AWS Project: Scalable EC2 Deployment with Load Balancer

This documentation showcases the process, architecture, and validation of deploying a scalable web application using EC2 instances behind a Load Balancer in a private VPC, with Nginx serving a custom page on each node.

Architecture Overview

The solution leverages the following AWS resources:

- Virtual Private Cloud (VPC) with public and private subnets in two Availability Zones
- Linux web servers (EC2) running Nginx in private subnets
- Application Load Balancer distributing HTTP traffic
- Jump server in a public subnet for SSH
- Security group and route table setup for segmentation

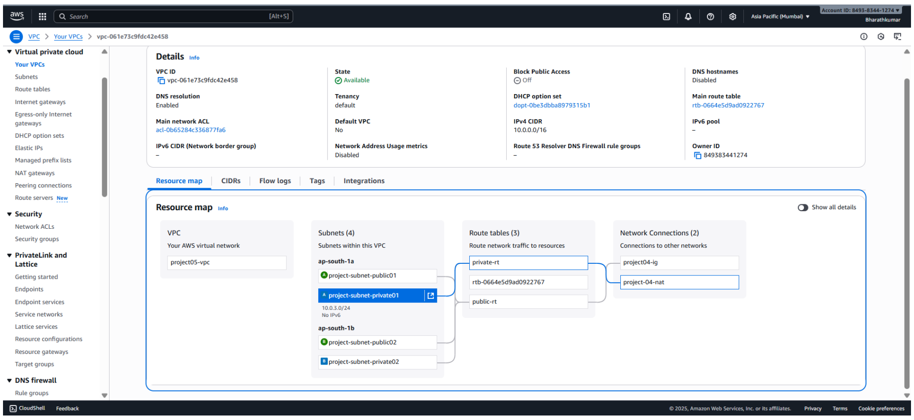
Step-by-Step Process

1. VPC and Subnet Design

Custom VPC with /16 range.

Two public and two private subnets in two Availability Zones.

Route tables and Internet Gateway for connectivity.



VPC > Your VPCs > vpc-061e73c9fdc42e458

Details Info

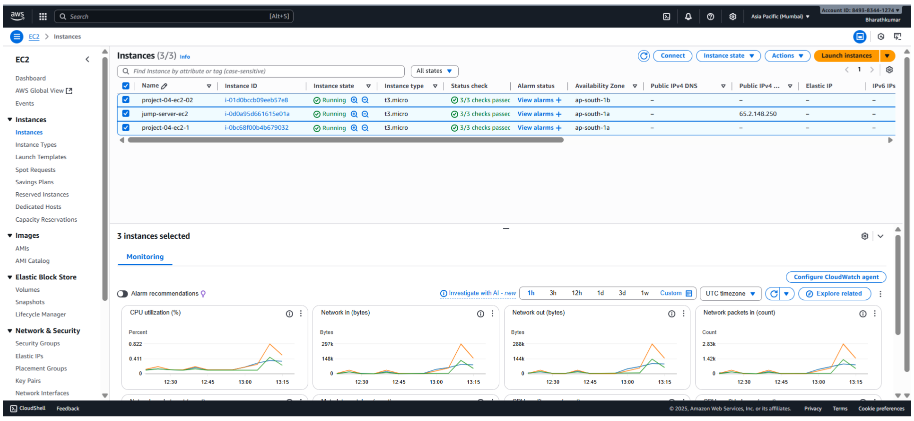
VPC ID: vpc-061e73c9fdc42e458
State: Available
Block Public Access: Off
DNS hostnames: Disabled
DNS resolution: Enabled
Tenancy: default
DHCP option set: dopt-0be3dbba8979315b1
Main route table: rtb-0664e5d9ad0922767
Main network ACL: acl-0b65284c336877fa6
Default VPC: No
IPv4 CIDR: 10.0.0.0/16
IPv6 pool: –
IPv6 CIDR (Network border group): –
Network Address Usage metrics: Disabled
Route 53 Resolver DNS Firewall rule groups: –
Owner ID: 849383441274

2. EC2 Instances Provisioning

Launched two Linux servers in private subnets for Nginx web hosting.

Jump-host in public subnet for SSH access.

Verified instance health and connectivity.



Instances (3/3) Info

Find Instance by attribute or tag (case-sensitive)

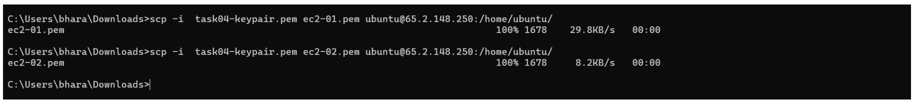
☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs
☑	project-04-ec2-02	i-01d0bccb09eeb57e8	Running	t3.micro	3/3 checks passec	View alarms +	ap-south-1b	–	–	–	–
☑	jump-server-ec2	i-0d0a95d661615e01a	Running	t3.micro	3/3 checks passec	View alarms +	ap-south-1a	–	65.2.148.250	–	–
☑	project-04-ec2-1	i-0bc68f00b4b679032	Running	t3.micro	3/3 checks passec	View alarms +	ap-south-1a	–	–	–	–

3 instances selected

3. Security Group & Key Management

Configured SSH key for secure EC2 login.

Security groups allow SSH to jump host and HTTP from Load Balancer to web servers.



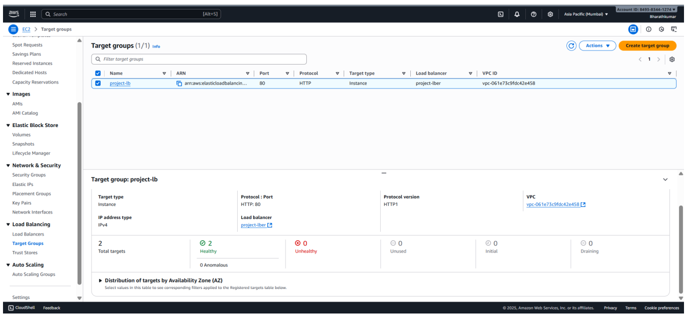
```
C:\Users\bhara\Downloads>scp -i  task04-keypair.pem ec2-01.pem ubuntu@65.2.148.250:/home/ubuntu/
ec2-01.pem                                                                            100% 1678    29.8KB/s   00:00

C:\Users\bhara\Downloads>scp -i  task04-keypair.pem ec2-02.pem ubuntu@65.2.148.250:/home/ubuntu/
ec2-02.pem                                                                            100% 1678     8.2KB/s   00:00

C:\Users\bhara\Downloads>
```

4. Nginx Setup and Validation

Nginx installed and enabled.

Customized default welcome page per instance to confirm load balancing.

Validated service health with systemctl and curl.



Target groups (1/1) Info

Filter target groups

☑	Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
☑	project-lb	arn:aws:elasticloadbalancin...	80	HTTP	Instance	project-lber	vpc-061e73c9fdc42e458

Target group: project-lb

Target type
Instance

Protocol : Port
HTTP: 80

Protocol version
HTTP1

VPC
vpc-061e73c9fdc42e458

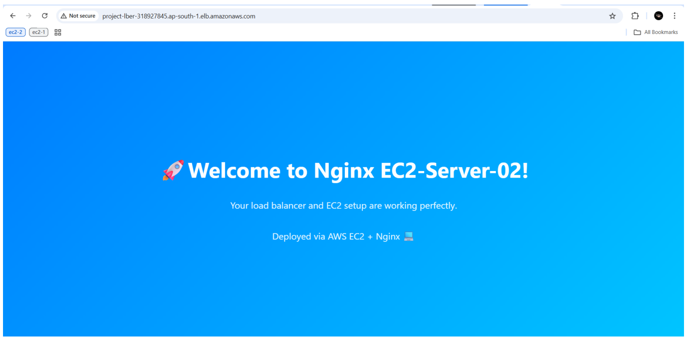
IP address type
IPv4

Load balancer
project-lber

2 Total targets | 2 Healthy | 0 Unhealthy | 0 Unused | 0 Initial | 0 Draining

0 Anomalous

▶ Distribution of targets by Availability Zone (AZ)
Select values in this table to see corresponding filters applied to the Registered targets table below.



🚀 Welcome to Nginx EC2-Server-02!

Your load balancer and EC2 setup are working perfectly.

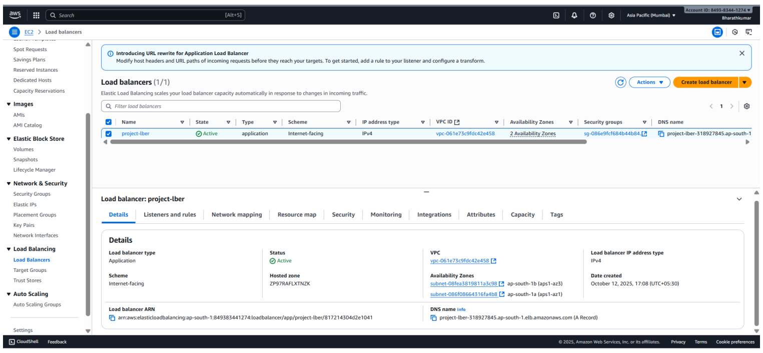
Deployed via AWS EC2 + Nginx 💻

5. Load Balancer Configuration

Deployed Application Load Balancer across Availability Zones.

Registered web servers in target group for HTTP:80 traffic.

Both servers report healthy and serve requests.



Introducing URL rewrite for Application Load Balancer
Modify host headers and URL paths of incoming requests before they reach your targets. To get started, add a rule to your listener and configure a transform.

Load balancers (1/1)

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Name	State	Type	Scheme	IP address type	VPC ID	Availability Zones	Security groups	DNS name
project-lber	Active	application	Internet-facing	IPv4	vpc-061e73c9fdc42e458	2 Availability Zones	sg-086e9fcf684b44b84	project-lber-318927845.ap-south-1

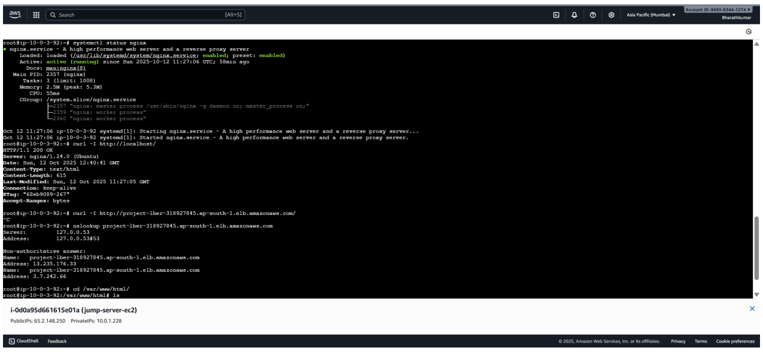
Load balancer: project-lber

Details

Load balancer type Application	Status Active	VPC vpc-061e73c9fdc42e458	Load balancer IP address type IPv4
Scheme Internet-facing	Hosted zone ZP97RAFLXTNZK	Availability Zones subnet-08fea3819811a3c98 ap-south-1b (aps1-az3) subnet-086f08664316fa4b8 ap-south-1a (aps1-az1)	Date created October 12, 2025, 17:08 (UTC+05:30)

Load balancer ARN
arn:aws:elasticloadbalancing:ap-south-1:849383441274:loadbalancer/app/project-lber/817214304d2e1041

DNS name
project-lber-318927845.ap-south-1.elb.amazonaws.com (A Record)

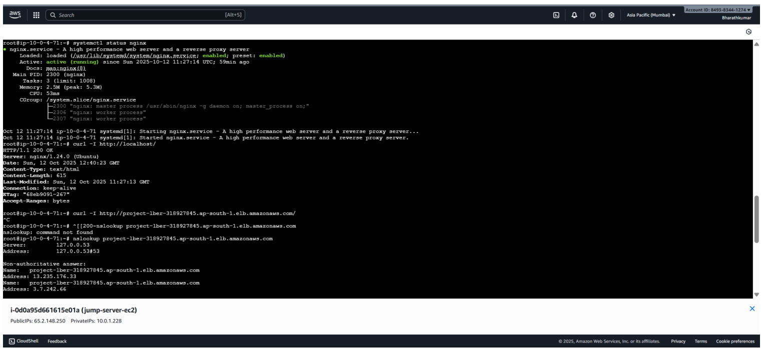


```
root@ip-10-0-3-92:~# systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
     Active: active (running) since Sun 2025-10-12 11:27:06 UTC; 58min ago
       Docs: man:nginx(8)
   Main PID: 2357 (nginx)
      Tasks: 3 (limit: 1008)
     Memory: 2.5M (peak: 5.3M)
        CPU: 55ms
     CGroup: /system.slice/nginx.service
             ├─2357 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             ├─2359 "nginx: worker process"
             └─2360 "nginx: worker process"

Oct 12 11:27:06 ip-10-0-3-92 systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server...
Oct 12 11:27:06 ip-10-0-3-92 systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server.
root@ip-10-0-3-92:~# curl -I http://localhost/
HTTP/1.1 200 OK
Server: nginx/1.24.0 (Ubuntu)
Date: Sun, 12 Oct 2025 12:40:41 GMT
Content-Type: text/html
Content-Length: 615
Last-Modified: Sun, 12 Oct 2025 11:27:05 GMT
Connection: keep-alive
ETag: "68eb9089-267"
Accept-Ranges: bytes

root@ip-10-0-3-92:~# curl -I http://project-lber-318927845.ap-south-1.elb.amazonaws.com/
^C
root@ip-10-0-3-92:~# nslookup project-lber-318927845.ap-south-1.elb.amazonaws.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   project-lber-318927845.ap-south-1.elb.amazonaws.com
Address: 13.235.176.33
Name:   project-lber-318927845.ap-south-1.elb.amazonaws.com
Address: 3.7.242.66

root@ip-10-0-3-92:~# cd /var/www/html/
root@ip-10-0-3-92:/var/www/html# ls
```

i-0d0a95d661615e01a (jump-server-ec2)
PublicIPs: 65.2.148.250 PrivateIPs: 10.0.1.228

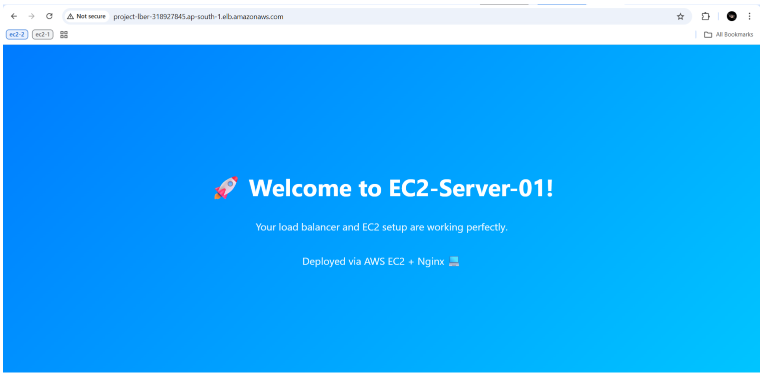
6. End-to-End Validation

Accessed Load Balancer endpoint in browser to confirm unique greeting pages. Successful round-robin routing demonstrated.



```
root@ip-10-0-4-71:~# systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
     Active: active (running) since Sun 2025-10-12 11:27:14 UTC; 59min ago
       Docs: man:nginx(8)
   Main PID: 2300 (nginx)
      Tasks: 3 (limit: 1008)
     Memory: 2.5M (peak: 5.3M)
        CPU: 53ms
     CGroup: /system.slice/nginx.service
             ├─2300 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             ├─2306 "nginx: worker process"
             └─2307 "nginx: worker process"

Oct 12 11:27:14 ip-10-0-4-71 systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server...
Oct 12 11:27:14 ip-10-0-4-71 systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server.
root@ip-10-0-4-71:~# curl -I http://localhost/
HTTP/1.1 200 OK
Server: nginx/1.24.0 (Ubuntu)
Date: Sun, 12 Oct 2025 12:40:23 GMT
Content-Type: text/html
Content-Length: 615
Last-Modified: Sun, 12 Oct 2025 11:27:13 GMT
Connection: keep-alive
ETag: "68eb9091-267"
Accept-Ranges: bytes

root@ip-10-0-4-71:~# curl -I http://project-lber-318927845.ap-south-1.elb.amazonaws.com/
^C
root@ip-10-0-4-71:~# ^[[200~nslookup project-lber-318927845.ap-south-1.elb.amazonaws.com
nslookup: command not found
root@ip-10-0-4-71:~# nslookup project-lber-318927845.ap-south-1.elb.amazonaws.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   project-lber-318927845.ap-south-1.elb.amazonaws.com
Address: 13.235.176.33
Name:   project-lber-318927845.ap-south-1.elb.amazonaws.com
Address: 3.7.242.66
```

i-0d0a95d661615e01a (jump-server-ec2)
PublicIPs: 65.2.148.250 PrivateIPs: 10.0.1.228



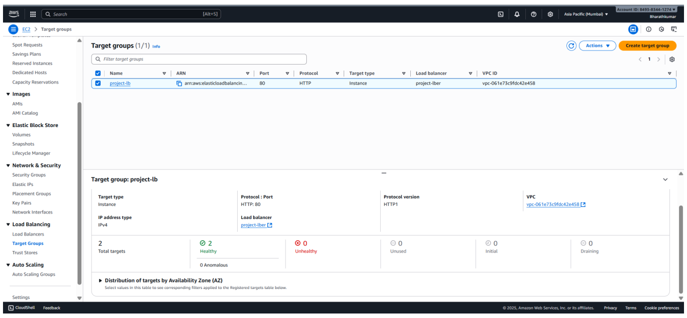
🚀 Welcome to EC2-Server-01!

Your load balancer and EC2 setup are working perfectly.

Deployed via AWS EC2 + Nginx 💻

Additional Validation

DNS resolution and HTTP headers confirmed via CLI from jump server. Network monitoring confirms resource health.



Target groups (1/1) Info

☑	Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
☑	project-lb	arn:aws:elasticloadbalancin...	80	HTTP	Instance	project-lber	vpc-061e73c9fdc42e458

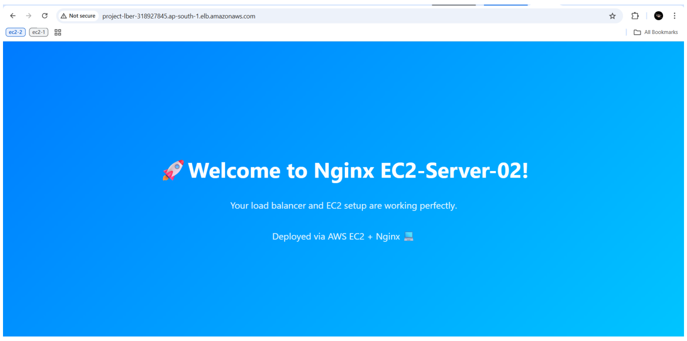
Target group: project-lb

Target type	Protocol : Port	Protocol version	VPC
Instance	HTTP: 80	HTTP1	vpc-061e73c9fdc42e458
IP address type	**Load balancer**		
IPv4	project-lber		

2	2	0	0	0	0
Total targets	Healthy 0 Anomalous	Unhealthy	Unused	Initial	Draining

Distribution of targets by Availability Zone (AZ)

Select values in this table to see corresponding filters applied to the Registered targets table below.



🚀 Welcome to Nginx EC2-Server-02!

Your load balancer and EC2 setup are working perfectly.

Deployed via AWS EC2 + Nginx 💻

Conclusion

This setup demonstrates the principles of secure, resilient architecture on AWS:

- Isolated subnets for security

- Centralized jump host access

- Automated high-availability load balancing

- Web application scaling foundation

All steps are validated by screenshots for reproducibility.