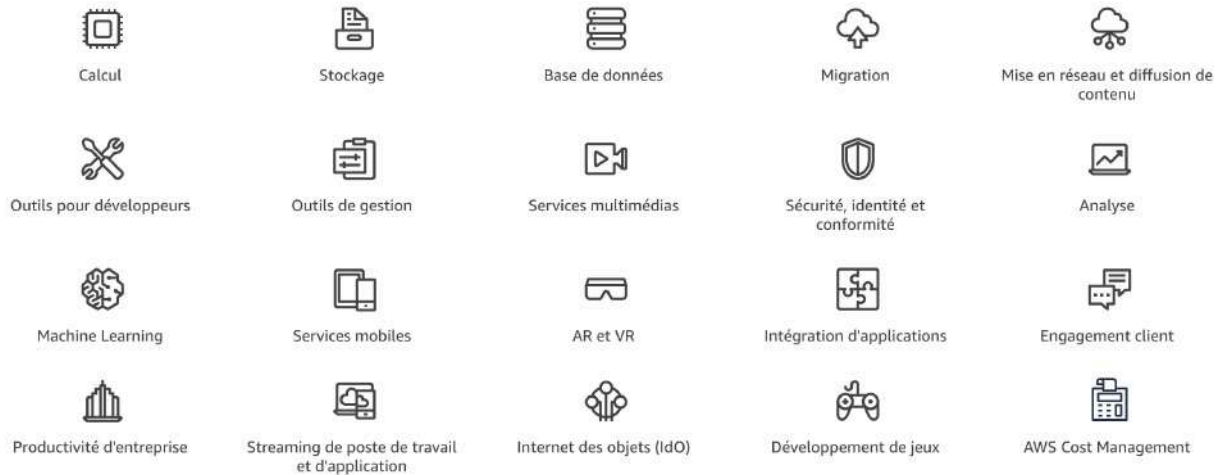


II. Core Services

Categories of the services offered by Amazon :



a. Global Infrastructure

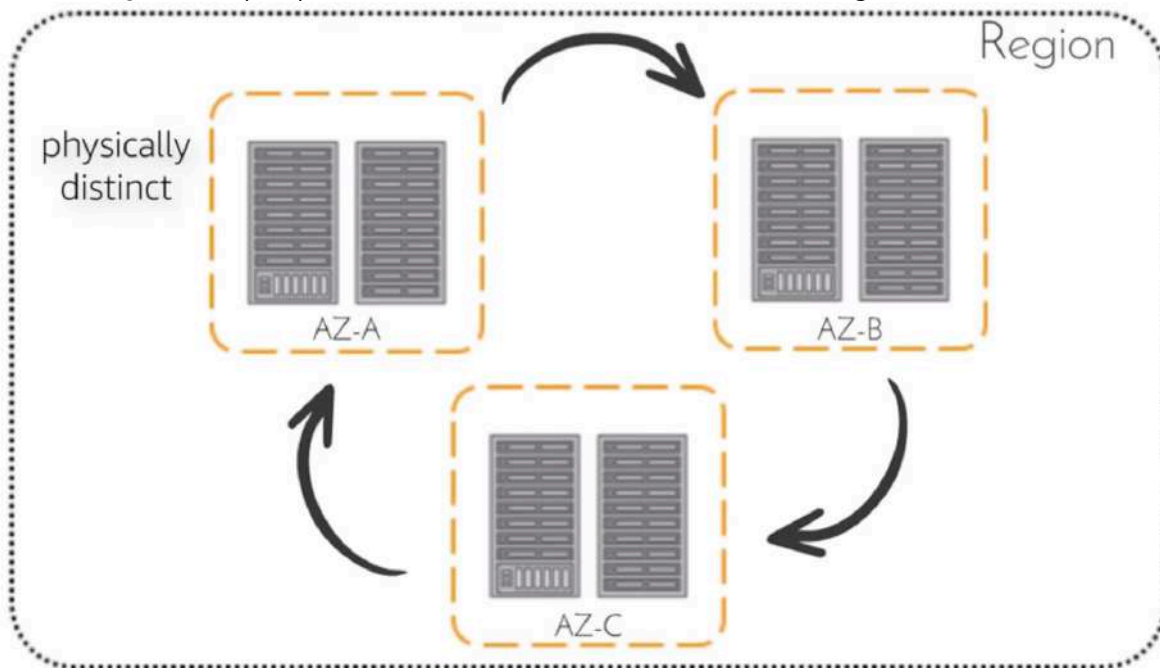
The Global Infrastructure is made of :

- Regions
- Availability Zones
- Edge location

Regions are geographic areas that host 2 or more availability zones. Offers possibility for the customer to know where the information is stored. A customer should know which region will help minimize latency of minimize costs, but also taking regulatory aspects into account.



Availability zones (AZs) are collections of data centers within a region.



When a common point of failure occurs, it does not impact all the availability zones, as they are isolated. Each AZ is a physically distinct independent infrastructure :

- physically distinct
- Each zone has its own uninterruptible power supply
- And onsite backup generators, cooling equipments and networking connectivity

The fact that they are isolated protects from failure and insures high availability. One of the best practices is to split data over multiple Availability Zones.

Edge Location

Content delivery Network (CDN) called Amazon CloudFront. Used to deliver content to customer, Requests to content are automatically sent to nearest Edge Location so that the content is delivered faster to the end user.

Edge locations are located in highly populated areas.

b. Amazon Virtual Private Cloud (VPC)

The AWS cloud offers Pay as You Go and on demand services. These services must be accessible via normal IP protocols.

Amazon VPC :

- allows you to create a private virtual network in the AWS cloud, using the same concepts as on premise networking.
- allows complete control of network configuration, with an ability to isolate and expose resources inside VPC, such as IP address, subnets and routing tables.
- offers several layers of security controls, with an ability to allow and deny specific internet and internal traffic.
- Other AWS services deploy into VPC : services inherent security built into network



Amazon VPC is an AWS foundational services and integrates with several AWS services. For instance, Amazon EC2 are deployed in Amazon VPC.

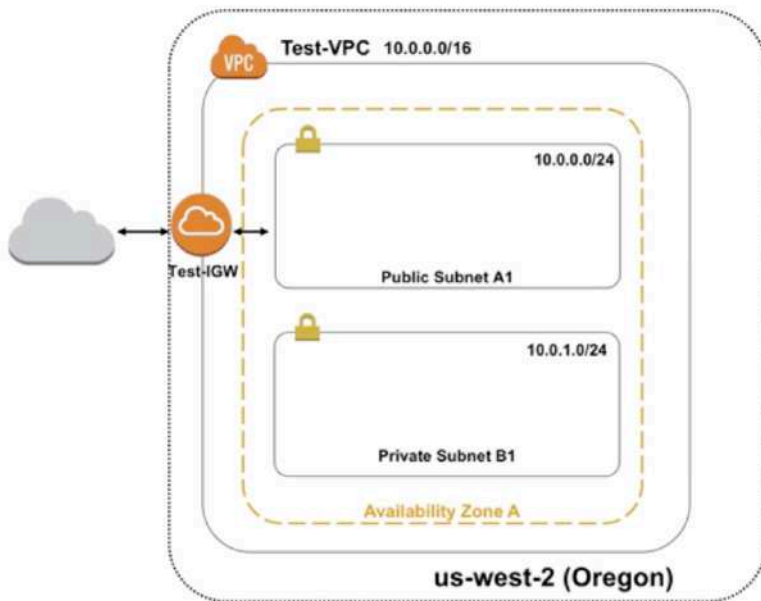
The features of Amazon VPC :

- Builds upon high availability of AWS Regions and Availability Zones (AZ). Allows you to easily take advantage of the high availability provided by AWS Cloud. Amazon VPCs live within regions and can span across multiple AZs. Each AWS account can create multiple VPCs, that can be used to segregate environments. A VPC designs an IP address base, and is then divided by subnets.
- Subnets are used to divide Amazon VPC. Allows Amazon VPC to span multiple AZs. You can create many subnets in a VPC. Best practice : Not too many Subnets for complexity purposes.
- Route tables can be used to manage traffic and the Internet. It controls traffic going out of the subnet. By default, all subnets of a VPC can communicate with each other. Subnets are usually classified as public or private. Public have direct access to the internet, and private not.
- For a subnet to be public, we need to attach an Internet Gateway (IGW) that allows access to the Internet from Amazon VPC. Then need to update the route table of the public subnet to send non-local traffic to the internet gateway. EC2 instances also need an IP address to route to an internet gateway
- NAT Gateway : Allows private subnet resources to access Internet
- Network Access Control Lists (NACL) : Control access to subnets, stateless

Example VPC :

Create a network that supports high availability and has multiple subnets.

- Since VPCs are region based, we need to select a region : us-west-2 (Oregon)
- Create the VPC, give it a name : Test-VPC
- Define an IP address base : 10.0.0.0/16. Over 65'000 IP addresses to use
- Create a Subnet named Subnet A1. Attach an IP base that contains 256 IP Addresses : 10.0.0.0/24
- Specify that the subnet will live in the Availability Zone A.
- Create a second subnet Subnet B1, that contains 512 IP addresses : 10.0.2.0/23
- Add an Internet Gateway Test-IGW
- Subnet A1 becomes a public Subnet, where non-local traffic is routed through the internet gateway.
- Subnet B1 will be a private subnet, isolated from the Internet.



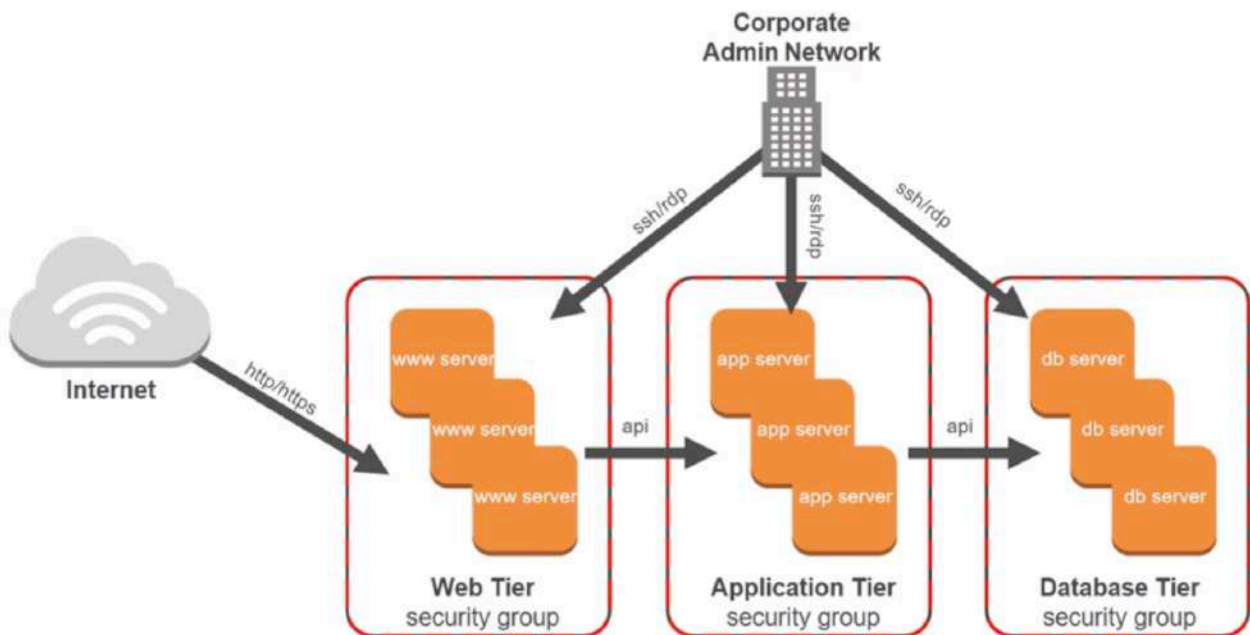
c. Security Groups

Helps secure data in the AWS Cloud. Security groups are built-in firewalls for AWS services. Allows full control on how accessible the instances are.

Lets you control what traffic to your instances you want to allow or deny. To determine who has access to your instances, configure security groups. Rules can vary from :

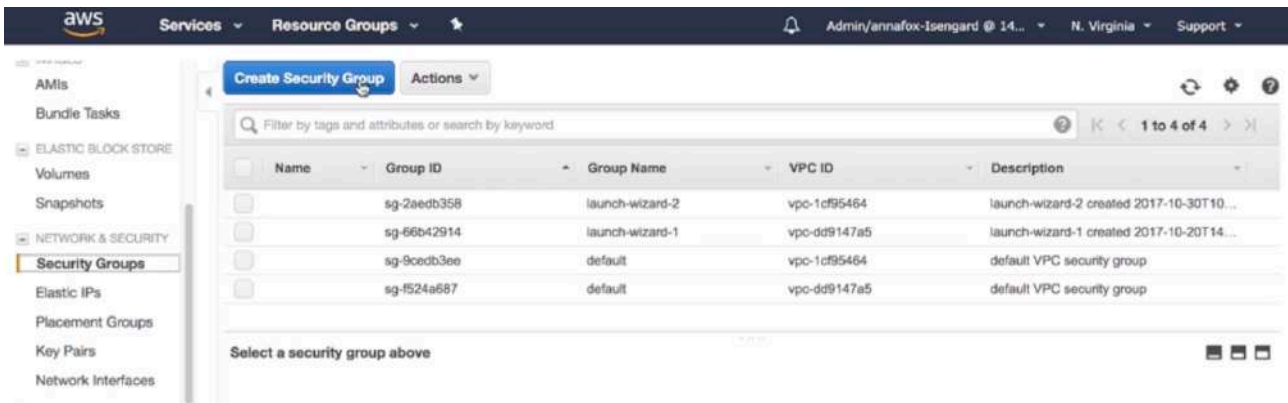
- Keeping instances completely private
- Totally public
- Or somewhere in the middle

In this example, multiple security groups to fit the multiple Tier architecture.

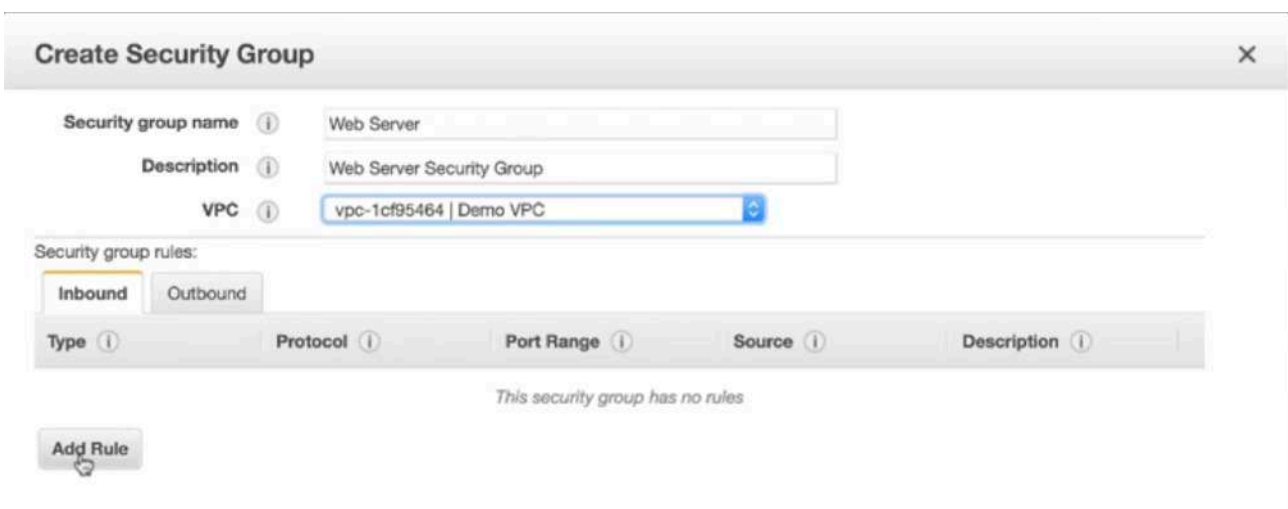


Typically here, traffic is allowed from the internet. The application Tier only allows connexion from the web tier, and the database tier only accepts traffic form the application tier. An administration rule has been created to allow connexion using SSH ports from the corporate network.

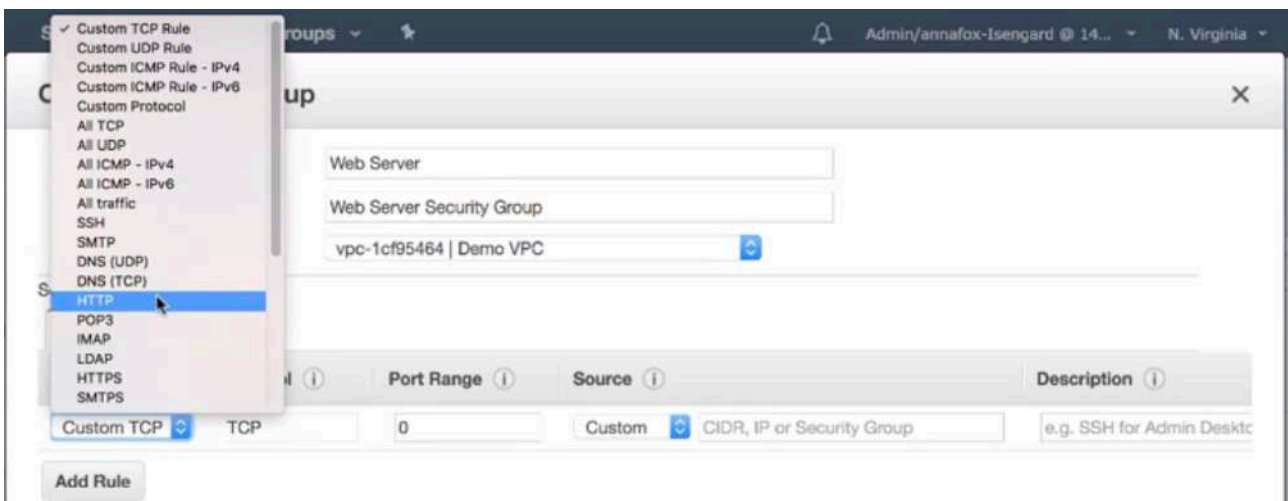
Step 1 : Create Security Group



Step 2 : Name the group, attach a VPC, and Click Add Rule



Step 3 : Add rule. By default, all inbound traffic is denied, and all outbound traffic is allowed.



Best practice : Find out what our traffic need is, and only allow this one.

d. AWS compute services

AWS computing services are :

- Flexible
- Scalable

AWS EC2 (Elastic Compute Cloud) offers a flexible configuration, not locked in into a specific configuration, and allows user to run applications at any scale. Cost-Effectively resources up and down as demand goes

AWS Lambda allows you to run code without provisioning or managing servers. You pay only for the compute time you use. Can run code for almost any application : mobile, IoT. No administration.

AWS Lightsail is made for small services such as websites and blogs. Allows you to launch virtual private servers in just minutes, and easily manage simple web and application servers. Low predictable price. Includes everything needed to start a project :

- Virtual Machine,
- SSD Base Storage,
- Data Transfer,
- DNS management
- and a static IP address.

AWS ECS is a highly scalable high performance container management that supports Docker containers and allows to easily run applications on a managed cluster of EC2 instances. No need to install, operate and manage your own cluster manager.

e. AWS EC2

What is EC2 ?

Elastic Compute Cloud.

- ✓ Application Server
- ✓ Web Server
- ✓ Database Server
- ✓ Game Server
- ✓ Mail Server
- ✓ Media Server
- ✓ Catalog Server
- ✓ File Server
- ✓ Computing Server
- ✓ Proxy Server
- ✓ Etc.

Cloud hosted compute resources, that can be increased or decreased in capacity according to the demand automatically. These servers are called EC2 instances.

These instances are :

- Pay as you go
- Broad selection of HardWare/SoftWare
- Global hosting

Process :

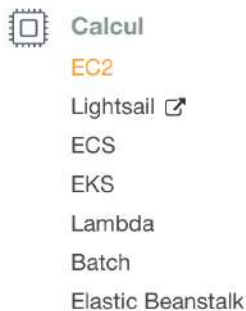
- ✓ Login to AWS Console
- ✓ Choose a region
- ✓ Launch EC2 Wizard
- ✓ Select AMI (SW)
- ✓ Select instance type (HW)
- ✓ Configure network
- ✓ Configure storage
- ✓ Configure key pairs
- ✓ Launch & connect

Demonstration :

1) Set region



2) Select EC2 Computation Service



3) Create an Instance

Créer une instance

Pour commencer à utiliser Amazon EC2, vous devez lancer un serveur virtuel, appelé instance Amazon EC2.

Lancer une instance

4) Select Amazon Linux AMI

 **Amazon Linux 2 AMI (HVM), SSD Volume Type** - ami-051707cdba246187b Sélectionner

Amazon Linux Admissible à l'offre Amazon Linux 2 est accompagné de cinq ans de support. Ce service fournit un noyau Linux 4.14 pour des performances optimales sur Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1 et les derniers packages logiciels via des extras. 64 bits (x86)

Type de périphérique racine: ebs Type de virtualisation: hvm

5) Select the instance type

Actuellement sélectionné : t2.micro (Variable ECU, 1 vCPU, 2.5 GHz, Intel Xeon Family, 1 Gio mémoire, EBS uniquement)

	Famille	Type	vCPU	Mémoire (Gio)	Stockage d'instance (Go)	Disponible en version optimisée pour EBS	Performances réseau	Prise en charge IPv6
<input type="checkbox"/>	Usage général	t2.nano	1	0.5	EBS uniquement	-	Faibles à modérées	Oui
<input checked="" type="checkbox"/>	Usage général	t2.micro <small>Admissible à l'offre gratuite</small>	1	1	EBS uniquement	-	Faibles à modérées	Oui

6) Configure the details of the Instance

Étape 3 : Configurer les détails de l'instance

Configurez l'instance en fonction de vos besoins. Vous pouvez lancer plusieurs instances à partir de la même AMI, demander des instances Spot pour bénéficier d'un tarif inférieur, attribuer un rôle de gestion d'accès à l'instance et bien d'autres choses encore.

Nombre d'instances Lancer dans un groupe Auto Scaling

Option d'achat Demander des instances Spot

Réseau Créer un nouveau VPC

Sous-réseau Créer un nouveau sous-réseau

Attribuer automatiquement l'adresse IP publique

Groupe de placement Ajoutez une instance au groupe de placement.

Capacity Reservation Create new Capacity Reservation

Rôle IAM Créer un nouveau rôle IAM

Comportement d'arrêt

Activer la protection de la résiliation Protéger contre la résiliation accidentelle

Supervision Activer la surveillance détaillée de Cloudwatch

7) Configure Storage

Type de volume	Dispositif	Instantané	Taille (Gio)	Type de volume	IOPS	Débit (Mbit/s)	Supprimer à la résiliation	Chiffré
Racine	/dev/xvda	snap-081b30dc6a058113b	12	Volume à usage général SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Non chiffré
EBS	/dev/sdb	Recherche (insensible)	16	Volume à usage général SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Non chiffré

8) Add tags to make the instances easily callable

Clié (127 caractères maximum) Valeur (255 caractères maximum) Instances Volumes

Name ec2-demo

9) Configure security group

Attribuer un groupe de sécurité: Créer un nouveau groupe de sécurité Sélectionner un groupe de sécurité existant

Nom du groupe de sécurité:

Description:

Type	Protocole	Plage de ports	Source	Description
SSH	TCP	22	Personnali: 0.0.0.0/0	Par exemple, SSH pour le bureau d
HTTP	TCP	80	Personnali: 0.0.0.0/0, ::0	Par exemple, SSH pour le bureau d

10) Launch using a new key-pair, required to connect overs SSH

✕

Sélectionnez une paire de clés existante ou créez une nouvelle paire de clés

Une paire de clés se compose d'une **clé publique** conservée par AWS et d'un **fichier de clé privée** que vous conservez. Ensemble, elles vous permettent de vous connecter à votre instance en toute sécurité. Pour les AMI Windows, le fichier de clé privée est requis pour obtenir le mot de passe utilisé pour se connecter à votre instance. Pour les AMI Linux, le fichier de clé privée vous permet d'accéder en toute sécurité à votre instance via SSH.

Remarque : La paire de clés sélectionnée sera ajoutée à l'ensemble de clés autorisé pour cette instance. En savoir plus sur [la suppression de paires de clés existantes d'une AMI publique](#).

Créer une nouvelle paire de clés

Nom de la paire de clés

Vous devez télécharger le **fichier de clé privée** (fichier *.pem) avant de pouvoir continuer. **Conservez-le dans un endroit sûr et accessible**. Vous ne pourrez pas télécharger une nouvelle fois le fichier après sa création.

11) Launch instances

Statut de lancement

✔ **Votre instance est en cours de lancement**
Le lancement d'instance suivant a été initié : i-067719a024d395437 [Afficher le journal de lancement](#)

i **Être informé des frais estimés**
Créer des alertes de facturation pour obtenir une notification par e-mail lorsque les frais estimés imputés sur votre facture AWS dépassent un montant que vous définissez (par exemple, si vous dépassez le niveau d'offre gratuite).

12) Instance is now accessible

search : i-067719a024d395437

Name	ID d'instance	Type d'instance	Zone de disponib	État de l'instanc	Contrôles des s	Statut des alarm	DNS public (IPv4)	IP pu
ec2-demo	i-067719a024d3954...	t2.micro	eu-west-3c	running	Initialisati...	Aucun	ec2-52-47-121-55.eu-w...	52.47...

13) Copy public DNS

Instance: **i-067719a024d395437 (ec2-demo)** DNS public: **ec2-52-47-121-55.eu-west-3.compute.amazonaws.com**

Description Contrôles des statuts Supervision Balises

ID d'instance	i-067719a024d395437	DNS public (IPv4)	ec2-52-47-121-55.eu-west-3.compute.amazonaws.com
État de l'instance	running	IP publique IPv4	52.47.121.55
Type d'instance	t2.micro	Adresses IP IPv6	-

14) Connect using SSH

```
MacBook-Pro-6324:~ maelfabien$ cd AWS
MacBook-Pro-6324:AWS maelfabien$ ls
ec2-demo.pem
MacBook-Pro-6324:AWS maelfabien$ chmod 400 ec2-demo.pem
MacBook-Pro-6324:AWS maelfabien$ ssh -i ec2-demo.pem ec2-user@ec2-52-47-121-55.eu-west-3.compute.amazonaws.com
```

```
__|  ( __|_ )
__|  ( __|_ /  Amazon Linux 2 AMI
__| \__|__|__|
```

```
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-34-128 ~]$
```

Where : ec2-52-47-121-55.eu-west-3.compute.amazonaws.com is the Public DNS.

15) Stop the instance when not needed anymore

Arrêter les instances [X]

Êtes-vous sûr de vouloir arrêter cette instance ?

i-067719a024d395437 (ec2-demo)

⚠ Notez que lorsque votre instance est arrêtée :

Toute donnée relative au stockage éphémère de votre instance sera perdue.

Annuler **Oui, arrêter**

f. AWS Lambda

AWS Lambda is a compute service that lets you run code without provisioning or managing service.



AWS Lambda

- Fully-Managed serverless compute
- Event-driven execution
- Sub-second metering
- Multiple languages supported

Key advantages :

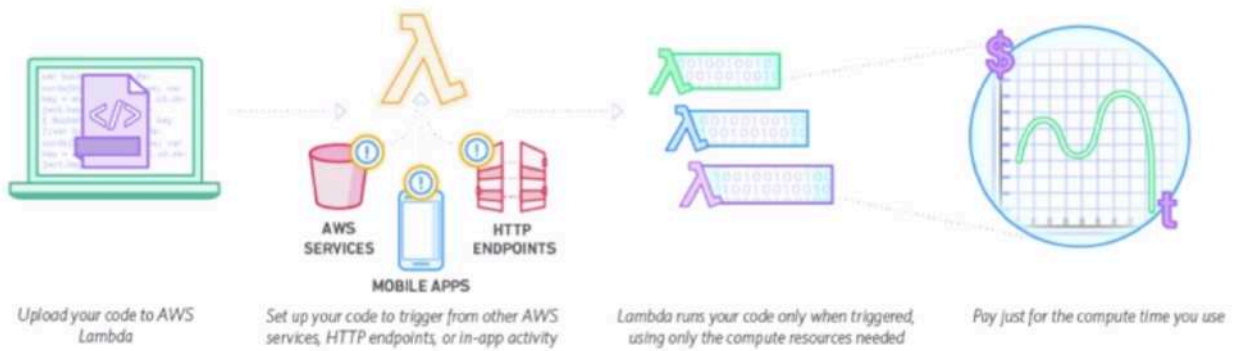
- no servers to manage : only pay for the resources used, nothing to pay when no traffic
- Continuous scaling

- Subsecond metering

Supports a variety of programming languages, including :

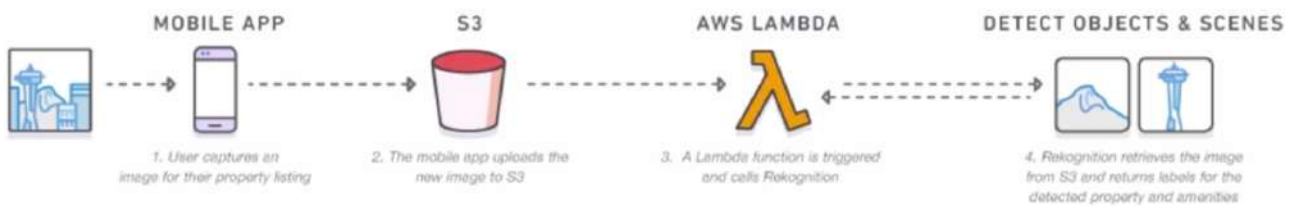
- NodeJS
- Java
- C-Sharp
- Python

Essentially used for event-driven computing

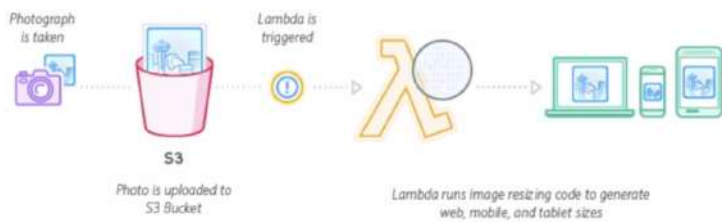


Disk space is limited to 512 Mb. Function execution is limited to a maximum of 5 minutes. Response packet size cannot exceed 6Mb.



Demonstration : Image recognition app



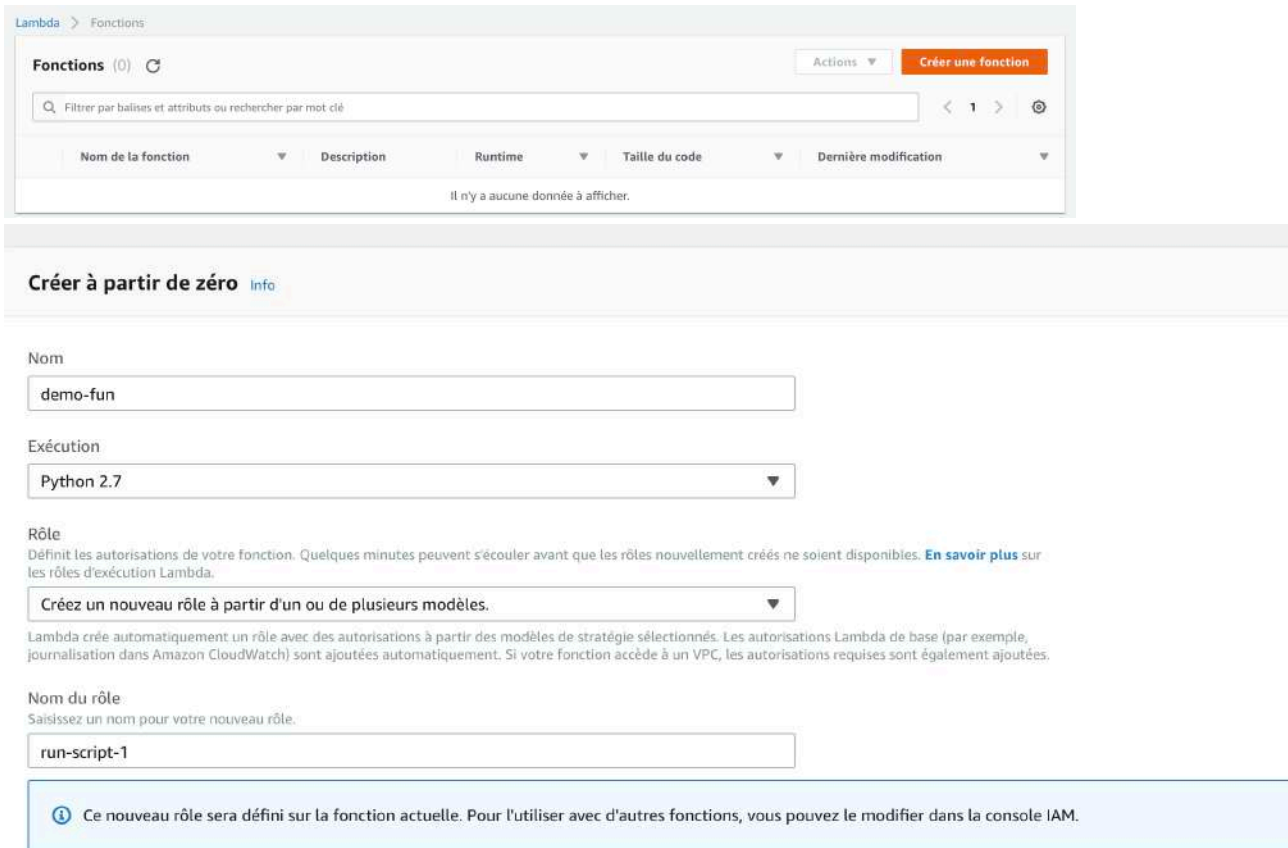
Example: Image Thumbnail Creation



1) Go to Lambda Section

-  **Calcul**
- EC2
- Lightsail 
- ECS
- EKS
- Lambda**
- Batch
- Elastic Beanstalk

2) Create a function



Lambda > Fonctions

Fonctions (0) Actions Créer une fonction

Filterer par balises et attributs ou rechercher par mot clé

Nom de la fonction Description Runtime Taille du code Dernière modification

Il n'y a aucune donnée à afficher.

Créer à partir de zéro [Info](#)

Nom

demo-fun

Exécution

Python 2.7

Rôle

Définit les autorisations de votre fonction. Quelques minutes peuvent s'écouler avant que les rôles nouvellement créés ne soient disponibles. [En savoir plus](#) sur les rôles d'exécution Lambda.

Créer un nouveau rôle à partir d'un ou de plusieurs modèles.

Lambda crée automatiquement un rôle avec des autorisations à partir des modèles de stratégie sélectionnés. Les autorisations Lambda de base (par exemple, journalisation dans Amazon CloudWatch) sont ajoutées automatiquement. Si votre fonction accède à un VPC, les autorisations requises sont également ajoutées.

Nom du rôle

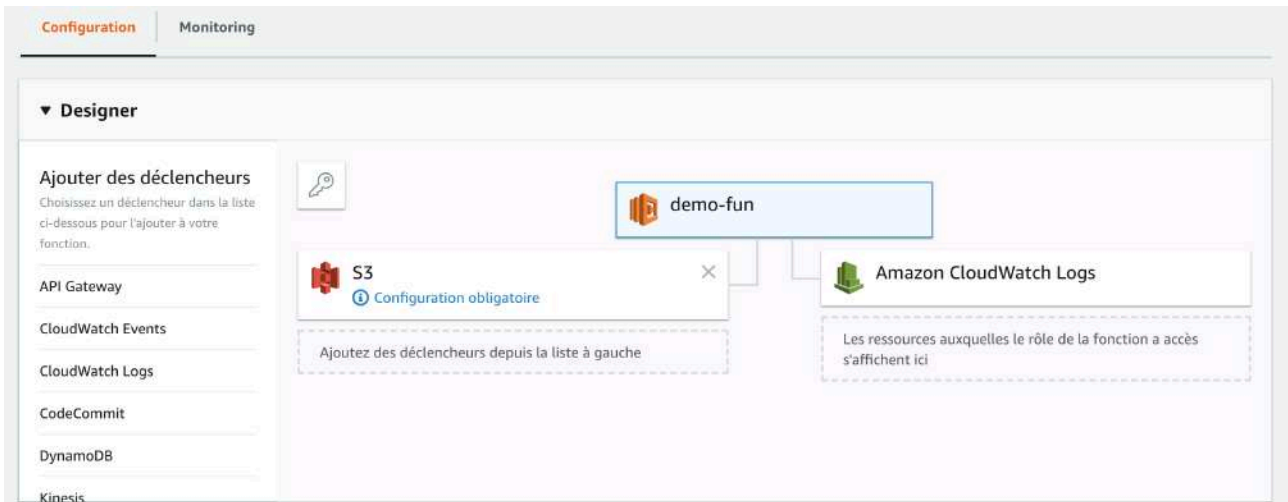
Saisissez un nom pour votre nouveau rôle.

run-script-1

! Ce nouveau rôle sera défini sur la fonction actuelle. Pour l'utiliser avec d'autres fonctions, vous pouvez le modifier dans la console IAM.

Modèles de stratégie

3) Design the trigger



Configuration Monitoring

▼ Designer

Ajouter des déclencheurs

Choisissez un déclencheur dans la liste ci-dessous pour l'ajouter à votre fonction.

- API Gateway
- CloudWatch Events
- CloudWatch Logs
- CodeCommit
- DynamoDB
- Kinesis

demo-fun

S3 Configuration obligatoire

Ajoutez des déclencheurs depuis la liste à gauche

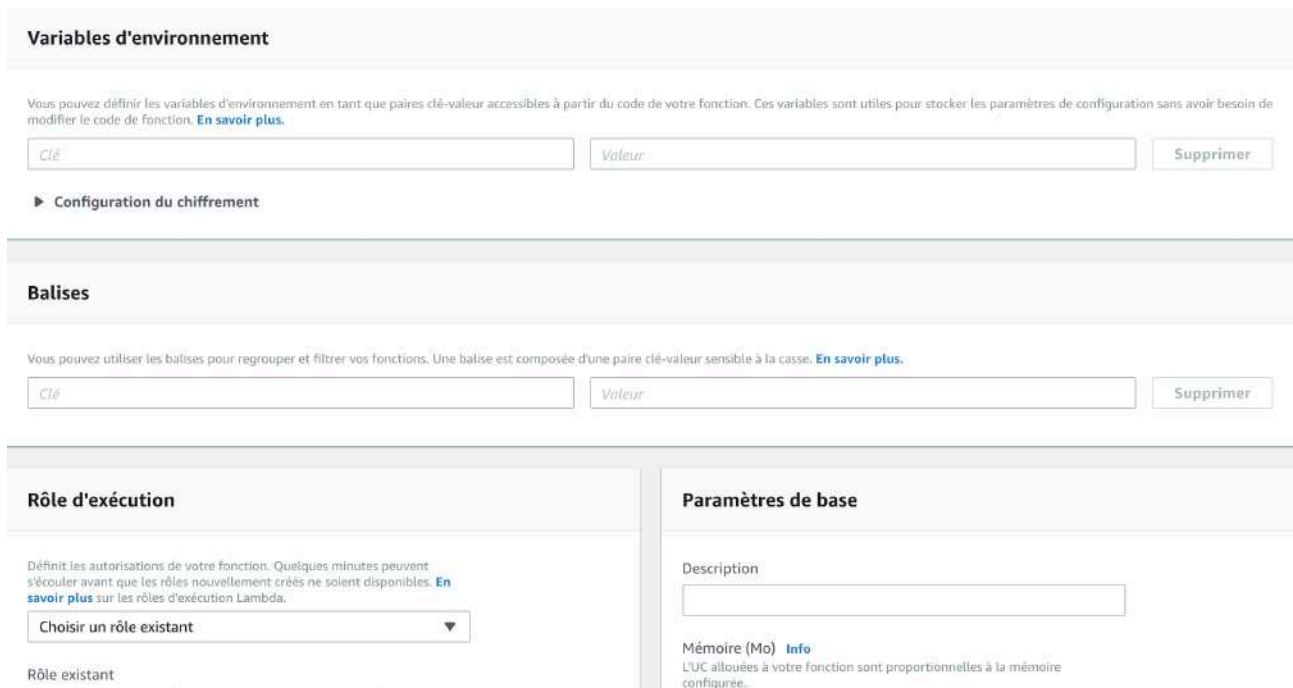
Amazon CloudWatch Logs

Les ressources auxquelles le rôle de la fonction a accès s'affichent ici

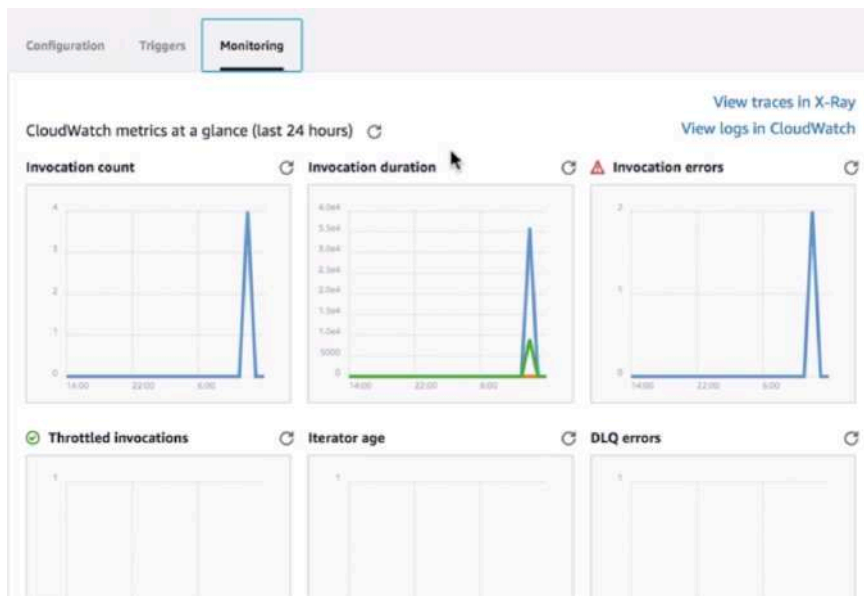
4) Add code of the function



5) Set options



6) Monitor the use of the lambda function



AWS Lambda use cases include :

- automated backups
- Processing objects uploaded to S3
- Event-driven log analysis
- Event-driven transformations
- IoT
- Operating several websites
- Real-Time stream processing
- ...

g. AWS Elastic Beanstalk

AWS Elastic Beanstalk is a Platform as a Service that allows a quick deployment of a given code into the Cloud. The web application package comes in on the top of the AWS Elastic Beanstalk framework. It highly reduces the management complexity.

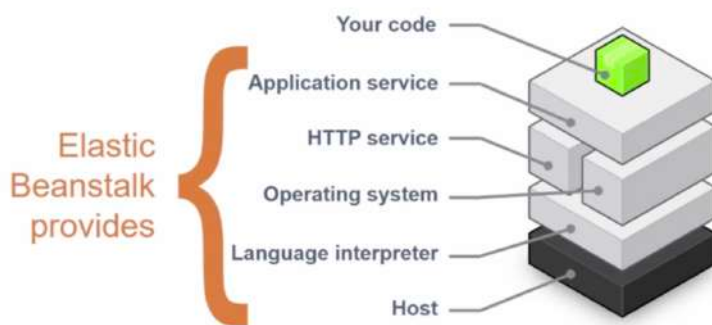
Control can be kept by user, by allowing :

- To choose instance type
- To choose the database
- And to set and adjust autoscaling
- Update application
- Access server log files
- Enable HTTPS on load balancer

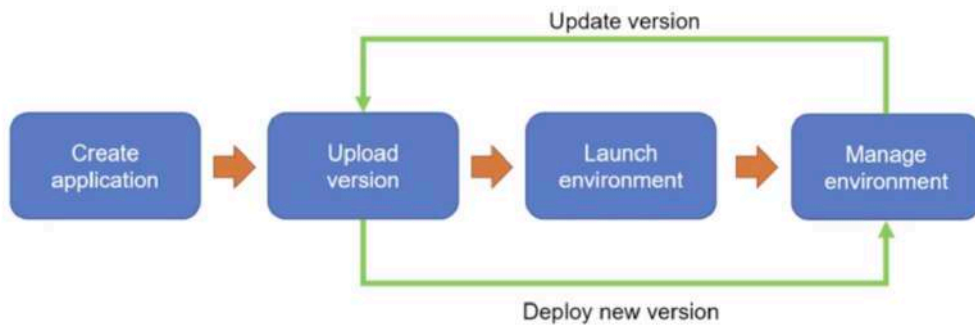
AWS Elastic Beanstalk supports a large range of platforms :

- Package builder
- Go
- Python
- PHP
- Ruby
- NodeJS
- Java SE
- Docker
- .NET
- ...

Components of AWS Elastic beanstalk :



Process of deployment :



Demonstration :

1) Connect to Elastic Beanstalk section

-  **Calcul**
- EC2
- Lightsail [↗](#)
- ECS
- EKS
- Lambda
- Batch
- Elastic Beanstalk**

2) Create a demo app

Créer une nouvelle application ×

Nom de l'application
100 caractères maximum, sans barre oblique (/).

Description
200 caractères maximum.

[Annuler](#) [Créer](#)

3) Create a web server environment

Sélectionner un niveau d'environnement

AWS Elastic Beanstalk possède deux types de niveaux d'environnement pour prendre en charge différents types d'applications web. Les serveurs web sont des applications standard qui écoutent puis traitent des demandes HTTP, généralement sur le port 80. Les applications de travail sont des applications spécialisées qui possèdent une tâche de traitement en arrière-plan qui écoute les messages sur une file d'attente Amazon SQS. Les applications de travail publient ces messages sur l'application grâce à HTTP.

Environnement de serveur web

Exécuter un site web, une application web ou une API web qui propose des demandes HTTP.
[En savoir plus](#)

Environnement de travail

Exécuter une application de travail qui traite des charges de longue durée à la demande ou réalise des tâches selon un calendrier.
[En savoir plus](#)

4) Configure the web server and upload the code

Nom de l'environnement

Domaine

Description

Configuration de base

Plateforme Plateforme préconfigurée
Plateformes publiées et gérées par AWS Elastic Beanstalk.

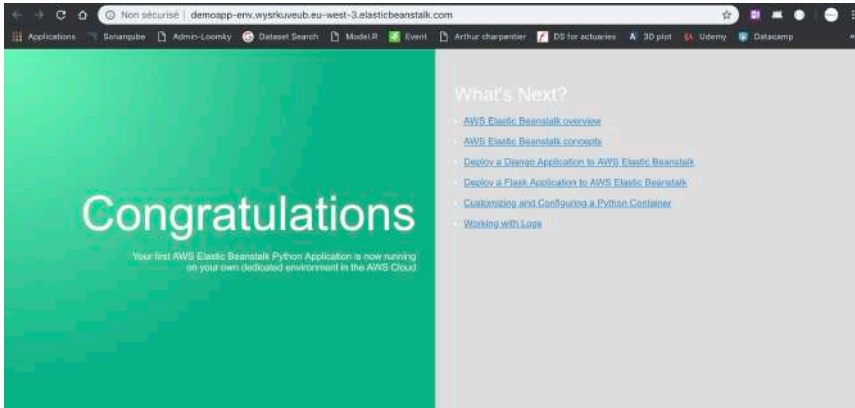
Plateforme personnalisée
Plateformes que vous avez créées et dont vous êtes propriétaire. [En savoir plus](#)

Code de l'application Exemple d'application
Démarrer immédiatement avec l'exemple de code.

Version existante
Versions de l'application que vous avez chargées pour **demo-app**.

Charger votre code
Charger un groupe source depuis l'ordinateur ou en copier un à partir d'Amazon S3.
 ZIP ou WAR

5) The app is available online



6) Resolve environment if not needed anymore

Valider la résiliation ✕

Résilier de façon permanente **DemoApp-env** ? Cette action ne peut pas être annulée.

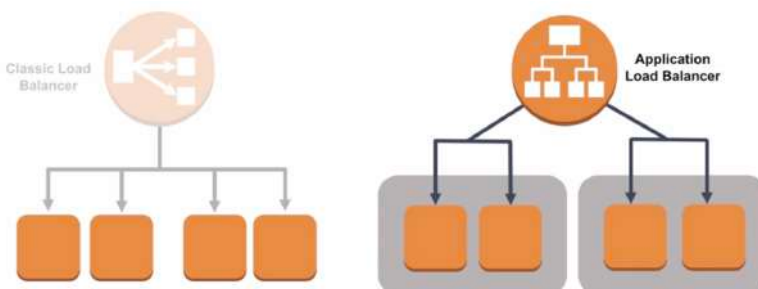
- Niveau : Serveur web
- Plateforme : Python 3.6 running on 64bit Amazon Linux/2.7.6
- Version : Sample Application
- Dernière modification : 24-11-2018 17:35:40 UTC+0100

La poursuite de cette action entraîne ce qui suit :

- *DemoApp-env.wysrkuveub.eu-west-3.elasticbeanstalk.com* sera lancé.
- Toute ressource supplémentaire associée à votre environnement Elastic Beanstalk **sera détruite**.

Entrez le nom de l'environnement pour confirmer :

g. AWS Application Load Balancer



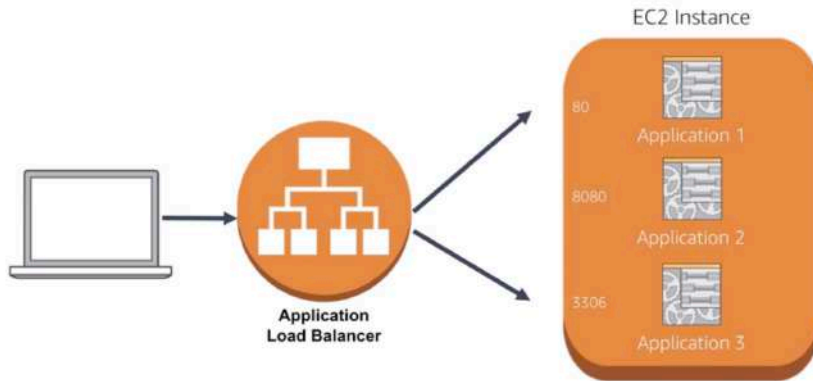
Pretty similar to a Classic Load Balancer, but it adds some new features :

- additional supported request protocols

- Enhanced metrics and access logs
- More target health checks

Use Cases :

- Ability to use containers to host micro-services and wrap to those applications from a single lead balancer
- Application load balancer allows you to wrap different requests to a single instance, but differ the path based on the port
- If we have different containers listening on different ports, you can set up routing rules to distribute traffic to only the desired backend applications

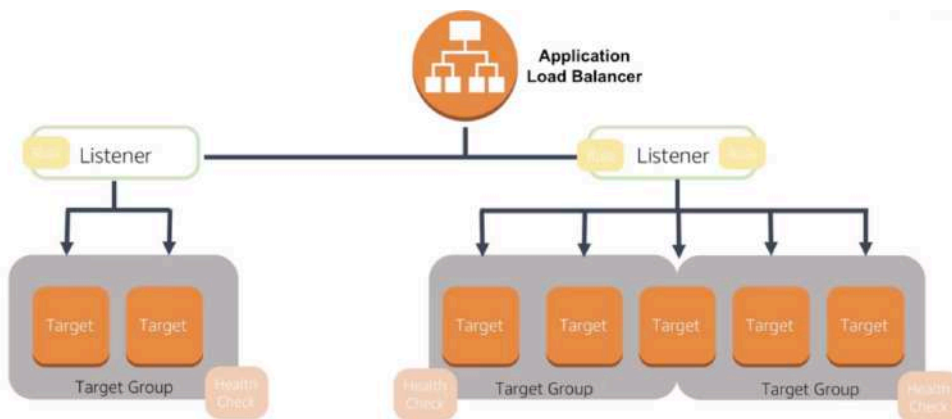


Key Concepts :

Concept	Description
Listeners	A listener is a process that checks for connection requests, using the protocol and port that you configure. The rules that you define for a listener determine how the load balancer routes requests to the targets in one or more target groups.
Target	A target is a destination for traffic based on the established listener rules.
Target Group	Each target group routes requests to one or more registered targets using the protocol and port number specified. A target can be registered with multiple target groups. Health checks can be configured on a per target group basis.

Use Case :

When configuring the listeners for the load balancers, we can create rules in order to direct how the order received by the load balancer will be routed to the backend targets. To register those targets to the load balancer and configure the health check the load balancer will use for the target, we can create target groups. Targets can be members of multiple target groups

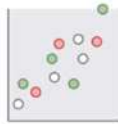


Enhanced and added features :



Supported Protocols

HTTP, HTTPS, HTTP/2, and WebSockets



CloudWatch Metrics

Additional load balance metrics and Target Group metric dimension



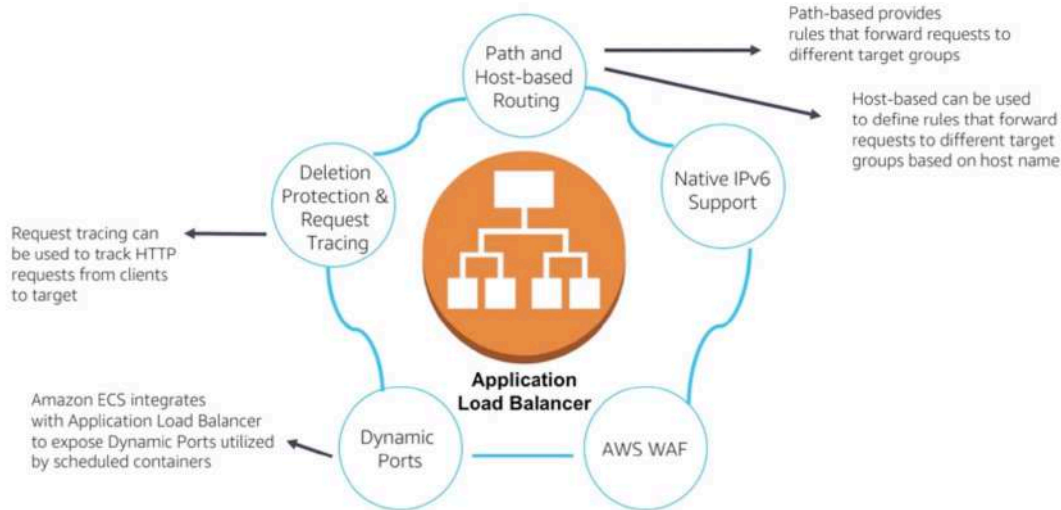
Access Logs

Ability to see connection details for WebSocket connections



Health Checks

Insight into target and application health at more granular level



The aim is to :

- launch the application from the Load Balancer
- Configure routing
- Register targets
- Verify operations of the application load balancer

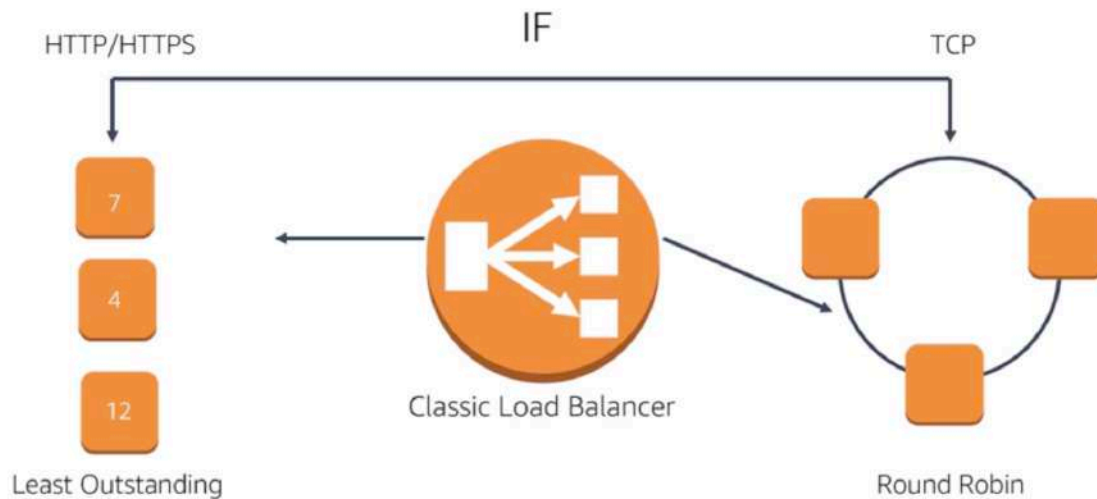
Set Health Check ping path to index.html.

h. AWS Elastic Load Balancer

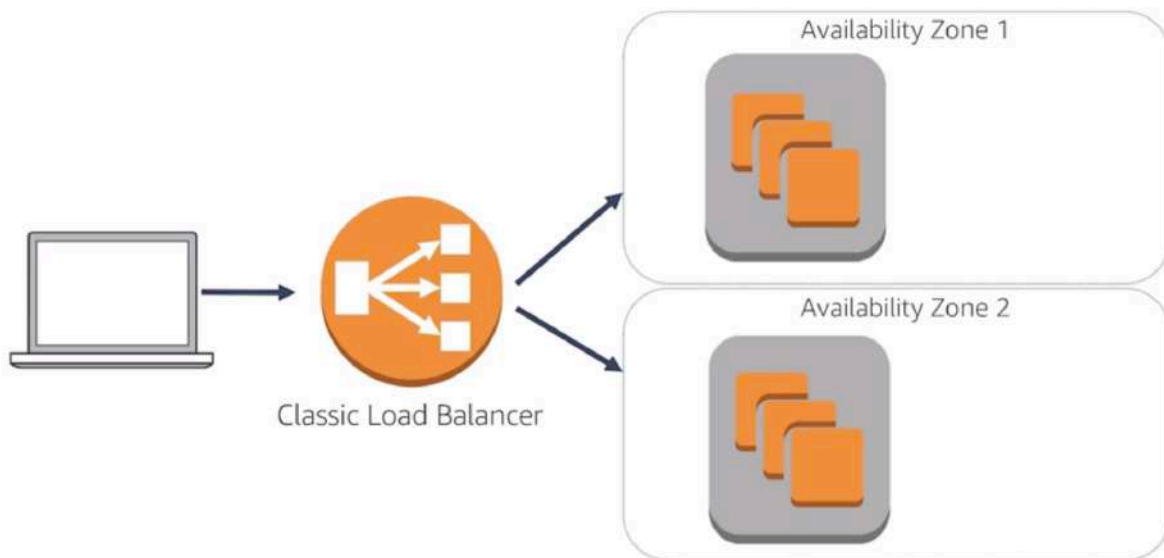
The Classic Load Balancer is a distributed software load balancing service that enables the use of many useful features packaged in a managed solution :

- Access web servers through a single exposed point of access
- Decoupling an application environment using both public facing and internal load balancers
- Providing high availability and fault tolerance with ability to distribute traffic across multiple availability zones.
- Increase elasticity and scalability with minimum overhead.

Ability of load balancer to distribute traffic depends on the type of requests we are distributing. If we use HTTP(S) requests, the load balancer will use Least Outstanding requests for the backend instances. If we are processing TCP requests, the elastic load balancer uses a Round Robin for these requests.



When distributing traffic among several availability zones, load balancer also helps. If we create the load balancer within the management console, this feature is enabled by default. But if we launch it from the command line tools or an SDK, this will need to be enabled as a secondary process.



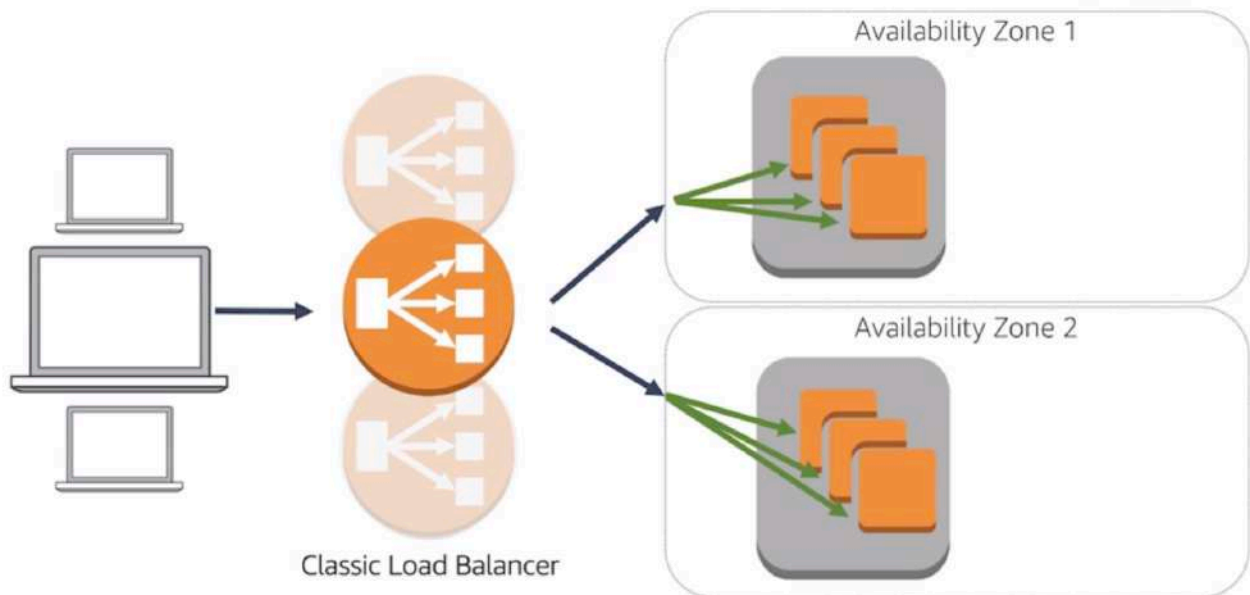
The elastic load balancer provides a single point of access to the backend instances. Need to set up an alias record to point our domain CName to the endpoint for the elastic load balancer.

If we would like to use cookies for our application, the elastic load balancer provides the feature of a sticky session. Sticky sessions allow us to bind a user session, for the duration of that session, and is set depending on whether we want to use duration based cookies or application controlled sticky sessions.

For monitoring, the application load balancer provides many metrics by default, allowing us to see :

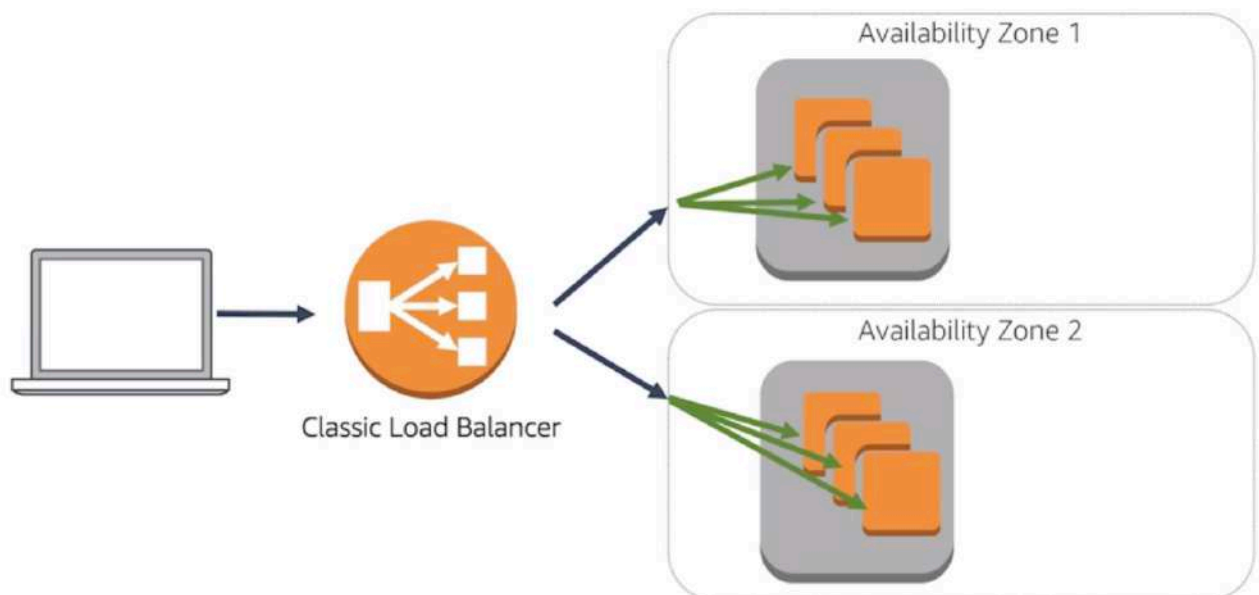
- HTTP responses
- Number of healthy and unhealthy EC2 hosts behind the load balancer (typically using ping requests).
- Filter metrics based on availability zones of the backend instances, or based on the load balancer that we were using.

Within our VPC, the load balancer helps with our scalability by providing a multi-zone load balancing, enabling us to distribute traffic across multiple availability zones. Additionally, the load balancer will scale according to the traffic pattern that it sees.

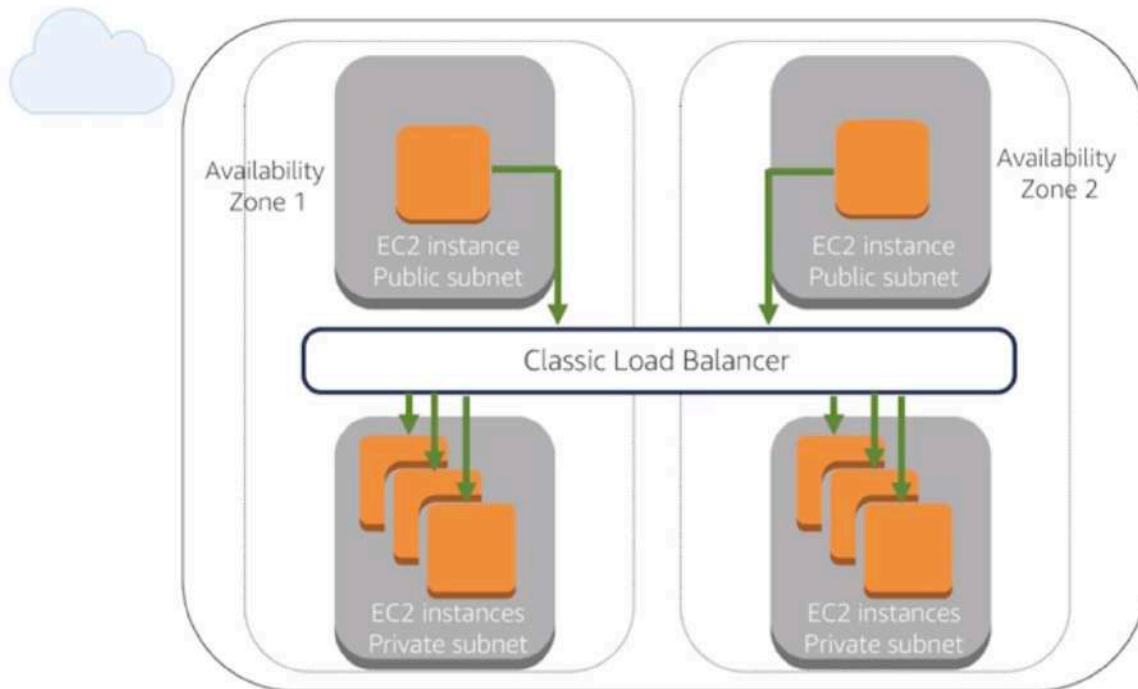


With the classic load balancer, we can create several types of load balancers :

- Internet-facing load balancer : gives us a publicly resolvable DNS name, that still allows cross zone balancing and allows you to route request to our backend instances from our single end point of our load balancer.

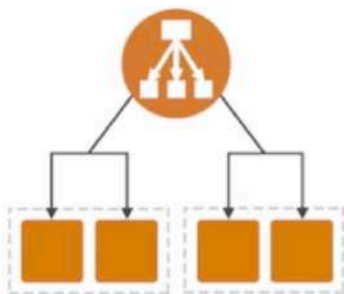


- Internal load balancer : has a DNS name that resolves only to private nodes so it can only be accessed through the VPC. This provides a decoupling of our infrastructure within the VPC and allows for the scalability of both frontend and backend instances while the load balancer handles its own scaling.



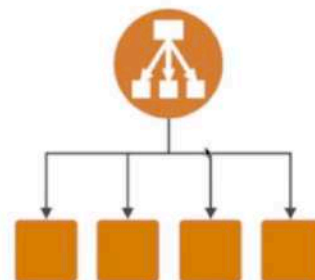
Application Load Balancer

Preferred for HTTP/HTTPS



An Application Load Balancer makes routing decisions at the application layer (HTTP/HTTPS), supports path-based routing, and can route requests to one or more ports on each EC2 instance or container instance in your VPC.

Classic Load Balancer



A Classic Load Balancer makes routing decisions at either the transport layer (TCP/SSL) or the application layer (HTTP/HTTPS), and supports either EC2-Classic or a VPC.

Set Health Check ping path to index.php.

When setting the advanced details of Health Checks :

Advanced Details

Response Timeout	<input type="text" value="5"/>	seconds
Interval	<input type="text" value="10"/>	seconds
Unhealthy threshold	<input type="text" value="2"/>	
Healthy threshold	<input type="text" value="10"/>	

- response timeout is the amount of time elapsed before we consider a timeout
- Interval is how often we do health checks
- Unhealthy threshold is the number of unhealthy responses we get before considering an unhealthy check

Steps :

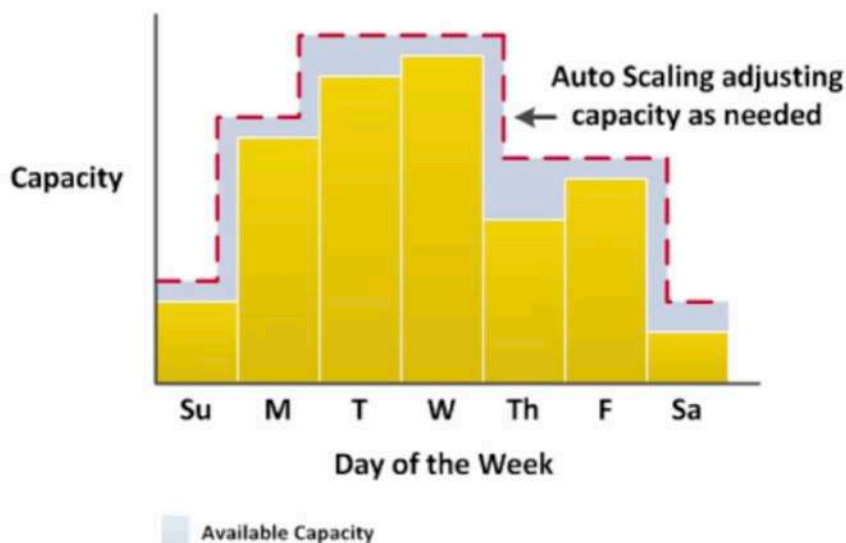
- Define Load balancer
- Assign security group
- Configure security settings (skip if not SSL)
- Configure Health Checks
- Add EC2 instances
- Add tags
- Review

i. Auto-Scaling

Autoscaling helps you making sure that you have the correct number of EC2 instances available for the load of your application.

When running an application, it is important to use CloudWatch in order to get a full view. However, Cloud Watch does not provide autoscaling.

Autoscaling avoids over and under capacity.



Autoscaling answers 2 critical questions :

- how can I ensure that my workload has enough EC2 to meet fluctuating performance requirements ? -> Scaling
- How can I automate EC2 resource provisioning to occur on-demand ? -> Automation

How do we auto-scale ?

- Launch configuration
- Create auto-scaling group
- Define at least one Auto-scaling policy

The launch configuration is about defining what will be launched by autoscaling :

- which Amazon Machine to use
- Instance types
- Security groups

- Roles

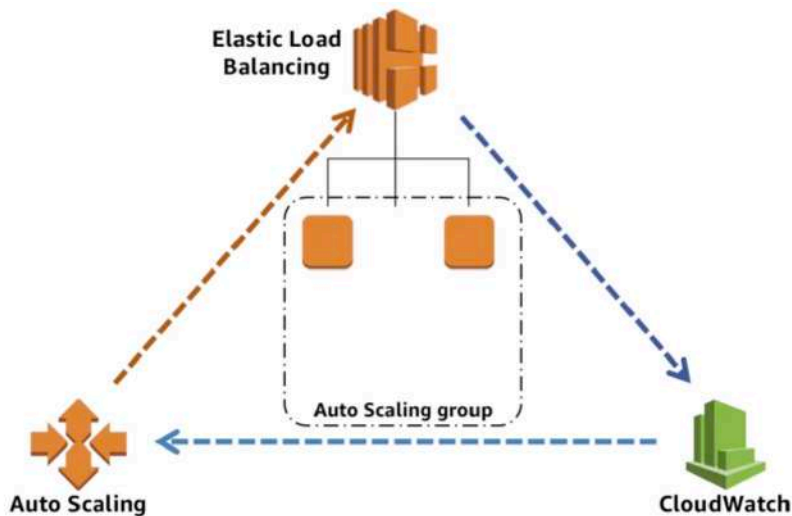
The autoscaling group is about defining where the auto scaling takes place and the boundaries for the deployment :

- Which VPC and which load balancers to interact with
- Minimum instances and maximum instances
- Desired capacity

The autoscaling policy specifies when to launch or terminated EC2 instances.

- we can schedule autoscaling (manually, e.g every Thursday a 3PM)
 - Or create conditions that define thresholds to trigger adding or removing instances
- Best practice : 1 condition on when to scale out (grow), and 1 for when to scale in (shrink).

How does dynamic autoscaling work ? Create CloudWatch alarms based on performance information on our EC2 instances. If a threshold is reached, creates an alarm, and scaling occurs.



Typically, these rules look like this :

Also set a rule when CPU utilization goes below 20% for more than 5 minutes for example. We can also specify the rule as a utilization target, and apply boundaries in terms of maximum and minimum instances.

j. Amazon Elastic Block Store (EBS)

EBS volumes are used when storing data on disk from AWS instances.

These volumes can be hard disk or SSD, and are paid according to use. Whenever we don't need volume anymore, we can just delete it.

Data stored are automatically replicated across multiple servers within the availability zone to offer high availability.

Provides ability to do point-in-time snapshots of our volumes. We can re-create a volume from a given snapshot.

A given snapshot can be encrypted and shared. We can have encrypted volumes, where encryption occurs on the EC2 side.

Steps :

- launch EC2 instance
- Create volume
- Choose AZ
- Attach volume to EC2 instance (Actions, Attach Volume)
- Log inside the instance (copy the SSH connection command line)

```
~ > Downloads ssh -i "default-lab-key.pem" ec2-user@ec2-34-226-122-211.compute-1.amazonaws.com

Last login: Fri Nov  3 14:11:26 2017 from 152.179.243.2

  __|  __|_  )
 _| (    /   Amazon Linux AMI
---|\---|---|

https://aws.amazon.com/amazon-linux-ami/2017.09-release-notes/
1 package(s) needed for security, out of 6 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-56-153 ~]$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
xvda        202:0    0   8G  0 disk
└─xvda1     202:1    0   8G  0 part /
xvdb        202:16   0  25G  0 disk
```

Then, we can create a file system on the new instance volume :

```
[ec2-user@ip-172-31-56-153 ~]$ mke2fs /dev/xvdb
mke2fs 1.42.12 (29-Aug-2014)
Could not open /dev/xvdb: Permission denied
[ec2-user@ip-172-31-56-153 ~]$ sudo !!
sudo mke2fs /dev/xvdb
mke2fs 1.42.12 (29-Aug-2014)
Creating filesystem with 6553600 4k blocks and 1638400 inodes
Filesystem UUID: d3818f69-3fea-4665-a00f-9dd4e123b518
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
    4096000

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

Then we need to mount this volume :

```
[ec2-user@ip-172-31-56-153 ~]$ mount /dev/xvdb /mnt
mount: only root can do that
[ec2-user@ip-172-31-56-153 ~]$ sudo !!
sudo mount /dev/xvdb /mnt
[ec2-user@ip-172-31-56-153 ~]$ sudo -s
[root@ip-172-31-56-153 ec2-user]# cd /mnt
[root@ip-172-31-56-153 mnt]# ls
```

Unmount command will unmount the volume. If a volume is available but not used, we can attach it to another instance within the same availability zone.