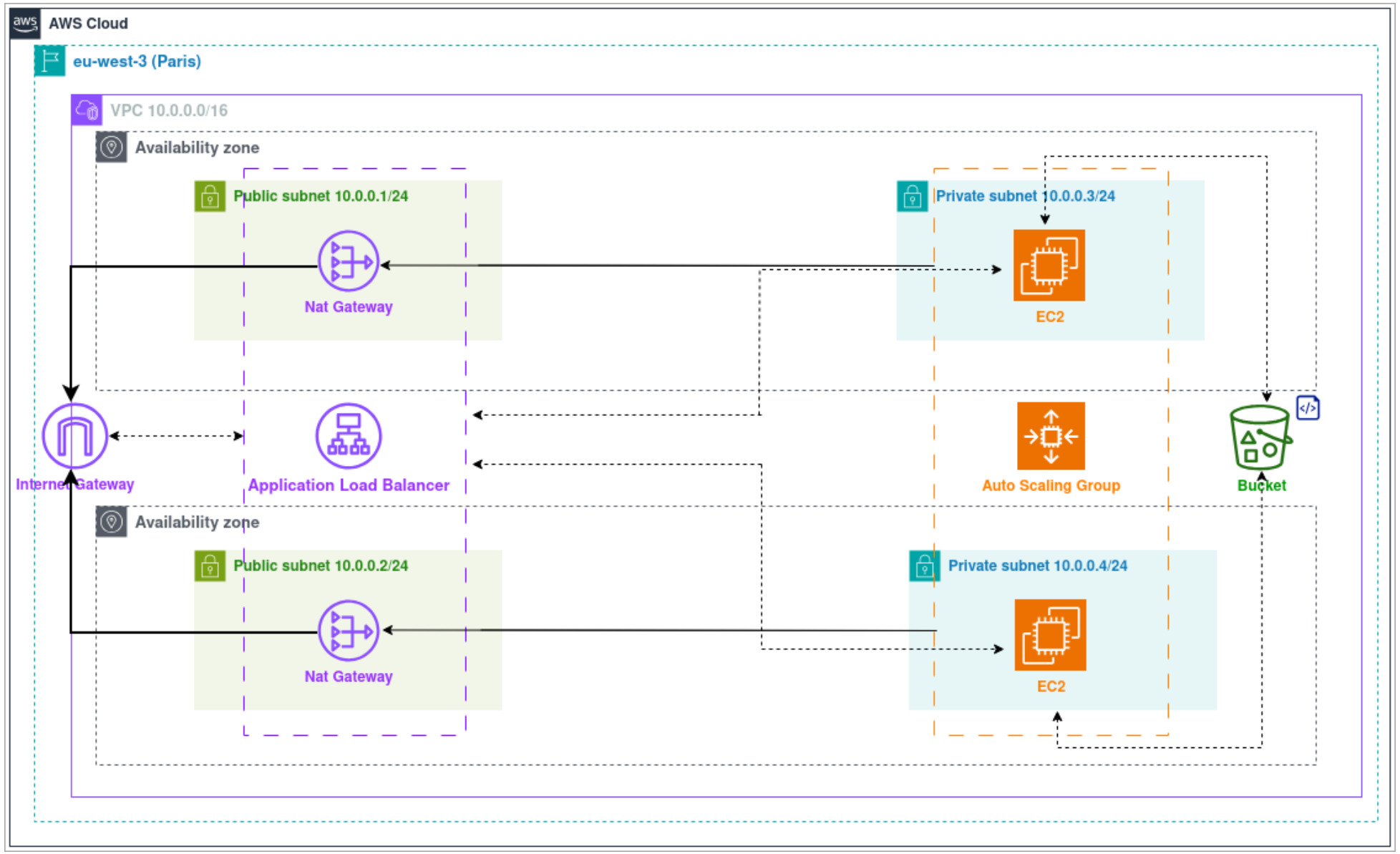
# Déploiement d'une infrastructure AWS hautement disponible

Clément Boyer – Simon Caumeil – Noé Labbe – Djibril Yapi – Maxime Chantepie

# Projet AWS - Schéma de l'infrastructure



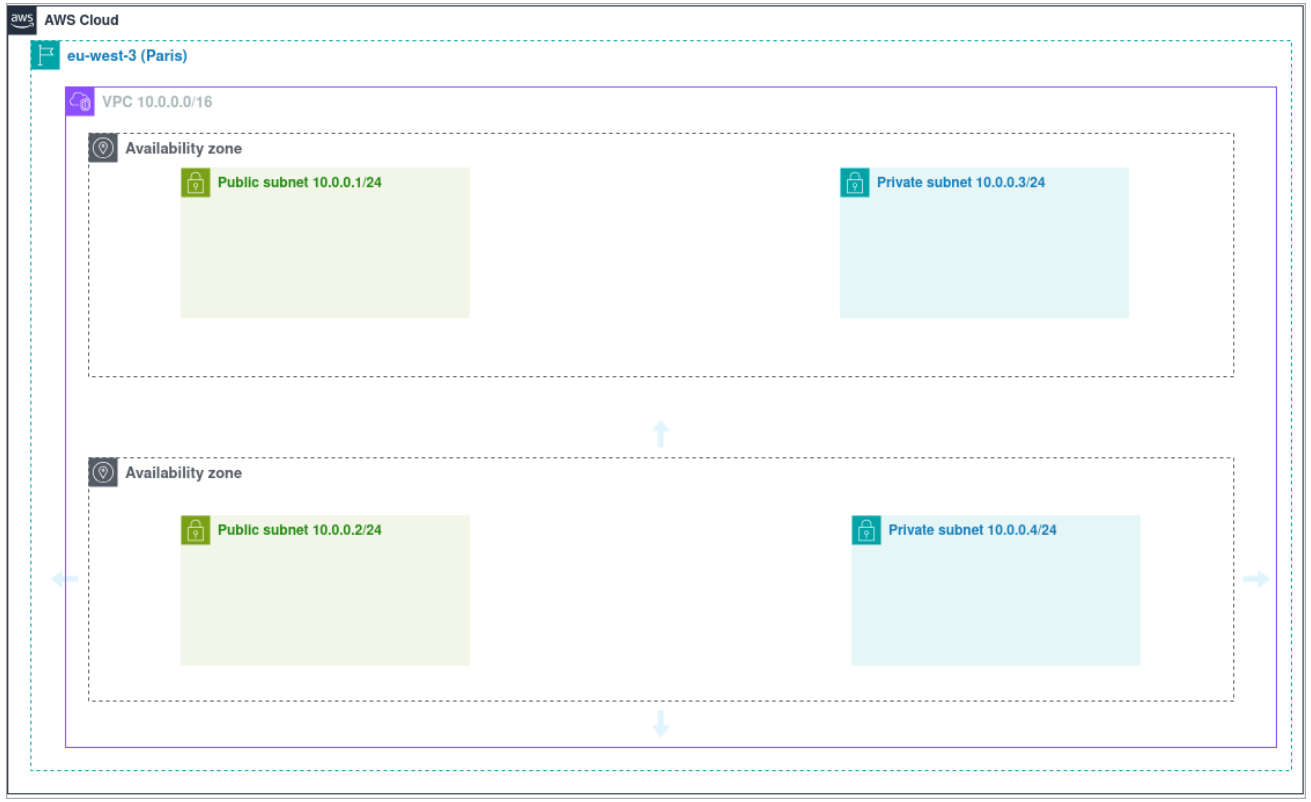
# Les Composants Réseaux

# Architecture réseau du VPC : Les sous-réseaux

- **1** Virtual Private Cloud **(VPC)**
- **2 Sous-réseaux publics** : Les ressources AWS dans ces sous-réseaux sont directement accessibles depuis internet.
- **2 Sous-réseaux privées** : Les ressources dans ces sous-réseaux sont isolées d'internet.

Chaque paire de sous-réseaux public/privé est située dans une AZ différentes.

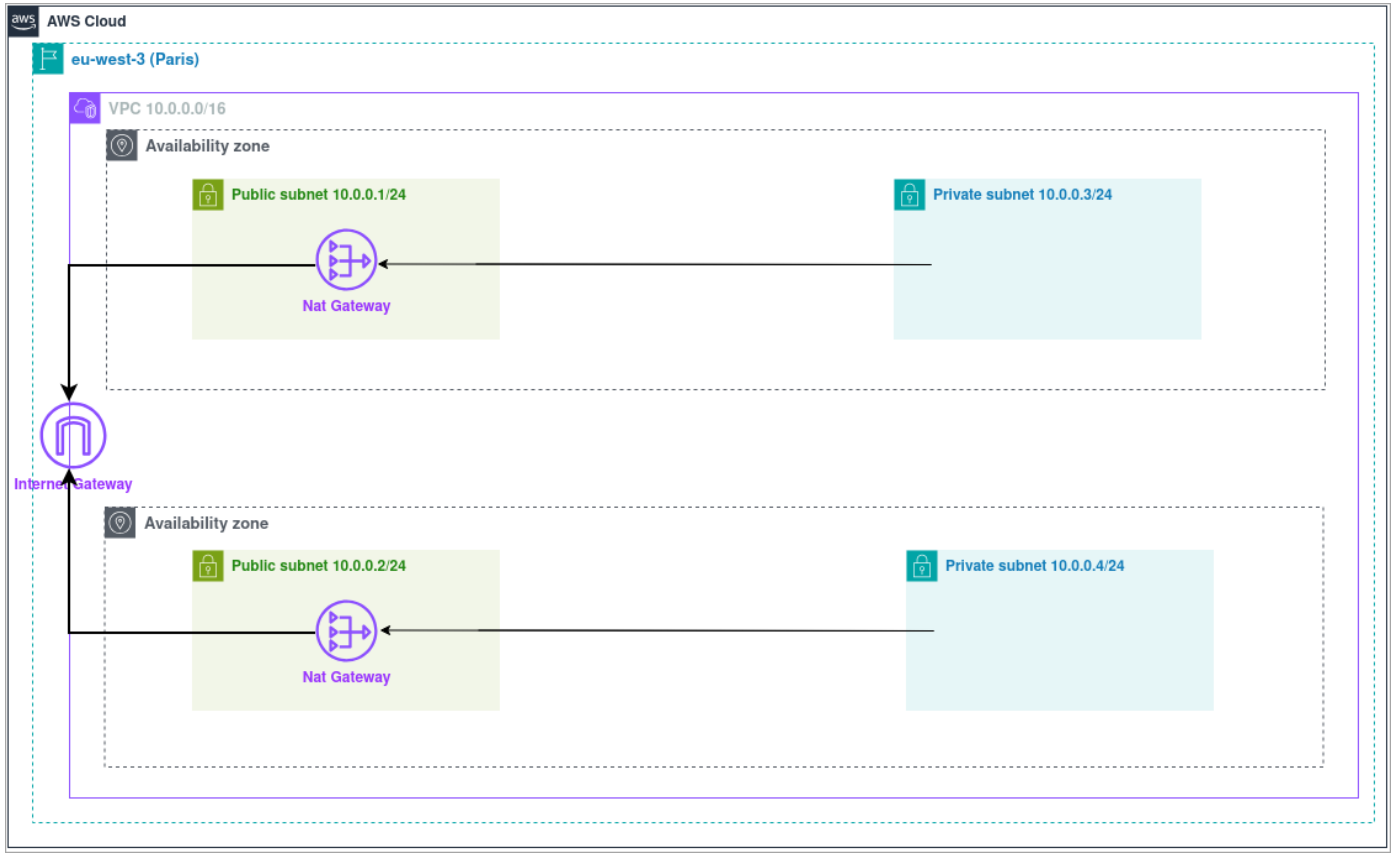
Cela permet de rendre l'infrastructure plus résiliente en cas d'indisponibilité d'une AZ.



# Architecture réseau du VPC : La connectivité

Pour pouvoir communiquer avec l'extérieur le VPC se doit d'avoir les éléments suivants :

- **1 Internet Gateway** permettant de communiquer avec internet.
- **2 NAT Gateway** permettant de donner accès aux sous-réseaux privés à internet.

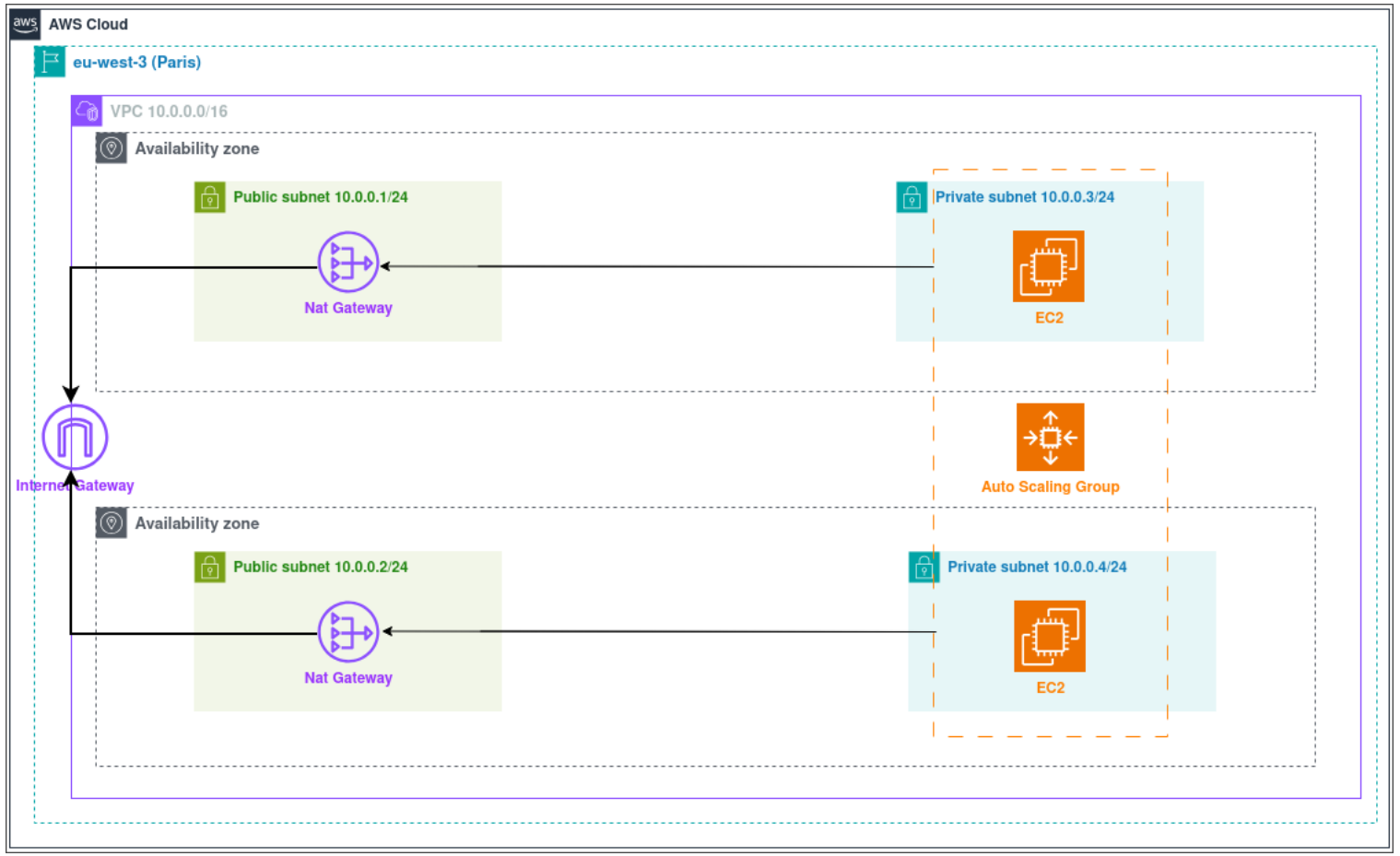


Comment est assurée la scalabilité de l’architecture ?

# Architecture système du VPC : L'AutoScalingGroup

L'AutoScaling est un des **léviers les plus intéressants** du Cloud Computing.

Il est possible de **scaler (out/in)** rapidement pour répondre au besoin des applications tout en maitrisant notre budget.



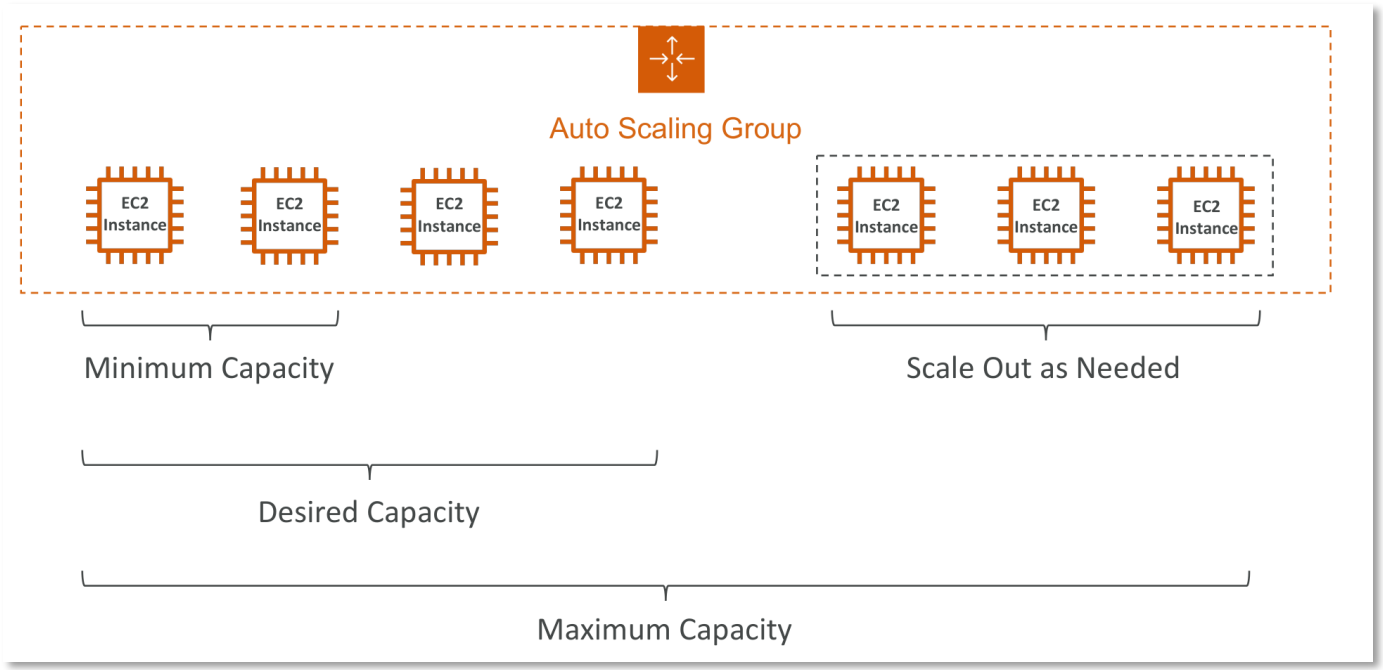
# AWS AutoScalingGroup : les différentes stratégies

Il est possible de définir différentes stratégies pour répondre à la montée en charge mais toutes doivent s'assurer de posséder les critères suivants :

**Minimum Capacity** : Détermine le nombre minimum d'instances que l'on doit toujours avoir en fonctionnement.

**Desired Capacity** : Indique le nombre idéal d'instances que l'on souhaite maintenir en temps normal si il n'y a pas de stratégie de scaling.

**Maximum Capacity** : Fixe une limite maximale d'instances pour éviter de dépasser notre budget.

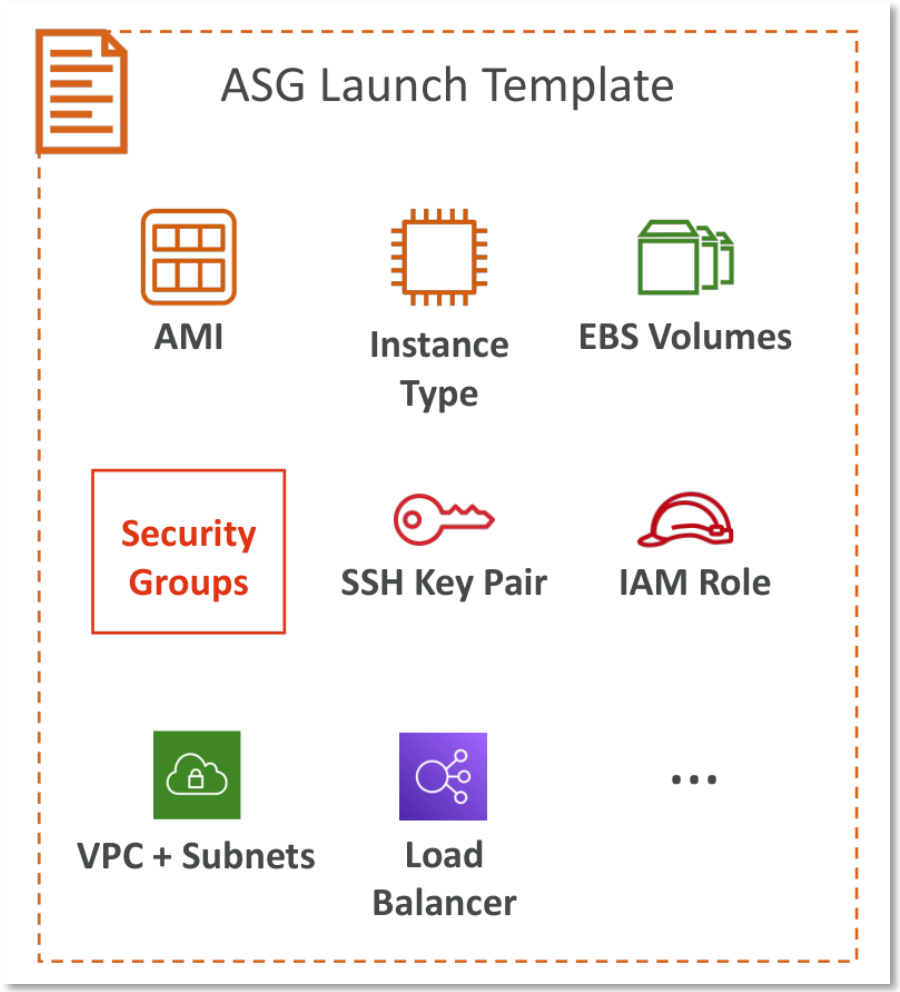


# AWS AutoScalingGroup : Le LauchTemplate

Pour pouvoir créer les instances de manière automatique l'AutoScalingGroup va se baser sur ce que l'on appelle un **LaunchTemplate**.

Le LaunchTemplate va permettre de définir plusieurs attributs relatifs aux instances :

- Le **type** d'instance (sa taille)
- L' **AMI** (OS de l'instance)
- Le **User Data** (Script d'initialisation de l'instance)
- Le(s) **Security Groups** à rattacher à l'instance
- Son **rôle** (Les permissions IAM)
- La **connectivité** de l'instance (VPC)



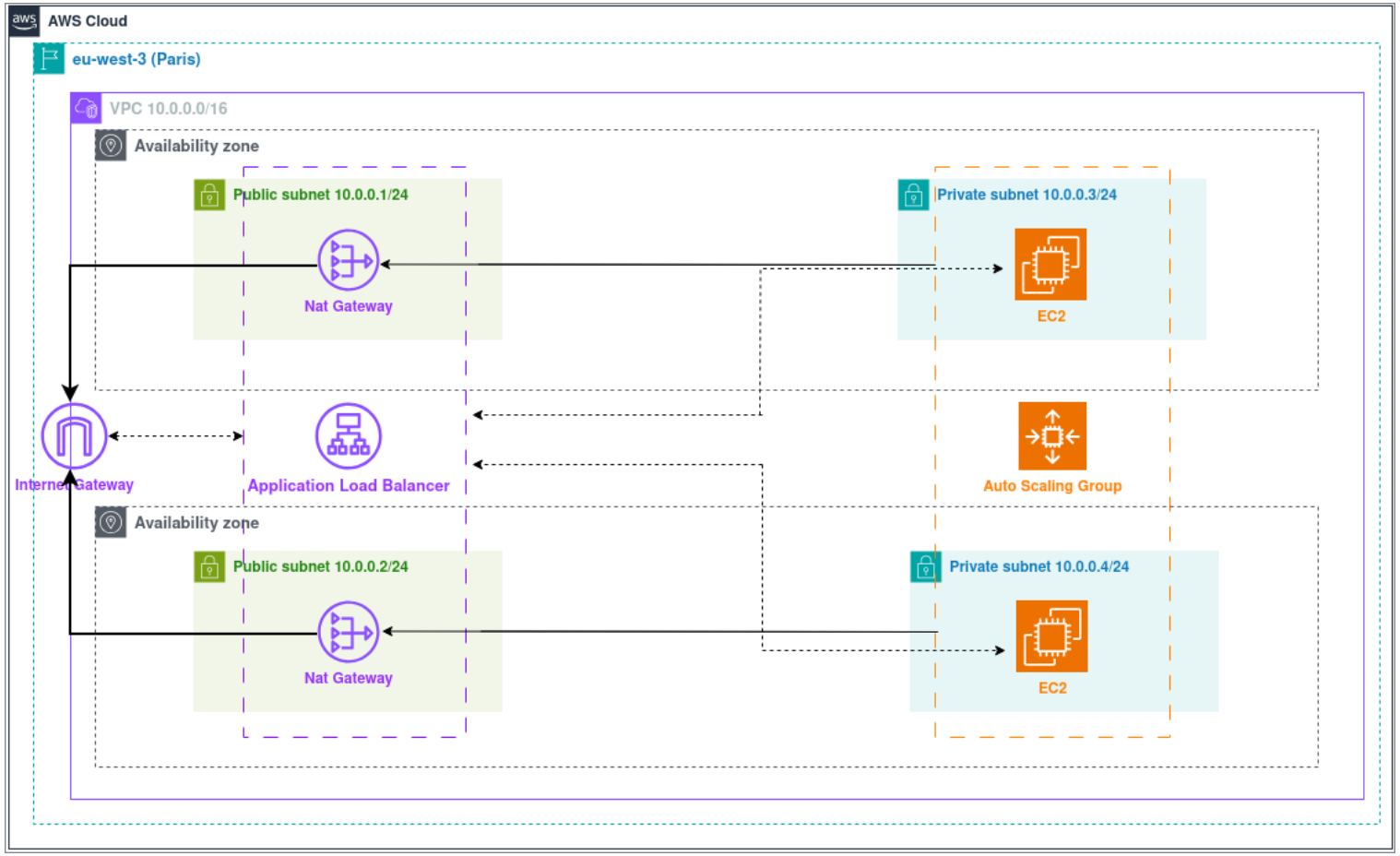

Pour aller + loin : Il est possible de versionner ces LaunchTemplates pour qu'ils puissent correspondre à différents environnements.

# La répartition des flux grâce à l'Application Load Balancer

# Architecture système du VPC : LoadBalancing

L'architecture système possède un Application Load Balancer.

- Il assure une **répartition équitable** des utilisateurs sur les instances.
- Il sert de **point d'entrée** pour les utilisateurs provenant d'internet, **protégeant les instances** de l'exposition directe.



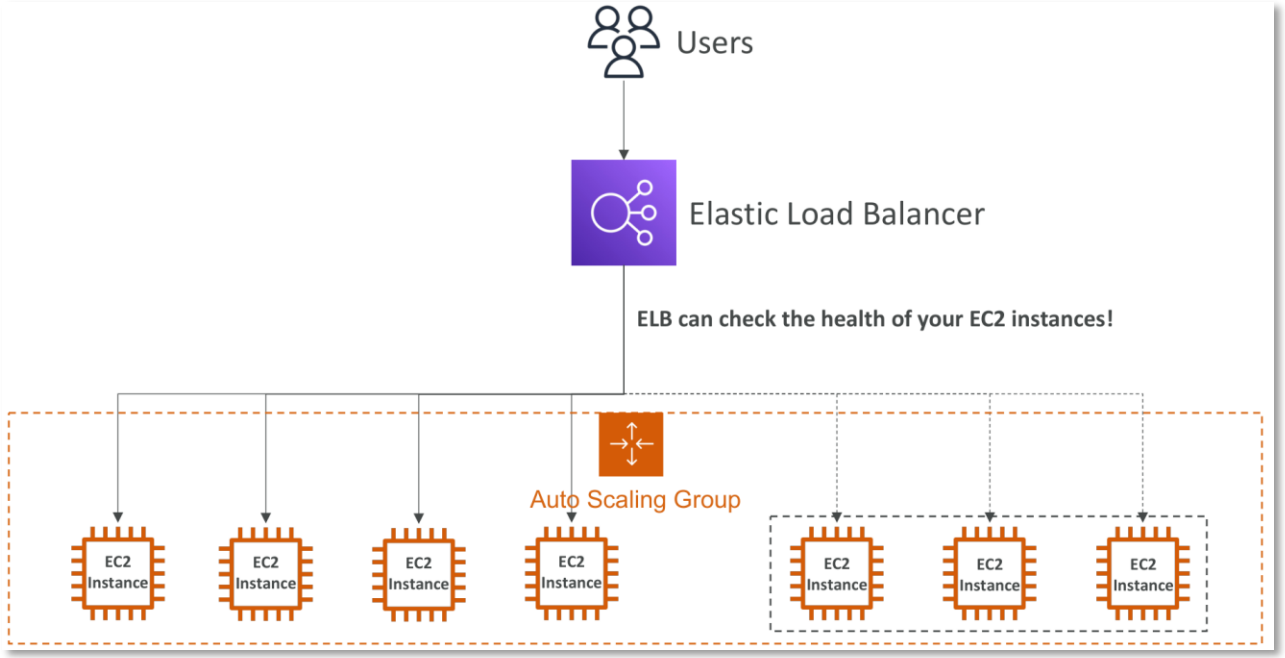

Pour aller + loin :

Il existe un Gateway Loadbalancer permettant de rediriger les flux entrants vers différentes appliances de sécurité pour analyser le trafic

# AWS ASG / ALB : Les Health Checks

L'**Application Load Balancer (ALB)** travaille de concert avec l'**AutoScaling Group (ASG)** pour assurer que les instances sont disponibles et performantes.

- L'ALB effectue des health checks pour vérifier la **disponibilité** des instances.
- Si une instance ne répond plus, elle est **désinscrite** de son **target group**.
- L'**ASG** crée alors une nouvelle instance pour **maintenir la capacité souhaitée**.




Pour aller + loin :

Il est possible de cumuler les différentes stratégies d'AutoScaling entre-elles.

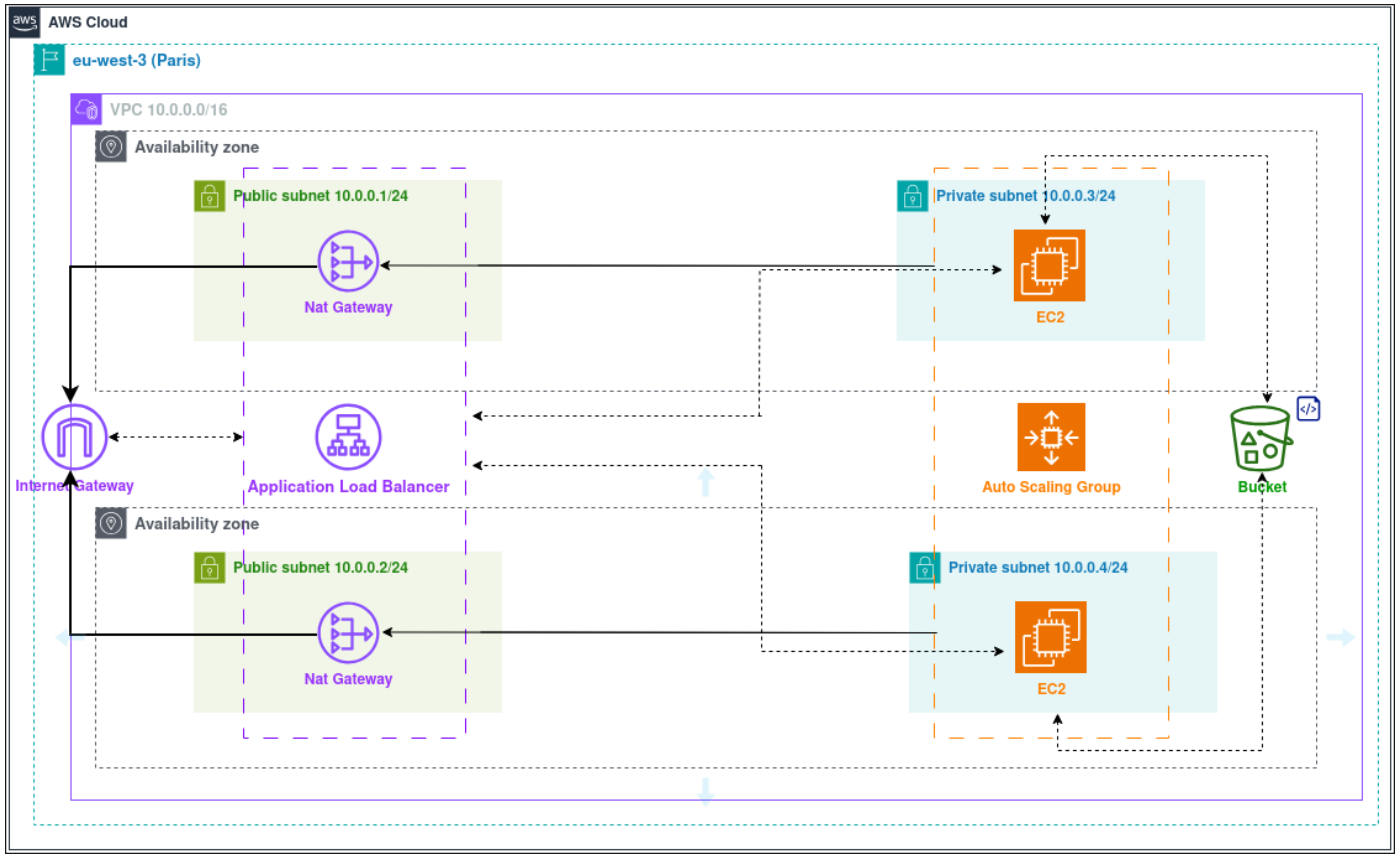
# Déploiement et Sécurité de l'Applicatif

# AWS AutoScalingGroup : Déploiement Applicatif

Le déploiement des dépendances et de l'applicatif se fait via l'utilisation d'un **script bash situé dans un bucket S3**.

A la création de l'instance le **user data va aller récupérer le script bash pour l'exécuter sur le localhost**.

On décorréle le provisionning des ressources du déploiement applicatif




Pour aller + loin :

S3 est un système de stockage par objet, la taille des objets peut aller de quelques octets à 5 TB. Chaque objet peut-être accessible via une URL.

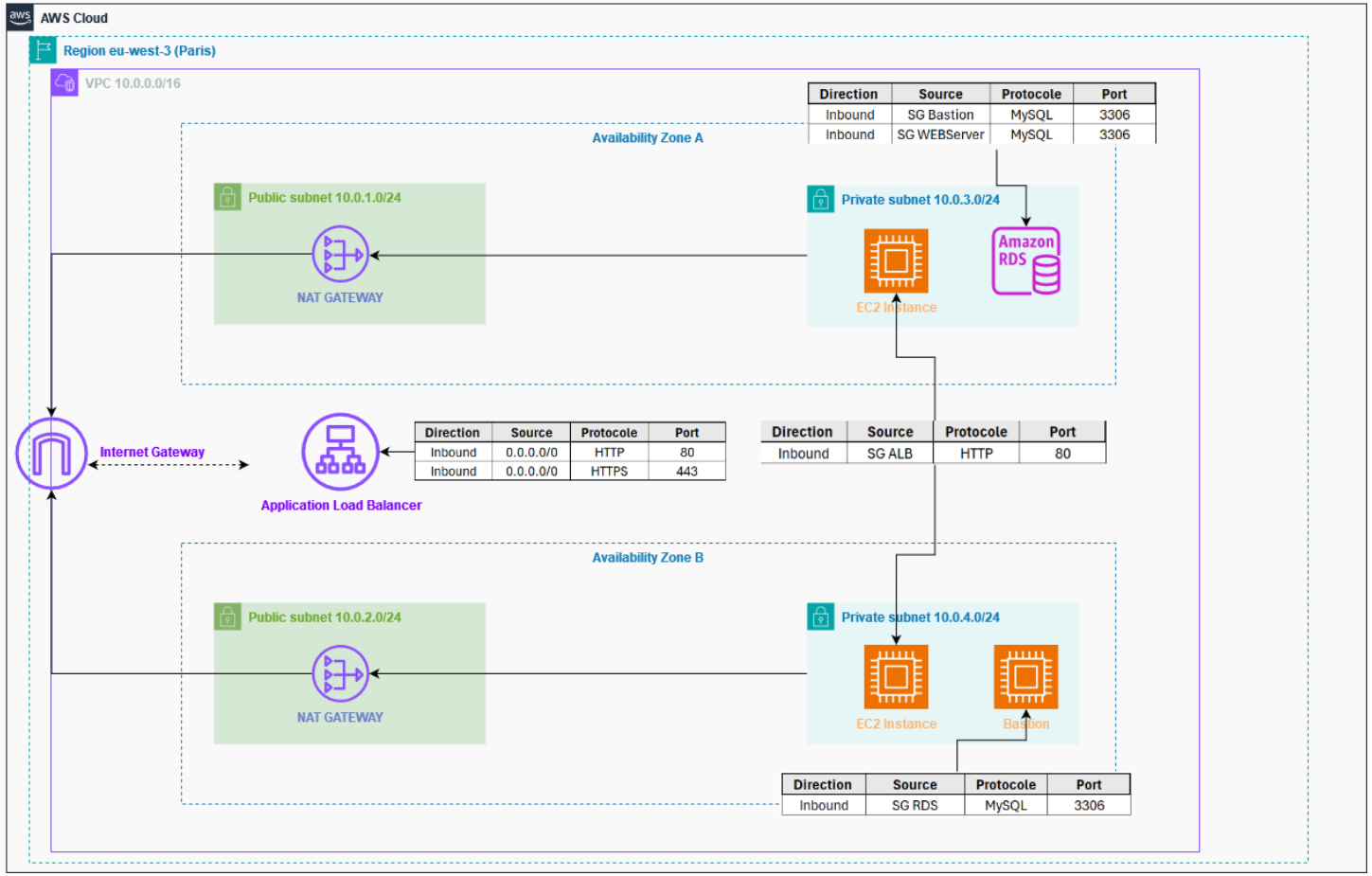
# Déploiement et Sécurité de l'Applicatif

# Architecture système du VPC : Les Security Groups

L'architecture système utilise des **Security Groups** pour gérer et contrôler le trafic réseau :

- Ce sont des **pare-feux virtuels** définissant les règles du trafic autorisé.
- **Gestion granulaire** on peut spécifier des adresses IP, des ports, des protocoles pour **permettre ou restreindre l'accès aux ressources**.

**Architecture Projet Annuel**




Pour aller + loin :

Un même Security Group peut-être attaché à plusieurs ressources, ce qui évite la duplication de ressources inutiles.

# Infrastructure As Code

# Le déploiement des ressources via CloudFormation

Le déploiement des ressources s'effectue via l'outil **d'Infrastructure As Code** : AWS CloudFormation.

CloudFormation utilise des templates YAML, un **langage déclaratif** dans lequel on vient décrire l'état des ressources que l'on souhaite créer/administrer.

Les bénéfices sont nombreux :

- **Idempotence**
- **Diminution des erreurs** humaines
- **Rapidité** de déploiement

**AWS CloudFormation**


```yaml
# Configuration du LaunchTemplate
LaunchTemplateWEBEC2:
  Type:  AWS::EC2::LaunchTemplate
  Properties:
    LaunchTemplateName: !Sub ${ProjectGroupName}-${Environment}-EC2-WEB-LaunchTemplate
    LaunchTemplateData:
      ImageId: ami-00ac45f3035ff009e
      InstanceType: !Ref InstanceType
      SecurityGroupIds:
      - !ImportValue SecGroupEC2WEB
      - !ImportValue SecGroupNoIngress
      IamInstanceProfile:
        Arn: !ImportValue InstanceProfileWebEC2Arn
      UserData:
        'Fn::Base64': !Sub |
          #!/bin/bash
          sudo apt remove awscli
          sudo apt update -y
          sudo apt install unzip
          curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
          unzip awscliv2.zip
          sudo ./aws/install
          sudo apt install nginx -y
          systemctl start nginx
          systemctl enable nginx
          sudo echo "<h1> Hello World from $(hostname -f) </h1>" > /var/www/html/index.nginx-debian.html
          sudo systemctl reload nginx
```

# WELL ARCHITECTED FRAMEWORK

# Well Architected Framework : Les 6 Pilliers


**Operational excellence**

Running and monitoring systems to deliver business value and continually improving processes and procedures.

**Reliability**

Ensuring a workload performs its intended function correctly and consistently when it's expected to

**Security**

Protecting your information systems

**Cost optimization**

Avoiding unnecessary costs

**Performance efficiency**

Using IT and compute resources efficiently

**Sustainability**

Minimizing the environmental impacts of running cloud workloads