8h/ jour d’usage (sans préciser si c’est lux, windows, la zone, le type de machine) pendant 1 ou 3 an pour avoir de réduc)

Reserved instance : pour les usage très constant ou prévisible, permet jusqu’à 75% de réduction pour un engagement de 1 à 3 ans, on doit bien parametré sont serveur avant (choisir l’os, la région (on peut prendre une région basculable mais on à une moins grosse réduc) etc..) -> payable soit upfront (en une fois), soit partiellement upfront, soit tout en pas upfront

Spot instace : Permet de prendre les capacité qu’amazon n’utilise pas pour 90% de réduc mais par contre Amazon peut reprendre l’instace dès qu’ils en ont besoin avec seulement deux minutes de warning pour tout save

Dedicated host : Pour avoir vraiment des serveur rien que pour toi qui doivent avoir certains requirement très particuliers

**Scalabilité des capacités** :

AWS EC2 propose de l’autoscaling (permet de ne pas sur aloué des serveur et de ne pas avoir de manque de capacité en cas de pic)

Deux type de scaling : Dynamique scaling (repond à un changement de demande)/ Predictive scaling prédit le nombre d’instace nécessaire et y alloue les ressources nécessaire (on peut combiner les deux)

Quand tu setup ton autoscale, tu peux paramétré un minimum d’instace qui tourne (par exemple 1 pour tjrs être online), un max, et un desired (celui que tu penses avoir le plus souvent). Si tu ne setup pas desired, la minimum est considérée comme desired.

**Elastic Load Balancing (ELB)**

Permet de gérer automatiquement la gestion des charge de chacune des instances . Ce module n’est pas obligatoire et tout peut être gérer par un DSI mais souvent c’est quand même bien pratique de ne pas tout avoir à faire à la main

Ce service fonctionne au niveau régional (et pas par instance) (ce qui est bien apparemment) cela permet de n’avoir qu’une seule Url pour le front et pour le back et le ELB repartira lui-même au bon composant

Il s’autoscale tout seul et gratuitement

**Messaging and queuing**

De base, les services sont tighly couple ce qui pose un pb car si le service B dysfonctionne, cela va faire dysfonctionner le service A qui lui envoie des requetes (approcho monolythique).

On essaie donc « d’ajouter un buffer » dans les requêtes et d’avoir des services loosely coupled (approche avec des microservices)

Pour faire cela il y a deux services proposés par amazon :

Amazon SQS : SQS permet de send, store et receive msg entre software pour n’importe quel volume de msg. SQS crée des queus avec les difvférentes requetes reçue et peu créer plusieurs queue si besoin de scaler il y a

Amazon SNS : assez similair mais permet également de créer des « topic sns » qui servent à envoyer des notifications aux utilisateuurs ; Les utilisateurs (réel ou vm ou serveur etc..) sont donc abonné aux notifications d’un service

Additionnal Compute service :

EC2 demande un peu d’investissmeent de temps pour créer les instance set les gérer.

Si on ne veut pas prendre ce temps part manque de temp, de compétence ou d’intérêt, AWS propose une option « Serverless» ou on n’a pas accès à l’environnement (créer et gérer les instances, les maintenir les updater etc) et ou AWS le fait à ta place.

**AWS Lambda** est une des application pour faire du serverless qu’AWS propose

AWS lambda permet de créer des fonctions, et y associer un trigger. La fonction est envoyé dans un serveur où elle est traitée et compilée et lorsque le trigger est activé, la fonction s’active. On ne s’occupe pas de savoir comment ou quand la fonction est compilée, on a juste le résultat voulu

Lamba va pourvoir scale en fonction du nombre de triger activé pour que tout soit tjrs fluide

Lambda est fait pour exécuter des fonctions de moins de 15 min . Avec lambda, on ne paie que quand le code tourne

Si on veut avoir la scalabilité de Lambda mais pouvoir avoir accès à l’environnement, on peut utiliser les container de AWS Elastic container service (ECS) ou de AWS Elastic Kubernetes Service (EKS)

Ce sont des Container orchestration tool (ils gèrent de nombreux container) .Ils utilisent Docker pour faire leurs container

Un container : Un package pour un code (c-a-d les biblio ou leur nom et version, le code et les paramètres)

ECS et EKS peuvent tourner sur EC2 et on doit le paramétrer mais on a un accès approfondi à son parametrage

De ce que je comprend, ECS c’est fait maison par amazon et EKS est basé sur Kubernetes, un ogicile open source populaire avec qui Amazon collabore.

Si on ne veut pas s’embeter avec EC2 pour les orchestration tool, on peut utiliser AWS **Fargate**.

C’est une solution serveurless pour EKS ou ECS

**Module 3 :**

Il créaint des « groupes de datacenters » appelée région. Chaque région est raccordée aux autres régions avec des fibres optiques à haut débit.

On peut choisir quelle sur quelles région on déploie nos données.

Les données stockées dans une région ne vont pas aller dans une autre sans notre accord explicite.

Une région est un espace physique (genre Paris ou l’IDF ou le tarn)

Comment choisir une région :

* Conformité aux normes et lois en vigueur (Facteur prioritaire et limitant)
* Proximité avec les users (si 95% de tes user sont à Singap, prend les serveurs de singap) pour avoir moins de ms
* Les fonctionnalités disponibles (toutes les fonctionnalités d’AWS ne sont pas disponible dans toutes les régions (car chère ou long à déployer ou en phase de test)
* Prix : Certaines régions sont juste plus chère que d’autre à équipement égal (cout de l’energie, du personnel cout fixe etc)

Une région est donc composée de plusieurs Datacenter appelé Availability Zone (AZ). Chaque AZ sont éloignée d’’une 15 aine de KM les unes des autres pour éviter qu’un problème impacte deux datacenter d’un coup

On recommande de toujours faire tourner ses instances sur aumoins 2 AZ

Certains élément comme le ELB tourne en parrallès au niveau régional. En règle générale , les services labelisée « Regional scope service » tournent au niveau régional et non des AZ et ont donc de base une protection réginal contre les aléas

Si on a des clients dans différentes région, on peut créer des copie de donnée dans différentes région. Mais pas forcement des copie entière, ddes copie de « mémoire cache » on appelle ça de CDN

AWS propose **Amazon Cloudfront** comme CDN. Il permet de transmettre les données le plus vite possible à travers le monde. Il utilise *Edge location* pour accélérer les communication peu import ou sont les utilistateur. Chaque Edge location a aussi une addresse URL (DNS) appelée **Amazon route 53** pour amener chaque utilisateur vers l’edge location la plus proche de sa position.

**AWS outpost** permet de créer une sorte de mini région dans les datacenter de son entreprise. Cela permet d’utiliser 100% des fonction nalité d’AWS avec les donnée stockée dans son datacenter. (C’est peu utilisé)

Pour interagir avec les différents services d’AWS on peut :

* Utiliser l’AWS Management console
* Utiliser l’AWS Command Line Interface (CLI)
* Utiliser les AWS Software Développement Kits (SDKs)
* AWS Elastick Beanstalk
* AWS CloudFormation

AWS Management console :

Depuis un navigateur, en étant connecté. Pratique pour débuté pour comprendre comment ça marche. Pratique aussi pour faire des environnement test, voir les factures AWS, monitorer ses projets et travailler avec les ressources non-techniques

Cependant cet outils est surtotut recommandé aux débutant car il faut passé par de multiples écrans et tout paramétrer à chaque fois pour créer une instance ce qui prend du temps et occasionne des erreurs.

AWS Command Line Interface (CLI) :

CLI permet de faire des appels aux API AWS pour créer des instances depuis un terminal. Comme ça on peut créer un scrit permettant de créer une instance comme on veut. Ca permet de ne pas avoir à tout refaire à chaque fois. En plus cela permet d’automatiser le lancement d’instance

AWS Software Développement Kits (SDKs) :

Permet d’interagir avec AWS avec différents langages de programmation (C++, JAVA, .Net, etc..).

AWS Elastick Beanstalk :

Si on fourni notre code et la configuration voulu, il créer l’environnement et le dimensionne à notre place. Il permet également de sauvegarder la configuration d’un environnement, pour pouvoir le redéployer facilement.

Il va :

* Adjust capacity
* Load balancing
* Automatic scaling
* Application health monitoring

AWS CloudFormation :

Un outils d’infrastructure as code. Permet de créer des instances avec des Json ou des fichier txt.

Comme ça il y a juste a fournir seulement les specs qui nous importe et le reste est géré par CloudFormation.

Ça marche pour les solution EC2 mais aussi le stockage, les BDD, les Analytics le ML.

Si on dit à CloudFormation que notre service doit être opérationnel dans plusieurs régions, il va cloner l’infra qu’il a créer dand chaque région

De même si CloudFormation detecte une erreur lorsqu’on modifie l’infrastructure, il va rollback jusqu’à un état ou il n’y en avait pas

**Module 4**

Amazon VPC permet de lancer tes instances dans des Virtual network que je définis ce qui permet entre autre de dire quelles IP sont privée et pulique.

On peut y créer des Subnet, des groupes d’IP. Ce qui est public (connecter à internet) va être dans un public Subnet, alors que ce qui est privé (genre du back ou du stockage) va être dans des Private Subnet.

Un VPC contient donc entre autre les instance EC2 et ELB.

Pour qu’un VPC soit public, il faut lui rattacher un Internet gateway. Pour qu’il sot privé, on lui met un Virtual Private Gateway qui ne laisse passé que les gens connecté (c’est-à-dire venant d’un réseau que l’on considère approuvé) A virtual private gateway enables you to create a VPN connection between your VPC and a private network, such as your company’s data center.

Cependant cela passe toujours pas des routes internet pas forcement sécurisé et susceptible aux aléas. Pour parer à ça il y a **AWS direct connect.** Cela permet de créer une ligne de fibre direct entre nos serveur et les serveurs AWS.

Cependant le subnet regroupe plusieurs EC2 mais le subnet ne peut pas faire de différenciation sur qui peut accéder à quel instance dans le subnet. Pour cela il faut mettre des *Security group*. Par défaut chaque instance EC2 se créer son propre sécurity group et ne laisse aucun pacquet entrer. On peut évidemment le modifier pour accepter certain type de trafic genre du https pour un site web

La grande différence entre un subnet et un security group est la suivante :

* Les subnet sont stateless : les pacquets sont vérifié à l’entré et à la sortie à l’allée ET au retour
* Les security group sont statefull : les pacquet ne sont vérifié à l’entrée et à la sortie (pas par défaut mais possible de le paramétrer) à l’allée mais pas au retour car les sécurity group « se rappellent » que le pacquet est safe

Route 53 : c’est le DNS de AWS

Il (a priori surtout cloudfront en fait) permet de router les users selon plusieurs critères :

* La latence
* La geolocalisation DNS => donne accès au bon service selon l’addresse IP (Netflix franc esi en France)
* La Géoproimity routing => Permet d’envoyer la requête vers le serveru situé le plus près physiquement de l’user
* Weighted round robin

Il permet également d’acheter des noms de domaine et de gérer ce nous appartenant

**Module 5 : Storage et Database**

Important : Quand on créer une instance EC2, on a avec quelques petis blocs mémoire fournis. Cependant, il faut bien comprendre que ces blocs sont rattachés à un AWS host et que si l’instance est stoppé ou change d’host, alors les données sont perdues (pratique pour des scrap data ou des donné facile a retrouver). C’est pour ça que les données que l’on veut conserver (même un peu) doivent être stockées.

Pour stocker il existe Amazon Elastic Block Store (EBS)

C’est du block storage

Avec EBS, on peut créer de HDD virtuels appelés volumes et les attachés à des instances EC2. Elles sont attaché mais indépendantes dans le sens ou même si l’instance est stoppé, les données persisteront .

Un HDD peut stocker jusqu’à 16 TB de donnée. Sont Solid states par défaut.

On peut configurer la taille, le type et la config des volumes. Il faut également configurer l’instances EC2 pour lui dire qu’il faut écrire sur ce volume.

EBS permet de créer des snapshots pour faire des backups des data stockés sur un volume. L’avantage d’une snapshot c’est que faire une 2ème snapshot ne sauvegarde que ce qui a change depuis la première (économie de mémoire)

S3 (OMG) : Permet de stocker et récupérer un volume infini de donnée (tant que tu as de l’argent)

Les données sont stockés comme des objets

Les données sont stockés dans des buckets (équivalent des dossiers)

On ne peut pas uploader d’objet pesant plus lourd que 5TB (c’est beaucoup)

On peut y faire du versionning d’objet pour ne jamais en supprimer par accident

On peut faire de la gestiion d’accès (qui peut voir, modifier etc)

Il y a plusieurs Tiers de data selon ce qu’on stockk (donnée à longue conservaion pour des audits, données fréquemment socillicité etc.) pour gérer le % de chances qu’une donnée soit perdu (entre autre :

* S3 standard : 10^-9% de chance de perdre la donnée sur 1 an. Utilisé pour les données auxquelles on cherche souvent à accéder. Les données sont tockés ds minimum 3 availability zone.

Le stockage est plus cher que les stockages pour les données peu accédées

S3 standard permet aussi de stocker des pages statiques de sites web. Pour cela on upload le code html sur S3 et on coche une case.

* S3 Standard-Infrequent Access (S3 Standard-IA): Pour les données dont on a pas souvent besoin mais quand on en a besoin, on en a besoin vite (exple les backups, les recovery en cas de catastrophe)

Similaire à S3 Standard mais le prix de stockage est moins cher et le prix d’accession est plus cher. Les données sont cependant aussi rapidement accessible que pour standard.

* S3 Intelligent-Tiering : idéal pour les donnée dont le besoin d’accès change de façon inconnu . Deande un petit peu de monitoring mensuel et à un petit cout d’automatisation par objet. L’idée est que si un objet stocké dans ce tier est de base au même priux que standard. Si il n’est pas utilisé pdt 30 jours consécutifs, il passe au même prix que standard infrequent. Si il est réutilisé, il repasse au prix de standard etc…
* S3 Glacier Flexible Retrieval => pour les données qui ont besoin d’être stockées longtemps mais pas accessible rapidement (exemple en cas d’audit), ça sert d’archive. On peut soit bouger de sdonnée dans glacier soit créer des coffre fort et le peupler. Si des document doivent être lock pdt un certain temps, on peut créer des vault policy et blocker les vault. On peut y créer des policy tq Write once/read many. / !\ une fois vérouillée, une policy ne peut pas être modifiée. La récupération de donnée peut prendre de 1 min à 12h
* S3 Glacier Instant Retrieval => pour les datas que l’on veut archiver et dont on veut pouvoir avoir accès rapidement. Peut Accéder à des objet en qq millisecondes. (aussi rapide que standard)
* S3 glacier Deep Archive => Pour les donnée d’archivage ou le temps d’accès n’est pas une priorité (prend entre 12 et 48h d’accès). Option de stockage la moins chère. Les données sont stockées dans au moins 3 Availability zone dans des zone géographique distinctes. (Fait pour les données dont on veut avoir accès max 1 à 2 fois par ans)
* S3 Outpost : Permet de créer des bucket S3 sur AWS outpost (AWS on premise). Cela permet entre autre de gérer facilement la redondance de donnée.

Les data peuvent bougé automatiquement de tier si on veut (genre on dit que tous les docs dans le dossiers compta vont dans X vault pdt 10ans tous les 5 janvier). On appelle ça des livecycle policy. Une policy peut impliquer de bouger de plusieurs tiers au fil du temps

S3 VS EBS :

S3 est web enable (tout a déjà son URL galadrim RPZ), Regionnaly distributed ( TRES faible chance de perte de donnée), très peu chère, serverless

!= entre object storage et block storage => dans l’object storage, chaque élément est un tout inséparable. Le problème est que contrairement au block storage, si on modifie un bout du fichier, il faut TOUT réuploader alors que le block ne réupload que la partie modifiée.

Donc si on utilise des fichiers que l’on ne va pas modifier ou alors très peu modifier => S3 sinon plutôt EBS

**Amazon file Système (EFS) :** sharefile système (Sharepoint) :

Permet à plusieurs instance d’accéder aux data en même temps. Et ça se scale automatiquement

C’est différent avec EBS car EBS doit être rattaché à une instance EC2, que l’instance EC2 doit être dans la même AZ que l’instance EC2, comme EBS est équivalent à un HDD, il ne s’autoscale pas . EBS est une ressource de niveau AZ.

EFS permet à plusieurs instances de lire et écrire dedans en même temps. C’est un Linux file système. Enfin c’est une ressource régionale (chaque instance EC2 dans la région peut écrire dans l’instance EFS). EFS s’autoscale.

EFS fait du file storage : façon de stocker optimisée pour le partage de fichier (plusieurs personnes doivent accéfder à la même donnée en même temps), contrairement au block storage.

**Amazon Relational Database Service (RDS)**:

AWS supporte les BDD MySQL, PostgreSQL, Oracle, Microsoft SQL Server, Amazon Auraura et d’autres

RDS permet de créer des bases de donnée relationnelle mais avec des features préfaite :

* Du patching automatique
* Des backups
* De la redondance
* Du failover (bascule immédiate d’un serveur à un autre en cas de défaillance d’un serveur)
* Du disaster recovery

**Amazon Aurora :**

BDD relationnelle compatible avec PostgreSQL et MySQL.

Est jusqu’à 5 fis plus rapide que MySQL et 3 fois plus que Postgre.

Ne coûte qu’ 1/10 des BDD trouvables sur le marché

Les données sont répliqués dans 6 serveurs différents pour ne pas avoir de pertes.

On peut créerr jusqu’à 15 read replicas (copies en lecture seules pour avoir encore plus d’availabilité)

Des backups sont stockés en permanence dans S3

La recovery peut se faire selon plusieurs date, pour choisir de recover ce que l’on veut

**Amazon Dynamo DB :**

C’est une BDD serveurless. On y créer des tables et tout se scale automatiquement. Les données y sont stockés avec redondances sur plusieurs AZ. Dynamo DB a un temps de réponse très cours (en millisecondes) de façon constante.

Dynamo DB n’utilise pas SQL. Apperemment, SQL peut avoir des soucis quand on a trop de requetes et ne permet pas de variation dans une table (après si t’as besoin du deuxième point, c’est que t’as chier dans la colle)

Dynamo est une BDD **non-relationnelle.** Cela permet d’avoir de la flexibilité (ajouter et supprimer des attributs à un objet à n’importe quelle moment). Cela permet de faire des query plus simple qui ne s’étalement que sur une table

Dynamo DB est Purpose built, c-à-d que cela ne convient qu’à certains cas d’usage mais pas tous.

RDS VS DynamoDB :

RDS : -High availability et recovery automatique/ Le client possède, construit et est responsable des données et les schema relationnelle / Le cluent contrôle le réseau de donnée

DynamoDB : Utilise des key value (noSQL) / Permet de délivrer un très grand nombre de donnée/ Peut stocker plusieurs PétaBytes de données) / l’api est très précuse et permet un contrôle fin des opération effectuées sur la BDD.

Si on veut faire des jointures complexes (pour faire de l’analyse de donnée par ex), RDD est à privilégier.

Sinon DynamoDB est largement suffisant et est mons complexes, moins cher et moins laggy

**Amazon redshift :**

Un data Warehouse est un espace de stockkage ou on va vouloir stocker un grand volume de donnée qui ne vont pas être changé mais être utilisées pour analyses (exemple l’ensembles de ventes Amazon lors du blackfriday 2024, cela ne vas jamais changer mais représente un énorme volume de données).

Il existe plusieurs providers de data warehouse. Si on en a déjà un, on peut le faire tourner sur AWS, c’est juste du transfert de données. Mais cela va prendre de l’énergrie pour s’assurer que tout fonctionne de tout scaler, de tout rendre redondant etc.

Redshift est de data warehousing as a service. C’est massivement scalable, c’est courant de fairre tourner des pétaBytes de données sur redshift. On peut faire tourner des query SQL sur de exabytes de données (10^18 octets). Redshift permet d’être jusqu’à 10 fois plus performant que les warehouse classique.

**AWS Database Migration Service (DMS):**

Permet de la migration de donnée sur AWS depuis du on premise sur AWS où même depuis un autre cloud sur AWS. DMS permet également de laisser la DB d’origine complètement opérationnelle durant la migration. DMS permet également de minimizer dle downtime pour les applications qui reposent sur la DB (pas sur de voir la diff avec le point précédent mais bon…).

DMS permet de migrer d’une DB d’un type vers une DB d’un AUTRE TYPE. Dans ce cas, il y a d’abord une convertion avec les AWS convertion tool dans le type voulu, ensuite DMS fait comme une migration normale.

DMS permet également de faire des migration pour le dev ou le test (remember galadrim), de la database consolidation (fusion de plusieurs DB en une seule pour réduiire la complexité, le cout et optimiser le fonctionnement) et de la replcation continue de DB (pour parer aux catastrophes ou pour avoir une grande availability géographique).

Il existe d’autres services AWS de BDD :

-Amazon DocumentDB => Un service de document database (autre tupe de noSQL à priorià qui support MongoDB

-Amazon Neptune => Un service de graph database utilse pour les datasets fortement connectés (outils de recommandation, la detection de fraude…)

-Amazon Quantum Ledger Database (QLDB) => C’est un service de ledger database. (En gros un truc écrit sur la DB ne peut plus jamais être modifié ou en tout cas un historique complet est conservé). Utilie pour la traçabilité de données (domaine bancaire, assurancielle etc)

-Amazon Managed Blockchain => Permet de créer et de gérer des réseaux de blockchain avec un framework open-source

-Amazon ElastiCache => permet d’ajouter des couches de cache sur une ou plusieurs DB pour permettre d’améliroer les temps de lecture pour des requetes que l’on fait souvent. Cela support deux type de data store Redis et Memcached

-Amzzon DynamoDB Accelerator  => Ajoute du cache pour DynamoDB, permt de réduire le temps de reponse qui passe de moins d’une dizaine de millisecondes à quelques microsecondes.

**Module 6 : Security**

Share responability model :

En terme de responsabilité de sécurité, AWS et le client se partagent les tâches :

AWS s’occupe de la sécurité de l’infrastructure, que le Network fonctionne et qie l’hyperviseur fonctionne en assurant une bonne sacalabilté, AWS est responsable du cloud

Le client lui doit s’assurer lui-même de ce qu’il se passe dans le cloud il est responsable dans le cloud, que l’OS, les applications fonctionne bien et que les datas soient bonnes

Cette distinction est claire et net et JAMAIS un opérateur AWS nous demandera les accès à notre OS. Ce n’est pas de leur responsabilité.

On a la responsabilité de qui accède à quelle donnée ou à quelle application. Pareille pour l’encryption, ça relève du client.

Permission utilisateur et accès :

Quand on créer un compte AWS, on a un compte avec un accès root sur toutes les instances qu’on va créer. Comme ce compte admin est surpuissant, on recommande la double authentification

**AWS Identity and Access Management (IAM) :** Permet de créer et de gérer finement les accès et les compte utilisateur. Par défaut, un utilisateur qui viens d’être créé n’a aucun accès. Il ne peut même pas se login. Chaque permission doit être donnée explicitement.

On est censé applique le principe de privilège minimum : toujours donner les accès minimaux requis à une personne pour sa mission.

On peut créer des Policies : des JSon qui spécifie l’accès à UNE fonctionnalité pour UNE instance que l’on peut rattaché à une ou plusieurs personnes

IAM permet de créer des groupes (des classes d’utilisateurs avec tous les mêmes accès)

On peut aussi créer des rôles. Les rôles eux sont temporaire (une personne qui un jour a accès à un sharepoint et le lendemain non (exemple AdS)) et donnent accès ou non à certaines actions. Un rôle peut être donnée à un utilisateur, à une entité externe, à des applications et même à d’autres services AWS. Le rôle surpasse toutes les permissions précédmeent établie (positive comme négative).

On peut mapper les identifiant entreprise sur AWS pour que les employés n’aient pas à se log sur AWS. Pour cela il faut (entre autre) mapper leur compte entreprise sur un rôle IAM

On peut vite se retrouve perdu avec tous les compte AWS créer. C’est pour cela qu’il y a **AWS Organisation.**

Cela permet de centraliser la gestion des comptes AWS. De centraliser les paiement pour les différentes instances (et donc d’avoir des reducs). Cela permet aussi de créer un groupement hiérarchique des comptes. Enfin cela permet de réguler les accès au services AWS et aux API disponible.

On peut créer des Organisation Unit (OU) où tous les comptes AWS placés dedans auront les mêmes accès maximums pour cela on y applique des Services control policies (SCP). On aussi appliquer une SCP à un compte sans qu’il fasse partie d’une OU.

IAM permet donc de gérer les accès dans UN SEUL compte AWS alors que Organization permet de gérer Plusieurs copté AWS.

IAM permet plus de granularité alors qu’organisation place des limites pour tous les membres du compte AWS assocué

Organizations : Est plus pertinent pour les grandes entreprises ou les organisations ayant plusieurs comptes AWS. Il ajoute une couche de gestion supplémentaire pour la consolidation de la facturation et l'application de politiques globales.

Le root de AWS organisation va être le root pour tous les compte AWS associés

Il y a des fonctionnalité pour vérifier si on respecte bien les normes en vigueur (RGPD par exemple)

AWS a une liste de norme qu’elle respecte au niveau de ce dont elle est responsable. On a donc besoin de ne gérer la compliance que sur ce dont on a la main.

Le choix de la région lors du déploiement d’instance peut permettre de respecter des normes que l’on veut atteindre mais que certaines zones géographiques ne permettent pas.

L’encryption des donnée (parfois obligatoire pour certains type de données) peut soit être fait par l’entreprise avec ses propres méthodes soit, lorsque c’est disponible, directement par AWS en cochant une case.

AWs s’engage à fournir les documents prouvant qu’il respectent les normes qu’il annoncent respecter en cas de besoin. Ces document sont trouvable sur AWS Artifact.

Artifact permet également de voir, accepter et gérer ses agreements avec AWS pour la compliance en terme de données.

On peut aussi trouver des informations de compliance sur AWS compliance qui compile bcp d’infos

Protection contre le DDOS :

Les sécurity groups permettent de se prémunir de DDOS de requêtes n’ayant rien à voir avec l’instance. Surtout, les données sont refusé en utilisant non pas la puissance de juste notre instance mais de toutes l’infrastructure de la région (pour 0 coût)

ELB permet de ne pas se faire bloquer par les attaque à la mauvaise connexion : le ELB va traiter en parallèle toutes les requête et n’envoyer celle de l’user à la mauvaise connexio nuniquement quand celle-ci lui a entièrement été communiqué sans bloquer les autres. (toujours 0 coûts). les données sont traités en utilisant non pas la puissance de juste notre instance mais de toutes l’infrastructure de la région (pour 0 coût)

Enfin AWS Shield with AWS WAF donen à l’intelligence artificielle le pouvoir de voir qui se connecte et de détecter les utilisateurs malveillant. Il a la particularité de toujours se mettre à jour et d’être à la pointe de la détection.

AWS WAF est un Web application firewall qui bloque des addresse IP

Il existe aussi AWS Shiel Standard qui est moins puissant mais compris d base dans tous les services et qui permet de se protéger de la grande majorité des attaques DDOS

Pour l’encryption :

Les donnée sont encrypter at rest (c-à-d quand elle se reposent sur une bdd) et en transit

C’est encrypter de base sur DynamoDB.

On peut gérer ses clef de crytpographie sur **AWS key management service (KMS)** . KMS permet également de choisir qui peut gérer les clés

**Amazon Inspector** permet de tester la sécurité de nos infrastructure en permanence. Il regarde si on respecte bien les bonnes pratiques. Comme l’expoition d’instances EC2, les vulnérabilités etc

Il est composé de trois parties :

UN *Network configuration reachability piece*

Un agent Amazon qui peut être déployé sur une instance EC2

Un service de notation de sécurité.

On obtiens ensuite une liste de problèmes rencontré avec une gradation du problèmes et des pistes pour les solutionner. On peut soit trouver ces infos dans la console soit les récup depuis une API

**Amazon GuardDuty :**

Il analyse les métadonnées et plein de techniques dont l’IA pour identifier des menaces. L’intérêt et aussi qu’il tourne indépendamment de nos serveu AWS donc il ne va pas ralentir leur fonctionnement.

Module 7 : Monitoring

**Amazon Cloudwatch :** permet de monitorer ses infrastrucures AWS et les applications tournant dessus en direct. Ca mesure des donnée qu’on lui dit de regarder.

Ca nous permet de créer des etrics à regarder et d’envoyer une alerte lorssqu’une certaine valeur est atteint (trop haute ou trop basse) on appelle ça une Cloudwatch Alarm. L’alarme peut être une simple notification ou une action plus concrète (comme créer une nouvelle instance par exemple)

Cloudwatch comporte également un dashboard qui permet de rassembler tout un tas d’information qui nous intéresse en temps réel. Le dashboard ne bouge pas en temps réel mais se met à jour quand on l’ouvre.

Cela permet de réduire le *Mean Time To Resolution* (MTTR) et améliorer le *Total Vost of Ownership* (TCO)

**AWS Cloudtrail :** chaque requête y est loggé avec comme info qui a fait une requête, quand elle a été faite, quelle était l’addresse IP de la personne qui l’a fait et quelle a été la réponse, qu’est ce qui a changé et quel est le nouvel état.Cloudtrail permet de filtrer et trier les logs pour plus de clareté. C’est très bien pour les audits

CloudTrail possède une option appelée **Cloudtrail Insights,** cette feature optionnelle permet de détecter des appelle suspect à l’api de notre compte AWS

**AWS trusted Advisor :** C’est service AWS qui va évaluer notre utilization d’AWS et nou prodiguer des conseil. Ses critères d’amélioration sont :

* L’optimisation des coûts
* La performance
* La sécurité
* La fault tolérance (gestion des backup, des availability zone etc)
* Les limites des services (les limites des services AWS)

Trust advisor va ensuite créer des recommandation visibles directement depuis la console. Certains conseils sont gratuit et déjà inclus dans notre abonnement AWS, d’autre sont accessible uniquement selon le Support plan auquel on a souscrit.

Il va par exemple nous recommander la double authentification, d’éteindre des instances inutiliser pour garder des sous ou même nous rappeler de faire des backups si le dernier date d’il y a trop longtemps.

Il a trois niveau d’alerte : -1 tout va bien mais il nous montre ce qu’il a check, 2- Investigue car il y a probablement moyen de faire mieux, 3- Alerte problèmes en vue

**Module 8 : Pricing**

AWS à un free tier. Selon le service il peut se présenter sous 3 manières différentes :

- Toujours gratuit

- 12 mois gratuit une fois ton inscrption à AWS faite

- Essaie (une courte période pour tester)

Exemple : Lambda permet 1 million d’utilisation gratuite mensuelle (à l’infini) de même, DynamoDB donne accès à 25 Go de donnée stockée gratuite mensuelle. S3 permet, pendant 12 mois, de stocker jusqu’à 5Go gratuitement en standard storage. AWS Lightsail (on en a pas parlé mais à priori ça permet de déployer des stacks déjà faite pour des appli), à un mois de trial où on peut l’utiliser jusqu’à 750 heures (1 mois non stop)

Les autres services avec un free tier :

-Sagemaker (Service permettant de former et de déployer des IA)

-Comprehend Medical (Un service de traitement du langage naturel (NLP) spécialisé dans l'analyse des textes médicaux.)

-Amazon Cagnito (Un service qui permet d'ajouter facilement des fonctionnalités d'authentification, d'autorisation et de gestion des utilisateurs à des applications web et mobiles.)

-Amazon SNS

Les concepts de pricing d’AWS :

Le pricing d’AWS repose sur 3 principes :

-Payer seulement ce pour quoi on utilise

-Payer moins quand on reserve (réserver n’est pas possible pour tous les services)

-Plus on utilise, moins on paie (plus on stocke sur S3, moins le giga coute cher à stocker)

AWS dispose d’un simulator de coût : on détaille notre consommation estimée et une fourchette de coût s’affiche (le calcul est partageable en créant un lien). On peut estimer les coût totaux ou par groupes. On peut aussi faire du comparatif entre région ou entre services

Particularité de pricing : Pour S3 : on paie le stockage utilisé, les requetes mais aussi le transfert de donnée :

Si on transfert des infos entre des bucket s3 c’est gratuit, de même si on transfert d’un bucket s3 à un service de la même région.

Les options de maagmeent de données et de replication de ces dernières sont aussi payant

Dans la managemebt console, onpeut chercher billing et tomber sur le billing dashboard . On pejt y voir les cout actuel classé par sevices, les coûts du mois précédent et les coût prévus pour le mois suivant

On peut aussi voir notre facture en détail en cliquant sur bill

Si on a plusieurs compte aws que l’on gère avec Organization, on peut wrap toutes les factures pour n’en n’avoir qu’une seule pour le compte root de l’organization. Organization agrège tous les usage ce qui permet de profiter des réduction quand on utilise bcp AWS (use more, pay less)

**AWS Budget** :

Permet de tracker les coût de s’assurer qu’on reste dans nos limites. On recevra alors une alerte quand nos coûts excerdront où seront forecaster de dépasser la limite prévue

**AWS Cost explorer :**

Permet de voirr et d’analyser nos dépenses AWS avec un historique de 12 mois. De voir pour quel service on dépense le plus. On peut aussi filtrer par tag (on peut tagguer chaque instance et service)

On peut y créer des rapport avec des graphes et les sauvegarder

**Le support fournis :**

Le basic support est gratuit. Il inclut :

* Service client 24/7
* De la doc
* Des whitepapers
* Les forums de support
* AWS trusted advisor (la partie gratuite)
* AWS Personal Health Dashboard

AWS Développer support inclut :

* Tout ce qu’il y a dans le basic
* On peut envoyer un mail au customer support qui s’engae à nous répondre sous 24h peu importe quand le mail est envoyé et en moins de 12h en cas de défaillance du système

C’est pour les gens qui font des test, des expériences, des proof-of-concept

AWS Business Support inclut :

-Tout ce q’il y a dans Basic et Développeur

-Accès à l’ensemble de Trusted Advisor

-Une ligne directe pour contacter les ingénieurs du support par téléphone avec un temps de réponse sous 1h si le système est défaillant et sous 1H s’il est down

-On nous donne aussi accès au Infrastructure event management (une aide au lancement de service en gros)

Si on migre des infrastructure critiques sur le cloud, AWS propose

AWS Entreprise On-Ramp Support :

* Contient tout ce qui a déjà été présenté
* Un temps de réponse sous 30 min en cas de problèmes critiques
* Un accès à un pool de Technical Account Managers (TAMs) qui vont aider a coordonner notre utilisation d’AWS. Notre accès au TAMs est cependant limité

Enfin si on migre des infrastructure qui mettent en péril notre mission en cas de dysfonctionnement, il y a

AWS Entreprise Support :

* Tout comme avant
* Un temps de réponse sous 15 min en cas de problèmes critiques
* Un TAM avec accès infini qui nous est associé (donc expert de notre infra) qui vas nous assister sur l’optimisation et monitorer notre environnement de façon proactive

Qu’est ce qu’un TAM : Ils fonct partie de l’équipe de « conciergerie » pour les deux derniers support plan, il fournissent de la gestion d’évnènement sur nos infra, de la revue d’architecture pour vérif que c’est bien) et de la revue de la prod

**AWS Marketplace :**

C’est un catalogue numérique personnalisé qui simplifie le fait de trouver, gérer et déployer des software third-party

Il donne des options qui facilitent la vie comme des one click deployment, peut proposer des réduction de prix.

Il est très souvent possible de basculer un contrat annuel avec un third party en cours sur un serveur onpremise à AWS grâce au marketplace sans payer.

Si on-premise beaucoup de thrid party propose uniquement des contrats anuels, sur AWS on peut trouver beaucoup de ces thir party en pay-as-you-go et en free trial

On y trouve également des features por les entreprises :

* Customisation des prix et des termes de license
* Un marketplace privé ne contenant que les thrid party répondant à nos critères (légal ou de cybersécu)
* Intégration dans notre système d’approvisionnement
* Des outils de gestion de coûts

Il y a 8 catégories dans le marketplace :

* Infrastructure software
* Devops
* Data Producs
* Professional Services
* Business Applications
* Machine learning
* Industries
* IOT

Module 9 : migration

Basculer sur le coud est de base une action assez complexe qui demande d’intégrer de nombreuses personnes dans différentes boucles, en informer certains, demander des validaion d’autres etc…

Pour nous faciliter la vie, AWS propose **AWS Cloud Adoption Framework (AWS CAF)**

AWS CAF fournis des conseils pour faciliter la migration. Les conseils se concentrent sur 6 perspectives et les différentes personne que l’on doit impliquer pour notre migration :

* La perspective Business (genre les business analyst)
* Les gens (donc par exemple implication des RH)
* La gouvernance
* Plateforme (genre les cloud architect)
* Sécurité
* Operations

Cela permet d’avoir des bon process pour créer des inputs

A partir de ça on peut créer des AWS CAF action plans : des plans d’actions qui nous aident à guider notre entreprise dans notre voyage dans le cloud

Lorsqu’o migre n’importe quelle application ou groupe d’application, on doit choisir parmi 6 options (les 6 R) :

* Rehosting (aussi connu sous le nom de lift and shift), on dépose juste ses applis sur AWS sans rien optimiser. Cela est quand même utile rien que pour réduire les coûts (jusqu’à -30% par rapport à du on premise). Sachant quee l’on peut ensuite optimiser plus tard (genre après qq semaines ou mois d’utilisation) quand on maîtrise mieux les outils.
* Replatforming (aussi appelé lift, tinker (bidouiller) and shift). Très similaire au Rehosting, ici on bidouille qq params pour optimiser qq trucs mais on ne touche pas au code des applis, aucun dev ne doit coder pour la migration.) Gnre on passe de Mysql à Aurora

Les deux R suivant sont la pour des appli que l’on ne va pas migrer)

* Retire : supprimer les applications plus utilisés (la migration est l’occasion de faire une revue des apps et voir lesquels ne sont plus utilisés et ne plus les host)
* Retaining : pour les apps dont on a plus besoin longtemps (qq semaine/ mois) et dont ça ne vaut pas le coup de prendre le temps de migrer sur AWS

-Repurchasing : profiter de la migration pour terminier un contrat pour une app obsolète et passer à un concurrent ou même à une soluce AWS existante

-Refactoring : Récriture de cod (from scratch ou non,) et la on ajoute des features importantes que lo’n veut vraiment

La migration de donnée prend du temps (parfois plusieurs mois pour migrer des pétaoctets). Pour faciliter cel AWSpropoise des services AWS Snow. On réserves un des services sur le terminal puis on reçoit au bout d’un moment un objet physique sur lequel stocké nos données :

* AWS Snowcone : Peut contenir jusqu’à 8 TO de donnée et peut faire de l’edge computing (EC2 et AWS IOT Greengrass). Une fois que tout y est stock » et qu’on valide, tout est envoyé sur un bucket s3
* AWS Snowball Edge (viens en deux version : edge compute optimizeoption et edge storage optimize option). Ca a plus de capacité de mémoire que snowcone .On peut y faire tournée des AWS Lambda, des EC2. Peut stocker jusqu’à 80TO
* -AWS Snowmobile : Transporte littéralement un serveur avec 100PétaOctets de stockage jusqu’à chez nous. Il est tracé avec GPS ? résistant au feu, est escorté par une escouade lors de son déplacement (oui oui)

Les 3 appareil de la snow family sont sécurisé et résistant à la température

Innovation avec AWS :

IA :

Speetch to text : Amazon Transcribe

Trouver des pattern dans du texte : Amazon Comprehend

Identifier des activité en ligne frauduleuse : Amazon Fraud Detector

Construire des chatbot écrit et vocaux : Amazon Lex

Machine Learning : Amazon SageMaker

Module 10 : the cloud journey

AWS Well archited framewor : c’est un framework permettant de construire des infra performante et résiliante pour nos applis. Ca a 6 pilliers :

* Operational excellence : La capacité à faire tourné et monitorer nos systèmes et faire de l’amélioration continue (opération as code, faire des annotation et de la doc, anticiper des failure et faire fréquemment des petit changement réversibles)
* Sécurité : Protéger nos données et nos systèmes
* Reliability : Récupérer en cas de parturbation des services ou des infras, scalability up et down des resources, réduire les perturbations du as des mauvaises configurations (teste des process de récupération, scaler dee façon horizontale sur pour maximiser l’availability et automatisation de la recovery en cas de failure
* Efficacité de la performance : utilisation efficace des infra mise à dispo, maintien de l’éfficacité quand la demande change et que les technos évoluent (utilisation du serverless et design de system capable de go global en qq minutes)
* Optimization de coûts
* Ecologie

Depuis la console, on peut accéder au Well architected tool. On peut lui demander de efaire un rapport notre compte AWS et les sujets à possiblement améliorer parmi les 6 pilliers