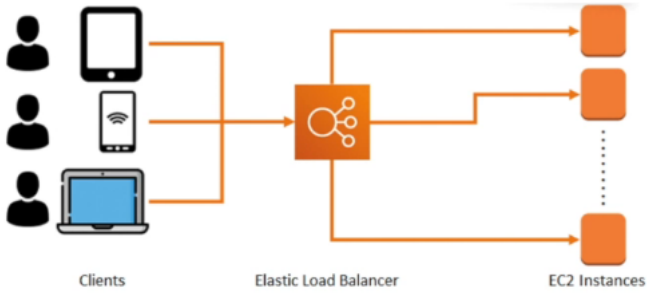
Load balancing

Video ⇒ Here



Le **load balancing** (équilibrage de charge) avec AWS est géré principalement par le service appelé **Elastic Load Balancing (ELB)**. ELB répartit automatiquement le trafic entrant des applications sur plusieurs cibles, comme des instances EC2, des conteneurs, des adresses IP, etc., dans une ou plusieurs zones de disponibilité.

Il existe trois types de load balancers sur AWS :

1. `Classic Load Balancer (CLB)` : Ancienne version qui répartit le trafic soit selon le niveau de l'application (HTTP/HTTPS) soit selon le niveau du réseau (TCP).
2. `Application Load Balancer (ALB)` : Conçu pour les applications modernes basées sur HTTP/HTTPS. Il répartit le trafic en fonction du contenu de la demande.
3. `Network Load Balancer (NLB)` : Répartit le trafic selon le niveau du réseau (TCP, UDP, TLS). Idéal pour gérer des milliers de demandes par seconde.

L'ELB est utilisé avec la surveillance et pour augmenter la disponibilité de votre application. Il vérifie la santé de vos cibles et ne redirige le trafic que vers les cibles saines.

En résumé, Elastic Load Balancing sur AWS assure que le trafic vers votre application est réparti uniformément pour éviter la surcharge d'un seul serveur, tout en augmentant la disponibilité et la fiabilité de votre application.

Elastic load Balancer with AutoScaling in terraform

Configuration du load balancer

```
# Create a new load balancer
resource "aws_elb" "ELB-01" {
  name               = "elb-01"
  subnets = [aws_subnet.pub-sb-01.id, aws_subnet.pub-sb-02.id]
  security_groups = [aws_security_group.ELB-SG.id]


  listener {
    instance_port      = 80
    instance_protocol  = "http"
    lb_port            = 80
    lb_protocol        = "http"
  }

  health_check {
    healthy_threshold   = 2
    unhealthy_threshold = 2
    timeout             = 3
    target              = "HTTP:80/"
    interval            = 30
  }
```

```
  cross_zone_load_balancing   = true
  connection_draining         = true
  connection_draining_timeout = 400

  tags = {
    Name = "elb-01"
  }
}
# Security group for AWS ELB
resource "aws_security_group" "ELB-SG" {
  name        = "elb-sg"
  description = "Security group for AWS ELB"
  vpc_id      = aws_vpc.vpc-01.id

  ingress {
    from_port        = 80
    to_port          = 80
    protocol         = "tcp"
    cidr_blocks      = [aws_vpc.vpc-01.cidr_block]
  }

  egress {
    from_port        = 0
    to_port          = 0
    protocol         = "-1"
    cidr_blocks      = ["0.0.0.0/0"]
  }

  tags = {
    Name = "elb-sg"
  }
}

# Security group for instances
resource "aws_security_group" "EC2-SG" {
  name        = "elb-ec2"
  description = "Security group for instances"
  vpc_id      = aws_vpc.vpc-01.id

  ingress {
    from_port        = 22
    to_port          = 22
    protocol         = "tcp"
    cidr_blocks      = [aws_vpc.vpc-01.cidr_block]
  }

  ingress {
      from_port        = 80
      to_port          = 80
      protocol         = "tcp"
      cidr_blocks      = [aws_security_group.ELB-SG.id]
    }

  egress {
    from_port        = 0
    to_port          = 0
    protocol         = "-1"
    cidr_blocks      = ["0.0.0.0/0"]
  }

  tags = {
    Name = "ec2-sg"
  }
}
```

Le code Terraform ci-dessus concerne la création d'un **Classic Load Balancer (ELB)** sur AWS.

1. Création du Load Balancer (ELB) :

`resource "aws_elb" "ELB-01"`: Ceci déclare une ressource de type Classic Load Balancer.

- `name`: C'est le nom unique donné au load balancer. Il sera utilisé pour l'identifier sur AWS.
- `subnets`: Ils identifient les sous-réseaux dans lesquels les instances de votre ELB seront lancées. Un ELB peut être lancé sur plusieurs sous-réseaux pour garantir sa haute disponibilité.
- `security_groups`: Ils définissent les règles de sécurité pour votre ELB. Ici, le groupe de sécurité "ELB-SG" est attaché.
- `listener`: Cette section détaille comment le trafic sera dirigé. Le trafic HTTP entrant sur le port 80 du ELB sera redirigé vers les instances sur leur port 80.

- `health_check` : C'est essentiel pour déterminer la santé des instances. Si une instance ne répond pas conformément aux critères du health check (ici une requête HTTP sur le port 80), l'ELB cessera de lui envoyer du trafic.
  - `healthy_threshold` : Le nombre de contrôles consécutifs réussis pour qu'une instance soit considérée comme saine.
  - `unhealthy_threshold` : Le nombre de contrôles consécutifs ratés pour qu'une instance soit considérée comme défaillante.
  - `timeout` : Durée après laquelle le health check doit répondre, sinon il est marqué comme échoué.
  - `interval` : Fréquence à laquelle le health check est effectué sur chaque instance.
- `cross_zone_load_balancing` : Lorsqu'il est activé, cela permet à l'ELB de distribuer le trafic de manière équilibrée entre les instances, indépendamment de leur zone de disponibilité.
- `connection_draining` **et** `connection_draining_timeout` : Ils garantissent que toutes les requêtes en cours sont traitées avant de déclasser une instance. Le timeout définit combien de temps l'ELB doit attendre.

2. Groupe de sécurité pour le Load Balancer (ELB-SG) :

`resource "aws_security_group" "ELB-SG"` : Il définit un ensemble de règles de trafic pour votre ELB.

- `ingress` : Spécifie les règles pour le trafic entrant. Ici, il est configuré pour autoriser le trafic HTTP (port 80) en provenance du VPC.
- `egress` : Spécifie les règles pour le trafic sortant. Par défaut, tout trafic sortant est autorisé.

3. Groupe de sécurité pour les instances (ELB-EC2) :

`resource "aws_security_group" "EC2-SG"` : Il détaille les règles de trafic pour les instances EC2 qui sont derrière l'ELB.

- `ingress (port 22)` : Autorise le trafic SSH. C'est utile pour se connecter et gérer les instances.
- `ingress (port 80)` : Autorise le trafic HTTP, mais seulement depuis le groupe de sécurité "ELB-SG", garantissant que seul le trafic provenant de l'ELB peut atteindre les instances sur ce port.

Ces groupes de sécurité agissent comme un pare-feu, contrôlant le trafic vers et depuis l'ELB et les instances EC2. Ils sont essentiels pour sécuriser votre infrastructure.

Configuration autoscaling connecté au groupe de sécurité de l'instance EC2

```
# Configuration de lancement pour AutoScaling
resource "aws_launch_configuration" "levelup-launchconfig" {
  name_prefix   = "levelup-launchconfig"
  image_id      = lookup(var.AMIS, var.AWS_REGION)
  instance_type = "t3.micro" # J'ai changé le type d'instance de t2.micro à t3.micro
  key_name      = aws_key_pair.levelup_key.key_name
  security_groups = [aws_security_group.ELB-EC2.id]
  user_data       = "#!/bin/bash\napt-get update\napt-get -y install net-tools nginx\nMYIP=`ifconfig | grep -E '(inet 10)|(addr:10)' |


  lifecycle {
    create_before_destroy = true
  }
}

# Générer une clé
resource "aws_key_pair" "levelup_key" {
    key_name   = "levelup_key"
    public_key = file(var.PATH_TO_PUBLIC_KEY)
}

# Groupe AutoScaling
resource "aws_autoscaling_group" "levelup-autoscaling" {
  name                      = "levelup-autoscaling"
  vpc_zone_identifier       = [aws_subnet.pub-sb-01.id, aws_subnet.pub-sb-02.id]
  launch_configuration      = aws_launch_configuration.levelup-launchconfig.name
  min_size                  = 2
  max_size                  = 2
  health_check_grace_period = 200
  health_check_type         = "ELB"
  load_balancers            = [aws_elb.ELB-01.name]
  force_delete              = true

  tag {
    key                 = "Name"
    value               = "Instance EC2 via LB"
    propagate_at_launch = true
```

```
  }
}

output "ELB" {
  value = aws_elb.ELB-01.dns_name
}
```

1. Configuration de lancement pour AutoScaling (`aws_launch_configuration`)

Cette section décrit comment les nouvelles instances EC2 seront lancées pour l'AutoScaling.

- `name_prefix`: Préfixe pour le nom de la configuration de lancement. Un identifiant unique sera ajouté après ce préfixe.
- `image_id`: L'ID de l'image Amazon Machine (AMI) qui sera utilisée pour lancer les instances.
- `instance_type`: Le type d'instance EC2 qui sera lancé. Le commentaire indique que le type d'instance a été changé de "t2.micro" à "t3.micro".
- `key_name`: La clé EC2 qui sera associée à ces instances.
- `security_groups`: Le groupe de sécurité qui sera attaché à ces instances.
- `user_data`: Script Bash qui sera exécuté lors de la première mise en route de l'instance. Ce script met à jour le système, installe `net-tools` et `nginx`, puis modifie la page d'accueil de nginx pour afficher l'adresse IP de l'instance.
- `lifecycle`: Assure que la nouvelle configuration de lancement est créée avant que l'ancienne ne soit détruite. C'est utile pour éviter des interruptions.

2. Générer une clé (`aws_key_pair`)

Cette section crée une paire de clés pour accéder à l'instance EC2.

- `key_name`: Nom de la clé.
- `public_key`: Chemin vers la clé publique sur la machine locale. La clé privée correspondante est nécessaire pour se connecter à l'instance.

3. Groupe AutoScaling (`aws_autoscaling_group`)

- `name`: Nom du groupe AutoScaling.
- `vpc_zone_identifier`: Sous-réseaux où les instances seront lancées.
- `launch_configuration`: La configuration de lancement (décrite ci-dessus) qui sera utilisée par ce groupe AutoScaling.
- `min_size` **et** `max_size`: Le nombre minimum et maximum d'instances dans ce groupe.
- `health_check_grace_period`: Le temps (en secondes) pendant lequel l'ELB attend avant d'effectuer des vérifications de santé sur les nouvelles instances.
- `health_check_type`: Type de vérification de santé à utiliser (ici, c'est ELB).
- `load_balancers`: Les noms des load balancers qui distribueront le trafic vers ce groupe AutoScaling.
- `force_delete`: Si "true", cela permettra de supprimer le groupe d'AutoScaling même s'il contient des instances.
- `tag`: Tags pour les ressources AWS créées. Ici, il définit le nom de chaque instance.

4. Sortie (`output`)

```
# Nom de domaine du ELB créé
elb-01-1200719656.us-east-2.elb.amazonaws.com
```

Cela affiche le nom de domaine du ELB créé. Utile pour accéder au Load Balancer une fois qu'il est opérationnel.

En résumé, ce code Terraform crée une configuration de lancement et un groupe d'AutoScaling pour gérer automatiquement la mise à l'échelle des instances EC2, en fonction des besoins. Ces instances sont associées à un Load Balancer et sont configurées pour exécuter un serveur web nginx affichant leur adresse IP.

Execution du serveur Nginx installé via le script bash via le nom de domaine de ELB

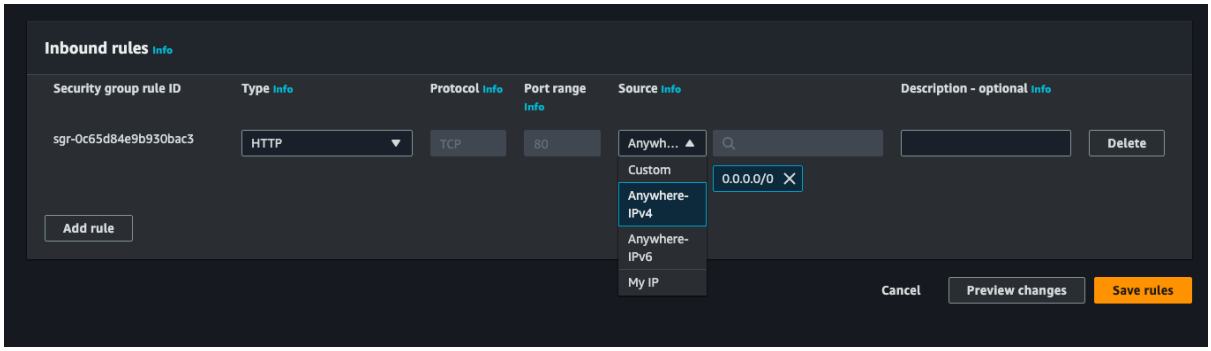
```
  user_data       = "#!/bin/bash\napt-get update\napt-get -y install net-tools nginx\nMYIP=`ifconfig | grep -E '(inet 10)|(addr:10)' | awk '{ print $2 }' | cut -d ':' -f2`\necho 'Hello Team\nThis is my IP: '$MYIP > /var/www/html/index.html"
```

```
curl elb-01-1200719656.us-east-2.elb.amazonaws.com
```

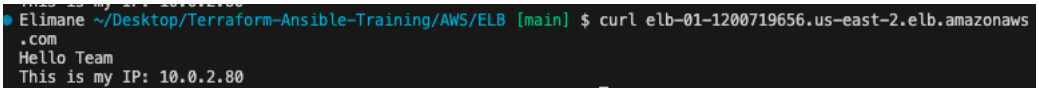
NB:

Au cas où l'execution de la commande ci-dessus ne fonctionne pas il faut aller configurer le groupe de sécurité attaché au ELB, en modifiant la source (Traffic entrant) de la règle entrante

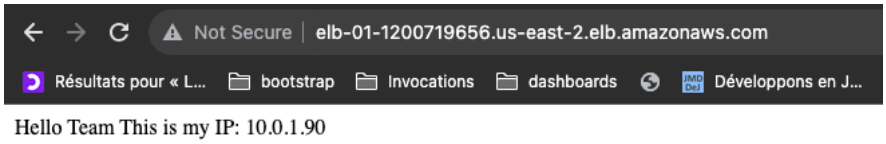
Choisir `AnyWhere-IPV4` et sauvegarder.



Résultat via le terminal:



Via le navigateur:



Changement d'adresse IP lors de la connection aux instances via l' ELB

Lorsque vous utilisez un Elastic Load Balancer (ELB) dans AWS pour répartir le trafic entre plusieurs instances EC2, l'adresse IP de l'ELB n'est pas fixe. De plus, les adresses IP des instances EC2 peuvent également changer, en particulier si elles sont dans un groupe d'auto-scaling et qu'elles sont terminées puis recréées.

Voici quelques raisons pour lesquelles l'adresse IP d'une instance EC2 peut changer lorsqu'elle est appelée via ELB :

1. **Elasticité de l'ELB** : L'ELB est conçu pour être hautement disponible et élastique. AWS peut changer les adresses IP associées à un ELB en fonction des besoins de capacité ou d'autres facteurs opérationnels.
2. **Instances EC2 dans un groupe Auto Scaling** : Si votre instance EC2 fait partie d'un groupe Auto Scaling et qu'elle est terminée pour une raison quelconque (par exemple, un échec de santé ou une mise à l'échelle), une nouvelle instance sera lancée pour la remplacer. Cette nouvelle instance aura une adresse IP différente.
3. **Redémarrage de l'instance** : Si une instance EC2 non associée à une adresse IP élastique est arrêtée puis redémarrée, elle peut se voir attribuer une nouvelle adresse IP.
4. **Conception de l'ELB** : L'ELB répartit le trafic entre les instances EC2 en fonction de plusieurs facteurs, tels que la charge de travail, la santé des instances et les algorithmes de répartition de charge. Il est possible que l'ELB dirige le trafic vers

différentes instances à différents moments, chacune ayant une adresse IP différente.

Pour avoir une adresse IP fixe pour une instance EC2, vous devez utiliser une "Adresse IP élastique". Cependant, il est important de noter que l'ELB lui-même n'a pas d'adresse IP fixe, même si AWS garantit que le nom de domaine fourni pour l'ELB restera constant.