# AWS Elastic Load Balancing tips and tricks: from basic to pro

12 August 2022 - 8 min. read

Advanced Networking | Application Load Balancer (ALB) | AWS Elastic Load Balancer (ELB) | Network Load Balancer (NLB)

An old Italian commercial claimed: "Two is better than one."

When it comes to **application availability**, having multiple instances available makes a business more resilient: this can help you achieve **fault tolerance** for applications and infrastructures. You know: "Everything can fail, all the time".

**Load balancing** is the fundamental building block that can make us achieve better uptime and application availability: redistributing traffic on different instances in auto scaling and checking their healthiness isn't always as easy as it seems.

In my past career as a system administrator, I always struggled to find a resilient, redundant, and elastic solution. After a lot of work and automation, I could have a decent night of sleep!

Using managed services, as always, can help us reduce the amount of work required to achieve our goals.

When it comes to load-balancing, AWS gives us a lot of flexibility: under the "Elastic Load Balancing (ELB)" umbrella, many options are available.

This article will explain ELB fundamentals and deep dive into not-so-common use cases.

## Basic Key Concepts

Three managed load balancer types are available: **Application**, **Network**, and **Gateway**.

In this article, we'll focus on **Application Load Balancers** and **Network Load Balancers**, while we will not consider the Classic Load balancer, as it is a mix of application and network load balancers with some missing features (check this page for comparison), and Gateway Load Balancers. We will write about them in a dedicated article.

Every type of ELB can span multiple Availability Zones and relies on three main components:

- **listener**
- **target group**
- **health checks**

ELB can be public (internet facing) or private (so that only private routed resources can access them).

A **listener** is a small portion of the ELB configuration that defines the entry point of the traffic. If our application uses the HTTP protocol on port 8080, the listener configuration will route traffic to the same protocol and port.

A **target group** is the set of computational resources that handle application traffic distributed by the load balancer.

A target group can use EC2 instances, ECS containers, IP addresses, lambda functions, or even another application load balancer!

Note that not every ELB type can use all the targets: for example, only an Application Load Balancer can have lambda functions as a target.

A **health check** is a test used by the target group to determine if the target is healthy, so failing compute resources are excluded and will not receive any traffic. For example, if our application receives TCP traffic on port 31337, the health check verifies that the service is listening.

## Application Load Balancer

The application load balancer (ALB) operates at ISO/OSI level 7, thus the name.

It's the most commonly used because it balances HTTP and HTTPS traffic and can **perform redirection, authentication, and SSL offloading** using Amazon Certificate Manager (ACM) certificates.

## Network Load Balancer

The network load balancer (NLB) operates at ISO/OSI level 4 (network or connection level). It can handle any traffic based on TCP or UDP without specialization for a specific application protocol. TLS offloading is supported, offering static IP addresses to use (for both internal and external load balancers).

## Gateway Load Balancer

Gateway load balancers are the last addition to the ELB family. It routes level 3 traffic using GENEVE encapsulation, enabling usage of third-party and custom network and security appliances.

Suppose you want to deploy a custom IDS solution to analyze traffic in transit: Gateway load balancing helps achieve high availability, using multiple instances in different availability zones instead of a single one.

So far so good. And straightforward. Let's now dive deep into **five (+1)** less common **configuration scenarios that can make your day**: how to configure **load balancers like a pro!**

## 1. Use ALB to secure WordPress

Refactoring applications and adapting them to use the Cloud efficiently isn't always an option when you are in a rush.

WordPress is one of the most common examples of an application that isn't easy to migrate; if you have the time, please consider containerization.

When migrating a WordPress installation, you need to keep the same DNS name and the connection on HTTPS. The quick and dirty solution could be to use a public amazon EC2 instance, but exposing a WordPress installation without a WAF can be risky.

Placing an Application Load Balancer in front of the WordPress instance can help **reduce administration overhead** and **improve security**, even if you aren't balancing traffic between different instances:

- The EC2 instance will not be publicly accessible
- You can place a WAF and use the WordPress specialized ruleset
- Amazon Certificate Manager will give us public certificates and renew them

ALB will perform HTTP redirection to the HTTPS listener in this use case, and LetsEncrypt CertBot will issue SSL certificates on the instance.

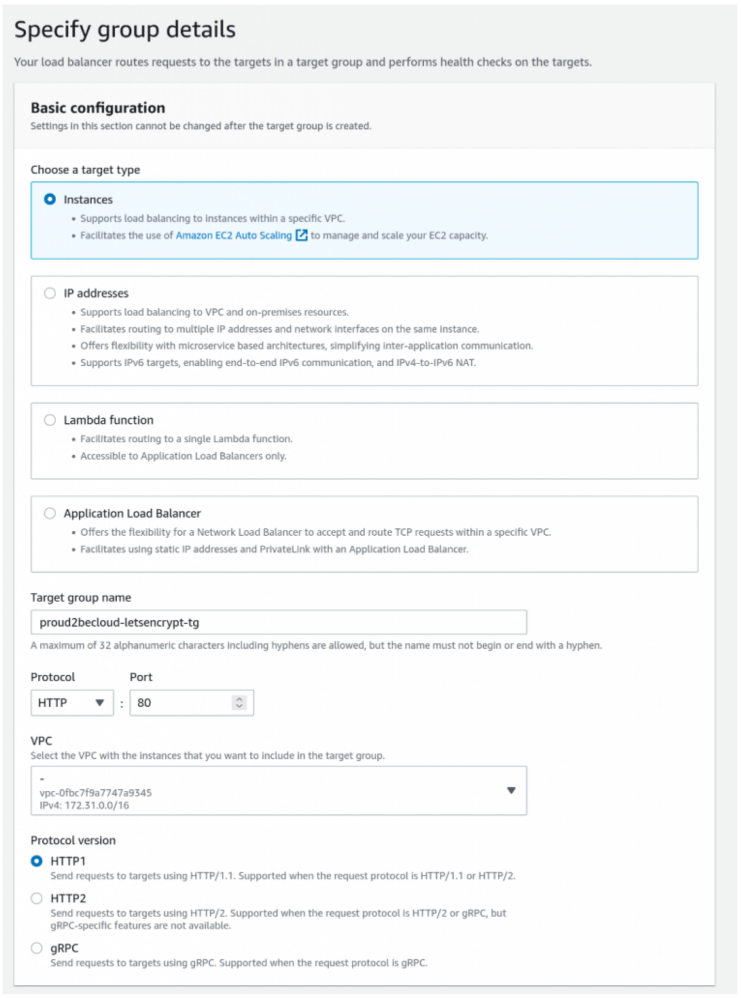
Associating a Web Application Firewall rule to the load balancer is relatively easy; let's see how to configure our ALB to allow certificate renewal with LetsEncrypt.

After obtaining CertBot (https://certbot.eff.org/instructions), you can issue free certificates for your website, but the verification process requires access to a path on your web server using HTTP

```
(/.well-known-acme-challenge/)
```

To achieve this, we'll need to define a target group that uses port 80 and then insert a rule that will use that target group for the acme-challenge path.

First, go to the management console and create a new target group:



**Specify group details**

Your load balancer routes requests to the targets in a target group and performs health checks on the targets.

**Basic configuration**

Settings in this section cannot be changed after the target group is created.

Choose a target type

[✓ Instances]
- Supports load balancing to instances within a specific VPC.
- Facilitates the use of Amazon EC2 Auto Scaling to manage and scale your EC2 capacity.

IP addresses
- Supports load balancing to VPC and on-premises resources.
- Facilitates routing to multiple IP addresses and network interfaces on the same instance.
- Offers flexibility with microservice based architectures, simplifying inter-application communication.
- Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.

Lambda function
- Facilitates routing to a single Lambda function.
- Accessible to Application Load Balancers only.

Application Load Balancer
- Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
- Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.

Target group name: proud2becloud-letsencrypt-tg

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Protocol: HTTP : Port: 80

VPC

Select the VPC with the instances that you want to include in the target group.

-
vpc-0fbc7f9a7747a9345
IPv4: 172.31.0.0/16

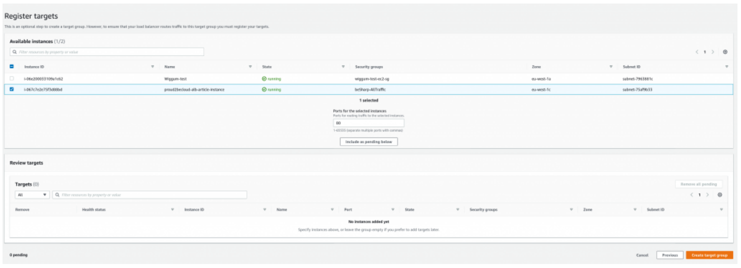
Protocol version

[✓ HTTP1]
Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.

HTTP2
Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.

gRPC
Send requests to targets using gRPC. Supported when the request protocol is gRPC.

Then select your instance and click "Include as pending below."



**Register targets**

This is an optional step to create a target group. However, to ensure that your load balancer routes traffic to this target group you must register your targets.

**Available instances** (1/2)

| | Instance ID | Name | State | Security groups | Zone | Subnet ID |
|---|---|---|---|---|---|---|
| ☐ | i-06e200033109a1c62 | Wiggum-test | running | wiggum-test-ec2-sg | eu-west-1a | subnet-7963881c |
| ☑ | i-067c7e2e75f3d88bd | proud2becloud-alb-article-instance | running | beSharp-AllTraffic | eu-west-1c | subnet-75af9b33 |

1 selected

Ports for the selected instances
Ports for routing traffic to the selected instances.
80
1-65535 (separate multiple ports with commas)

Include as pending below

**Review targets**

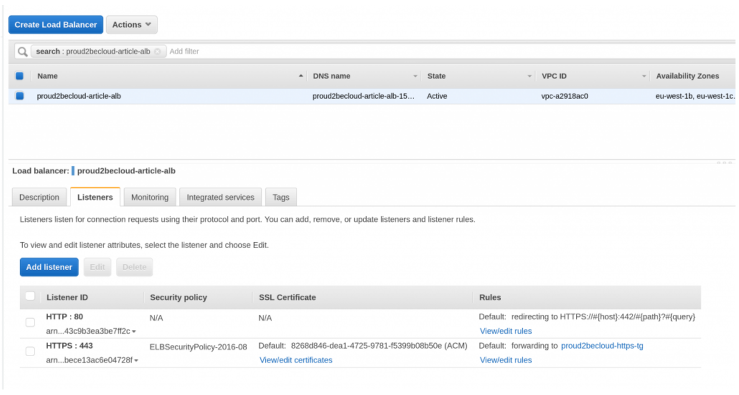
**Targets** (0)

| Remove | Health status | Instance ID | Name | Port | State | Security groups | Zone | Subnet ID |
|---|---|---|---|---|---|---|---|---|

No instances added yet
Specify instances above, or leave the group empty if you prefer to add targets later.

0 pending

After creating the target group, click on "**View/edit rules**" on the HTTP listener to add a new rule:



search : proud2becloud-article-alb

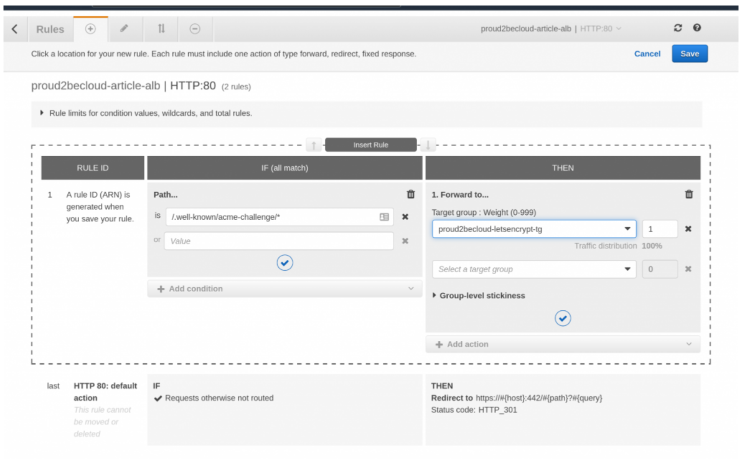
| Name | DNS name | State | VPC ID | Availability Zones |
|---|---|---|---|---|
| proud2becloud-article-alb | proud2becloud-article-alb-15... | Active | vpc-a2918ac0 | eu-west-1b, eu-west-1c |

Load balancer: proud2becloud-article-alb

Listeners listen for connection requests using their protocol and port. You can add, remove, or update listeners and listener rules.

To view and edit listener attributes, select the listener and choose Edit.

| Listener ID | Security policy | SSL Certificate | Rules |
|---|---|---|---|
| HTTP : 80<br>arn...43c9b3ea3be7ff2c | N/A | N/A | Default: redirecting to HTTPS://#{host}:442/#{path}?#{query}<br>View/edit rules |
| HTTPS : 443<br>arn...bece13ac6e04728f | ELBSecurityPolicy-2016-08 | Default: 8268d846-dea1-4725-9781-f5399b08b50e (ACM)<br>View/edit certificates | Default: forwarding to proud2becloud-https-tg<br>View/edit rules |



Rules

proud2becloud-article-alb | HTTP:80

Click a location for your new rule. Each rule must include one action of type forward, redirect, fixed response.

Cancel Save

proud2becloud-article-alb | HTTP:80 (2 rules)

Rule limits for condition values, wildcards, and total rules.

Insert Rule

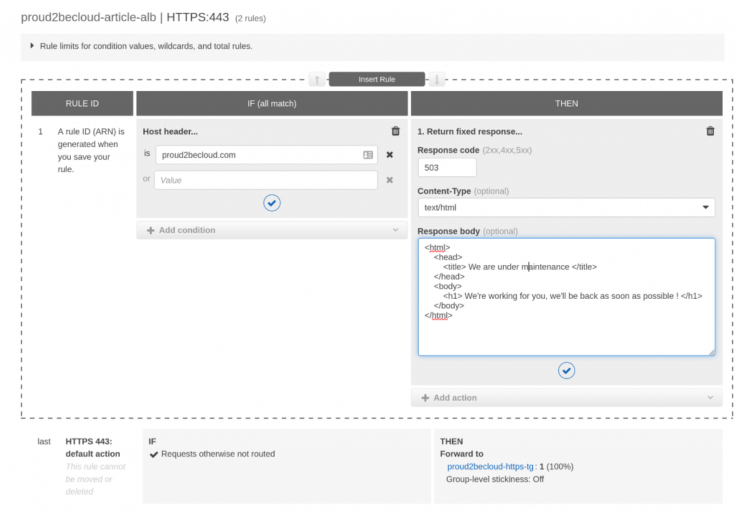
| RULE ID | IF (all match) | THEN |
|---|---|---|
| 1 A rule ID (ARN) is generated when you save your rule. | Path... is /.well-known/acme-challenge/* or Value | 1. Forward to... Target group : Weight (0-999) proud2becloud-letsencrypt-tg 1 Traffic distribution 100% Select a target group 0 Group-level stickiness |
| last HTTP 80: default action This rule cannot be moved or deleted | IF ✓ Requests otherwise not routed | THEN Redirect to https://#{host}:442/#{path}?#{query} Status code: HTTP_301 |

Add condition · Add action

Remember that this rule should have a higher priority than the default action that redirects HTTP to HTTPS

## 2. Simple and quick maintenance page for your site

It can sound obvious, but sometimes we forget it. Setting up a rule with a fixed response with an HTTP status code of 503 will enable us to serve a maintenance page quickly for a given website:



proud2becloud-article-alb | HTTPS:443 (2 rules)

Rule limits for condition values, wildcards, and total rules.

Insert Rule

| RULE ID | IF (all match) | THEN |
|---|---|---|
| 1 A rule ID (ARN) is generated when you save your rule. | Host header... is proud2becloud.com or Value | 1. Return fixed response... Response code (2xx,4xx,5xx) 503 Content-Type (optional) text/html Response body (optional) |
| last HTTPS 443: default action This rule cannot be moved or deleted | IF ✓ Requests otherwise not routed | THEN Forward to proud2becloud-https-tg : 1 (100%) Group-level stickiness: Off |

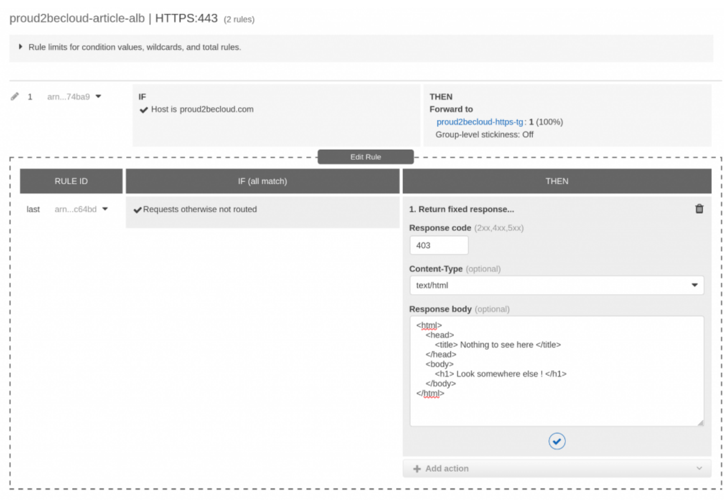
```
<html>
  <head>
    <title> We are under maintenance </title>
  </head>
  <body>
    <h1> We're working for you, we'll be back as soon as possible ! </h1>
  </body>
</html>
```

Add condition · Add action

## 3. Host-based HTTP routing to reduce bot scanning

When using the default action to target our web application, all traffic will be forwarded, even bot scanning for vulnerabilities targeting every public IP address with an open port. This behavior can make our logs "noisy".

To avoid this, change the default action to a fixed response and add a more specific rule for our host:


proud2becloud-article-alb | HTTPS:443 (2 rules)

Rule limits for condition values, wildcards, and total rules.

1 arn...74ba9 — IF ✓ Host is proud2becloud.com — THEN Forward to proud2becloud-https-tg: 1 (100%) Group-level stickiness: Off

Edit Rule

| RULE ID | IF (all match) | THEN |
|---|---|---|
| last arn...c64bd | ✓ Requests otherwise not routed | 1. Return fixed response... Response code (2xx,4xx,5xx): 403; Content-Type (optional): text/html; Response body (optional) |

```
<html>
  <head>
    <title> Nothing to see here </title>
  </head>
  <body>
    <h1> Look somewhere else ! </h1>
  </body>
</html>
```

## 4. Implement custom logic for health checks in ECS tasks balanced using an NLB

For this example, we'll use a Minecraft server, even if it's not a traditional workload. Minecraft runs on port 25565 using its custom protocol.

We can run it in an ECS Fargate Cluster, configuring a service to keep the server running with "auto-healing" and setting the desired count of 1.

With a network load balancer, we can have static IP addresses to connect to, so the healthy container will always receive traffic without worrying when its internal IP changes.

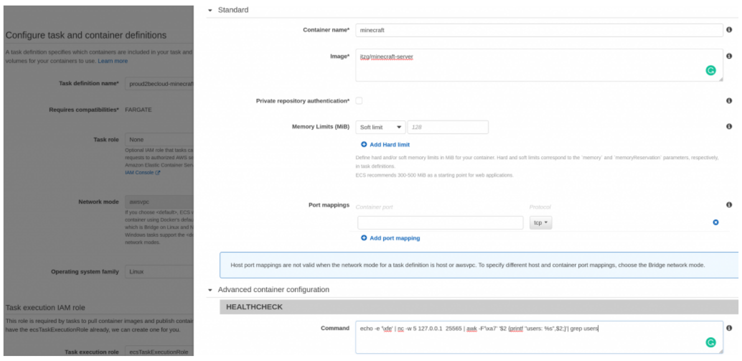
We will configure the target group to check if port 25565 is reachable, but we want to take a step further: we want to be sure that the server will respond to a command.

This bash script can check the number of connected users; unfortunately, we can't configure our target group to execute it.

```
echo -e '\xfe' | nc -w 5 127.0.0.1 25565 | awk -F'\xa7' '$2 {printf
 "users: %s",$2;}'| grep users
```

Lucky for us, ECS Fargate will let us define a custom health check that will run in our task instance.

Simply add to your container definition the health check command so that a failure will stop the container and a new instance will take its place. Remember that the health check script has to return 0 to be considered OK.


Configure task and container definitions

A task definition specifies which containers are included in your task and volumes for your containers to use. Learn more

Task definition name* proud2becloud-minecraf

Requires compatibilities* FARGATE

Task role None

Optional IAM role that tasks ca requests to authorized AWS se Amazon Elastic Container Serv IAM Console

Network mode awsvpc

If you choose <default>, ECS v container using Docker's defau which is Bridge on Linux and N Windows tasks support the <de network modes.

Operating system family Linux

Task execution IAM role

This role is required by tasks to pull container images and publish contai have the ecsTaskExecutionRole already, we can create one for you.

Task execution role ecsTaskExecutionRole

Standard

Container name* minecraft

Image* itzg/minecraft-server

Private repository authentication*

Memory Limits (MiB) Soft limit 128

Add Hard limit

Define hard and/or soft memory limits in MiB for your container. Hard and soft limits correspond to the `memory` and `memoryReservation` parameters, respectively, in task definitions.

ECS recommends 300-500 MiB as a starting point for web applications.

Port mappings Container port Protocol tcp

Add port mapping

Host port mappings are not valid when the network mode for a task definition is host or awsvpc. To specify different host and container port mappings, choose the Bridge network mode.

Advanced container configuration

HEALTHCHECK

Command

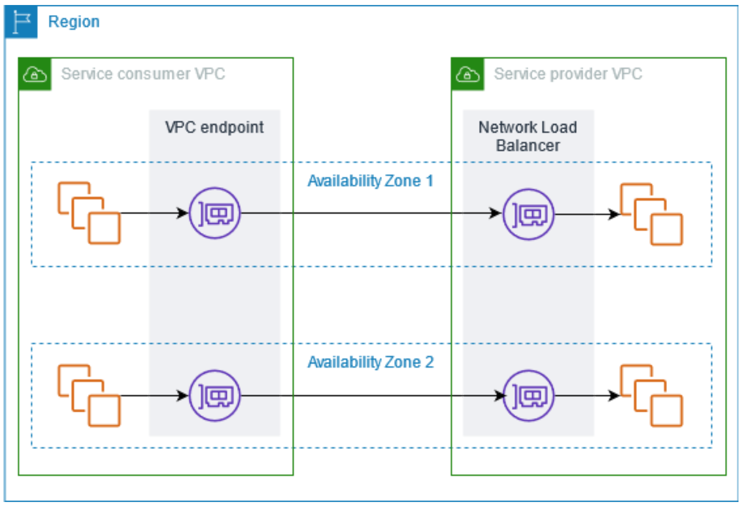
```
echo -e '\xfe' | nc -w 5 127.0.0.1  25565 | awk -F'\xa7' '$2 {printf "users: %s",$2;}'| grep users
```

## 5. Share services to other AWS Accounts without crossing the Internet

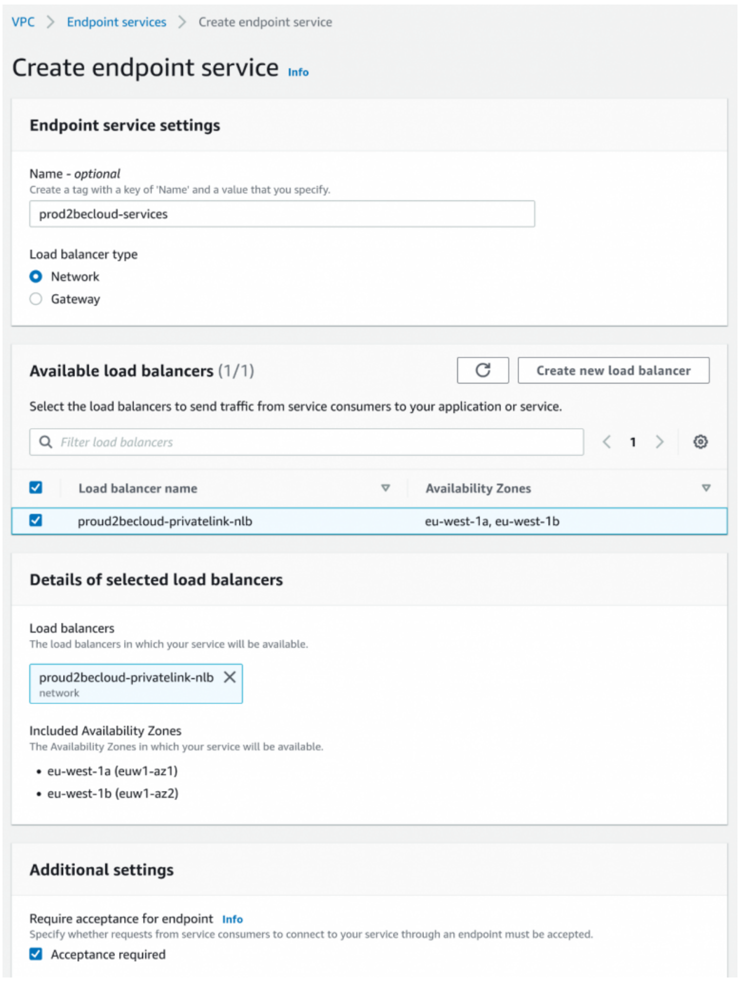
AWS PrivateLink is a technology that allows you to offer services to customers and partners, taking advantage of the AWS network infrastructure without crossing the internet.

When you share a service with other AWS accounts, you become the service provider; the other AWS accounts become Service consumers.

It works by linking a Network Loadbalancer's network interfaces with other accounts, placing a local interface in the consumer VPC.



To share services, you first need to create an **Endpoint Service**, go to the VPC console, select Endpoint Services, and create a new one by selecting **"Network"** as the load balancer type.



VPC > Endpoint services > Create endpoint service

Create endpoint service Info

**Endpoint service settings**

Name - *optional*
Create a tag with a key of 'Name' and a value that you specify.
prod2becloud-services

Load balancer type
[✓ Network]
Gateway

**Available load balancers** (1/1)

Select the load balancers to send traffic from service consumers to your application or service.

| | Load balancer name | Availability Zones |
|---|---|---|
| ✓ | proud2becloud-privatelink-nlb | eu-west-1a, eu-west-1b |

**Details of selected load balancers**

Load balancers
The load balancers in which your service will be available.
proud2becloud-privatelink-nlb
network

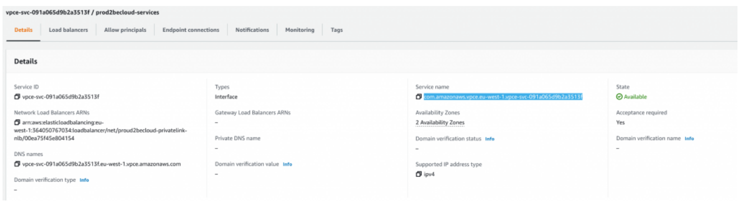
Included Availability Zones
The Availability Zones in which your service will be available.

- eu-west-1a (euw1-az1)
- eu-west-1b (euw1-az2)

**Additional settings**

Require acceptance for endpoint Info
Specify whether requests from service consumers to connect to your service through an endpoint must be accepted.
[✓ Acceptance required]

Once the service endpoint is available, please note the Service Name in the console.

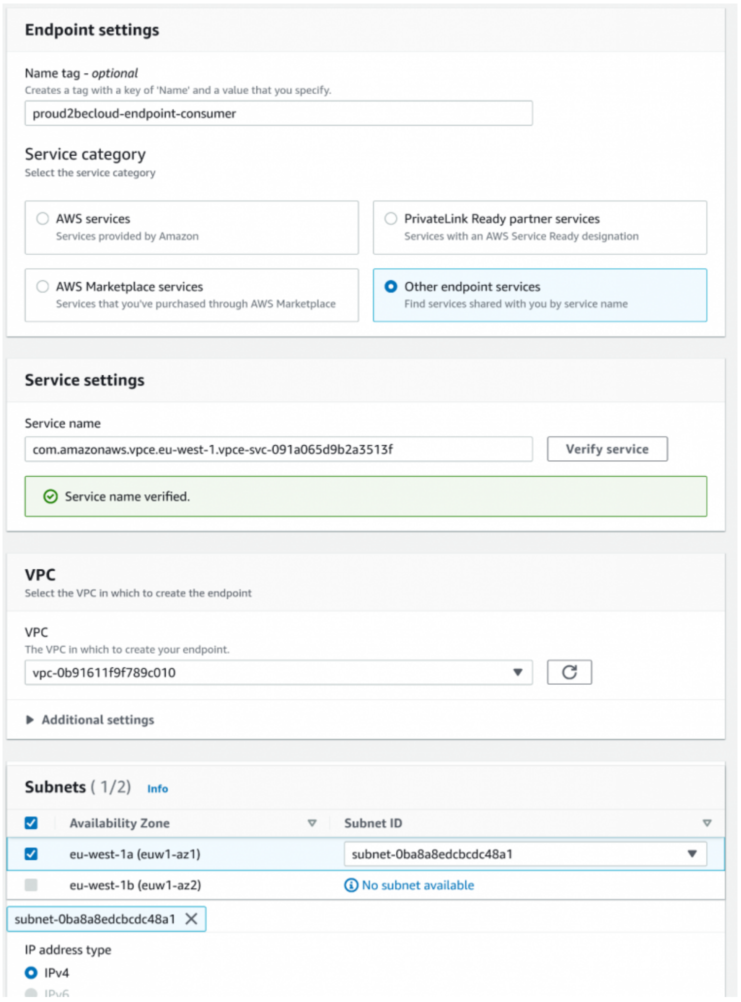


vpce-svc-091a065d9b2a3513f / prod2becloud-services

**Details**

| | | | |
|---|---|---|---|
| Service ID<br>vpce-svc-091a065d9b2a3513f | Types<br>Interface | Service name<br>com.amazonaws.vpce.eu-west-1.vpce-svc-091a065d9b2a3513f | State<br>Available |
| Network Load Balancers ARNs<br>arn:aws:elasticloadbalancing:eu-west-1:364050767034:loadbalancer/net/proud2becloud-privatelink-nlb/00ea75f45e804154 | Gateway Load Balancers ARNs<br>– | Availability Zones<br>2 Availability Zones | Acceptance required<br>Yes |
| DNS names<br>vpce-svc-091a065d9b2a3513f.eu-west-1.vpce.amazonaws.com | Private DNS name<br>– | Domain verification status Info<br>– | Domain verification name Info<br>– |
| Domain verification type Info<br>– | Domain verification value Info<br>– | Supported IP address type<br>ipv4 | |

**To use the endpoint service as a consumer,** go to the "Endpoints" section in the VPC, select "Create a new endpoint," and then select "**other endpoint services**." Paste the name you previously noted and then click "**Verify Service**." Once the service name is verified, you can select the desired VPC, subnet, and security group to use to deploy the endpoint.



**Endpoint settings**

Name tag - *optional*
Creates a tag with a key of 'Name' and a value that you specify.
proud2becloud-endpoint-consumer

Service category
Select the service category

- AWS services — Services provided by Amazon
- PrivateLink Ready partner services — Services with an AWS Service Ready designation
- AWS Marketplace services — Services that you've purchased through AWS Marketplace
- [✓ Other endpoint services — Find services shared with you by service name]

**Service settings**

Service name
com.amazonaws.vpce.eu-west-1.vpce-svc-091a065d9b2a3513f
Service name verified.

**VPC**
Select the VPC in which to create the endpoint

VPC
The VPC in which to create your endpoint.
vpc-0b91611f9f789c010

Additional settings

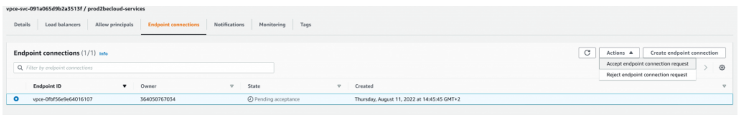
**Subnets** ( 1/2)

| | Availability Zone | Subnet ID |
|---|---|---|
| ✓ | eu-west-1a (euw1-az1) | subnet-0ba8a8edcbcdc48a1 |
| | eu-west-1b (euw1-az2) | No subnet available |

subnet-0ba8a8edcbcdc48a1

IP address type
[✓ IPv4]
IPv6

Remember that you need to accept the connection if you enable the "Acceptance Required" checkbox while creating the endpoint service.



vpce-svc-091a065d9b2a3513f / prod2becloud-services

Details
Load balancers
Allow principals
Endpoint connections
Notifications
Monitoring
Tags
Actions
Create endpoint connection
Accept endpoint connection request
Reject endpoint connection request
Filter by endpoint connections

Endpoint connections (1/1)

| Endpoint ID | Owner | State | Created |
|---|---|---|---|
| vpce-0fbf56e9e64016107 | 364050767034 | Pending acceptance | Thursday, August 11, 2022 at 14:45:45 GMT+2 |

After accepting the connection, the endpoint will be ready, and the consumer can access the service without traversing the Internet or using a VPN connection.

## Bonus tip

As the last tip, remember that you can **use an application load balancer as a target for a network load balancer**, so you can enable consumers to use an internal service without having to reconfigure the infrastructure.

## To conclude

The AWS ELB services offer great flexibility in terms of choice and configuration. Mixing and matching managed services can help reduce operational complexity and lower costs. We only scratched the surface, and the possibilities are endless.

Did you find other usage scenarios for elastic load balancing? Let us know in the comments below!

See you again in 14 days on **Proud2beCloud**!

---

## About Proud2beCloud

Proud2beCloud is a blog by beSharp, an Italian APN Premier Consulting Partner expert in designing, implementing, and managing complex Cloud infrastructures and advanced services on AWS. Before being writers, we are Cloud Experts working daily with AWS services since 2007. We are hungry readers, innovative builders, and gem-seekers. On Proud2beCloud, we regularly share our best AWS pro tips, configuration insights, in-depth news, tips&tricks, how-tos, and many other resources. Take part in the discussion!

---

**Damiano Giorgi**

Ex on-prem systems engineer, lazy and prone to automating boring tasks. In constant search of technological innovations and new exciting things to experience. And that's why I love Cloud Computing! At this moment, the only "hardware" I regularly dedicate myself to is that my bass; if you can't find me in the office or in the band room try at the pub or at some airport, then!

---