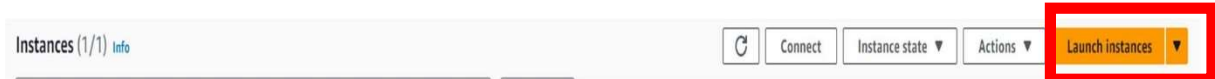
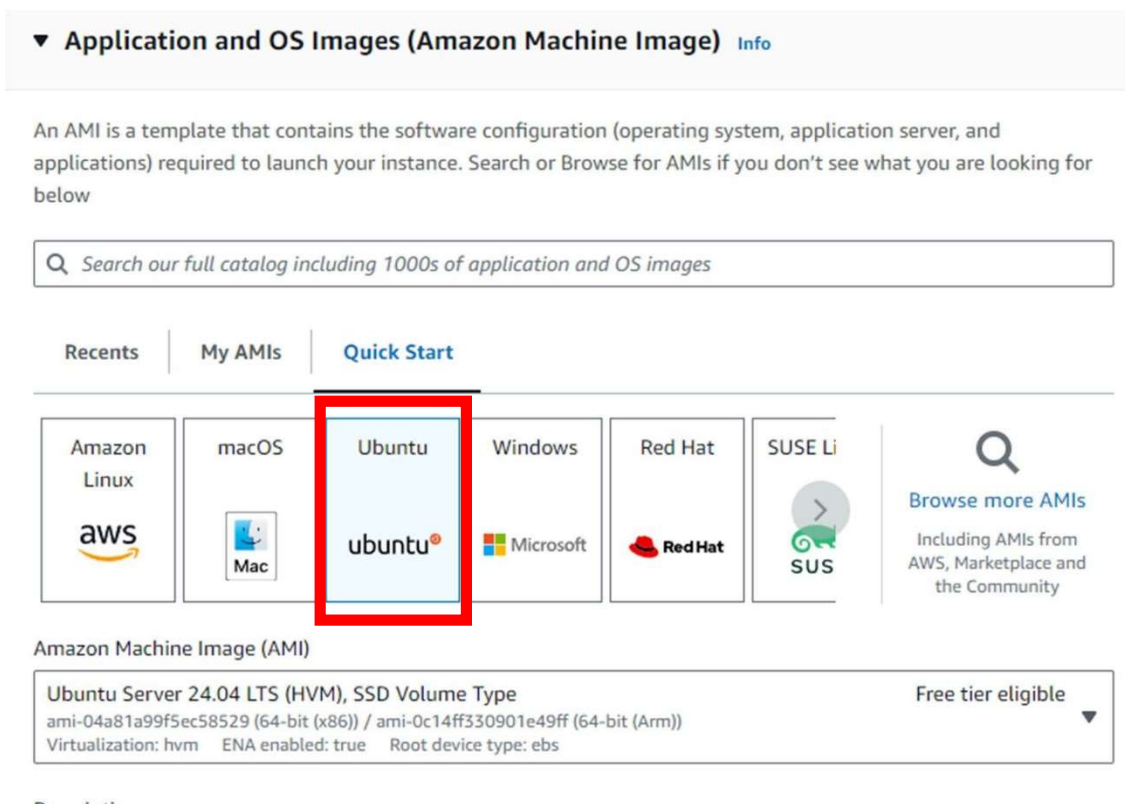


Step 1: Launch an EC2 Instance

1. **Log in to AWS Management Console:**
 - Go to the AWS Management Console at <https://aws.amazon.com/console/>
 - Sign in with your AWS credentials.
2. **Navigate to EC2 Dashboard:**
 - In the AWS Management Console, type "EC2" in the search bar and select EC2 to navigate to the EC2 Dashboard.
3. **Launch an Instance:**
 - Click on the "Launch Instance" button.



- Choose an Amazon Machine Image (AMI): Select "Ubuntu Server 20.04 LTS (HVM), SSD Volume Type".



- Choose an Instance Type: Select t2.micro (eligible for the free tier).

▼ **Instance type** [Info](#) | [Get advice](#)

Instance type

t2.micro

Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Windows base pricing: 0.0162 USD per Hour
On-Demand SUSE base pricing: 0.0116 USD per Hour
On-Demand RHEL base pricing: 0.026 USD per Hour
On-Demand Linux base pricing: 0.0116 USD per Hour

Free tier eligible

☐ All generations
[Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

- Configure Instance:
 - Select an existing key pair or create a new one.
 - Network: Choose the default VPC.
 - Subnet: Choose a subnet in the US-East-1 (N. Virginia) region.
 - Enable Auto-assign Public IP.

▼ **Network settings** [Info](#)

Edit

Network [Info](#)

vpc-01f4dd0a574fc4267

Subnet [Info](#)

No preference (Default subnet in any availability zone)

Auto-assign public IP [Info](#)

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☒ Create security group
☐ Select existing security group

We'll create a new security group called 'launch-wizard-2' with the following rules:

☒ Allow SSH traffic from

Helps you connect to your instance

Anywhere
0.0.0.0/0

☐ Allow HTTPS traffic from the internet

To set up an endpoint, for example when creating a web server

☒ Allow HTTP traffic from the internet

To set up an endpoint, for example when creating a web server

- Add Storage: Keep the default settings.
- Add Tags: Add a tag to identify your instance (e.g., Key: Name, Value: Nginx).

4. **Review and Launch:**
- Review your instance settings and click "Launch".

▼ Summary

Number of instances

Info

1

Software Image (AMI)

Canonical, Ubuntu, 24.04 LTS, ...read more

ami-04a81a99f5ec58529

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

❗ Free tier:

In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

×

Cancel

Launch instance

<input checked="" type="checkbox"/>	Name ↗	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
<input checked="" type="checkbox"/>	Ngix	i-0c00a91976ab448ec	Running 🔍 🔍	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-3-87-207-51.comp...	3.87.207.51	-

5. Configure Security Group:

- Add a new security group with the following rules:
 - Type: HTTP, Protocol: TCP, Port Range: 80, Source: 0.0.0.0/0
 - Type: SSH, Protocol: TCP, Port Range: 22, Source: 0.0.0.0/0

i-0c00a91976ab448ec (Nginx)

▼ Inbound rules

Name	Security group rule ID	Port range	Protocol	Source	Security groups	Description
-	sgr-0c96d25f45f8dbfc	80	TCP	0.0.0.0/0	launch-wizard-1	-
-	sgr-041906bbea0c8558a	22	TCP	0.0.0.0/0	launch-wizard-1	-

Step 2: Connect to Your Instance

1. Connect to the EC2 Instance:

- In the EC2 Dashboard, select your instance.
- Click on "Connect" and follow the instructions to connect to your instance using SSH.

Step 3: Install Nginx

1. Update Package List:

```
sudo apt update
```

2. Install Nginx:

```
sudo apt install nginx -y
```

3. Start and Enable Nginx:

```
sudo systemctl start nginx
sudo systemctl enable nginx
sudo systemctl status nginx
```

```
root@ip-172-31-89-169:/home/ubuntu# 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 Sat 2024-07-20 13:09:49 UTC; 58s ago
     Docs: man:nginx(8)
    Main PID: 2112 (nginx)
      Tasks: 2 (limit: 1130)
    Memory: 1.7M (peak: 1.9M)
       CPU: 10ms
    CGroup: /system.slice/nginx.service
            └─2112 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
               └─2113 "nginx: worker process"

Jul 20 13:09:49 ip-172-31-89-169 systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server...
Jul 20 13:09:49 ip-172-31-89-169 systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server.
root@ip-172-31-89-169:/home/ubuntu#
```

Step 4: Configure Nginx to Display "Hello World"

1. Modify the Default Nginx Webpage:

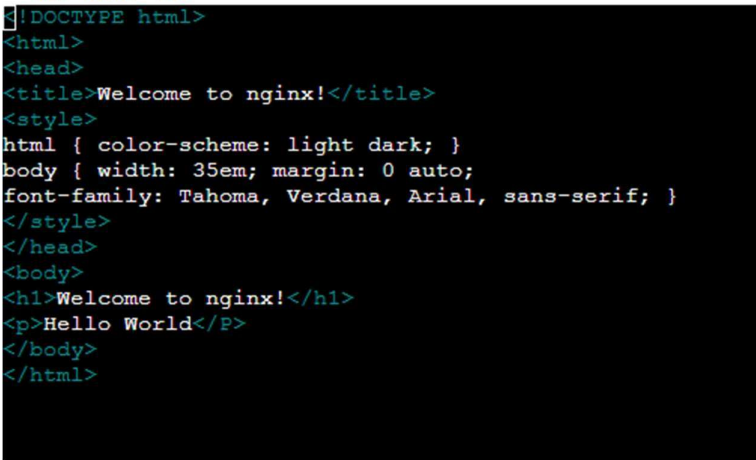
- Open the default Nginx configuration file:

```
sudo nano /var/www/html/index.nginx-debian.html
```

- Replace the content with the following HTML:

```
<!DOCTYPE html>
<html>
<head>
  <title>Welcome to Nginx!</title>
</head>
<body>
  <h1>Hello World</h1>
</body>
</html>
```

GNU nano 7.2



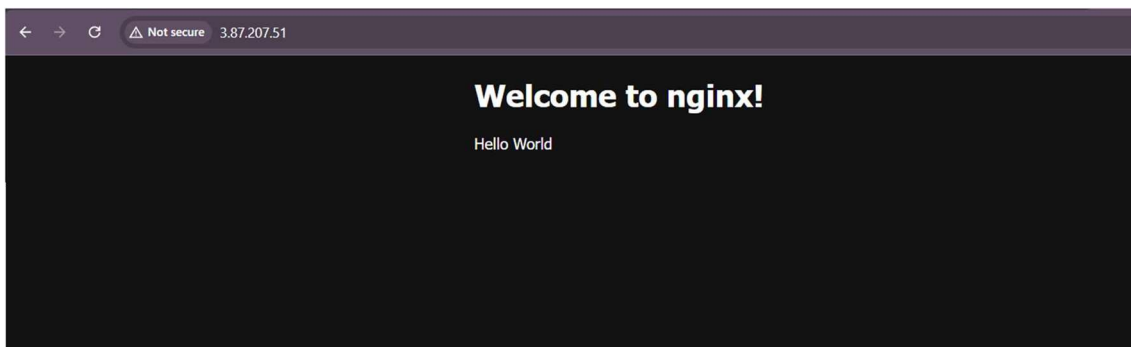
2. Save and Close the File:

- Press `Ctrl + x` to close the file.
- Press `y` to confirm changes, then press `Enter`.

Step 5: Verify the Configuration

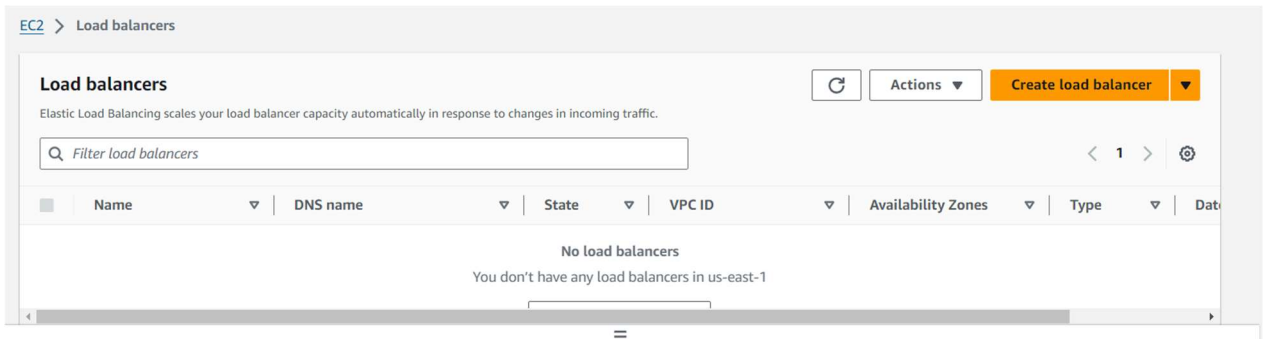
1. Open a Web Browser:

- Enter the public IP address of your EC2 instance in the address bar.
- You should see a webpage displaying the message: "Hello World".

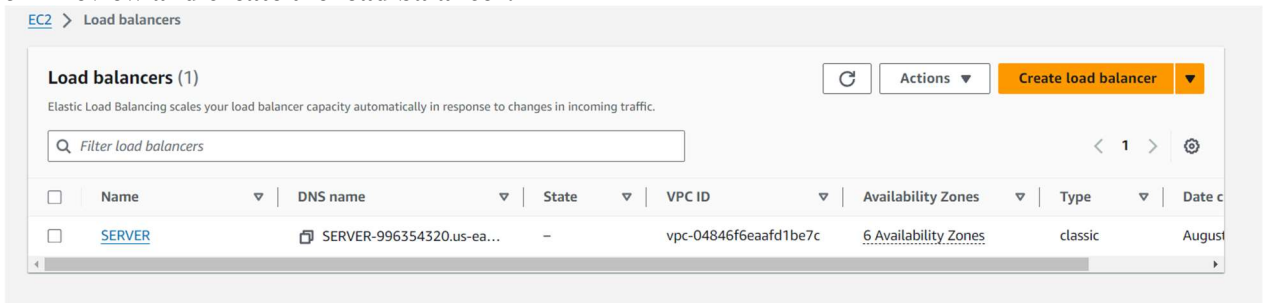


Step 6: Create a Classic Load Balancer

1. **Go to the EC2 Dashboard.**
2. In the left navigation pane, under **Load Balancing**, select **Load Balancers**.

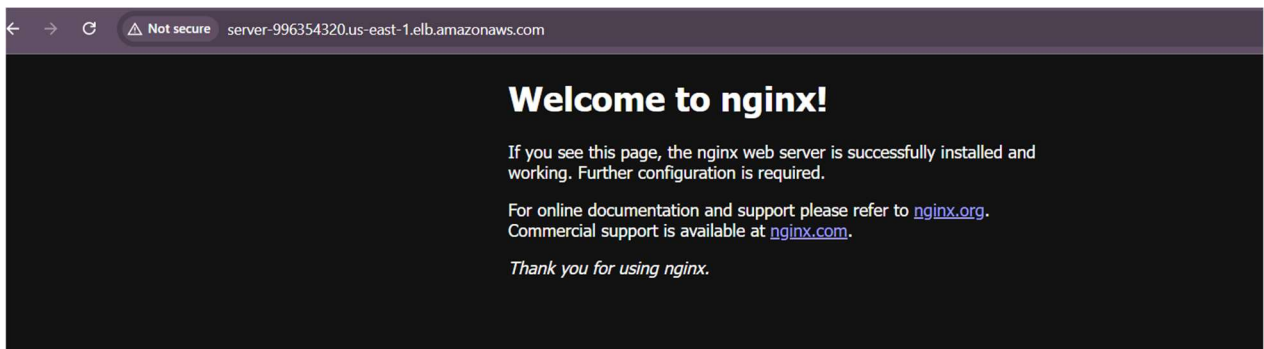


3. Click on **Create Load Balancer** and select **Classic Load Balancer**.
4. **Configure the Load Balancer:**
 - Name your load balancer.
 - Select the VPC and availability zones.
 - Configure listeners (e.g., HTTP on port 80).
 - Configure health checks (e.g., HTTP on /).
 - Select the instances you launched earlier to register with the load balancer.
5. **Review and create the load balancer.**



Step 4: Test the Classic Load Balancer

1. **Get the DNS name of the CLB** from the Load Balancers dashboard.
2. Access the DNS name in your web browser to see the load balancing in action.



Part 2: Migrate to an Application Load Balancer

Step 1: Create an Application Load Balancer

1. **Go to the EC2 Dashboard.**
2. In the left navigation pane, under **Load Balancing**, select **Load Balancers**.
3. Click on **Create Load Balancer** and select **Application Load Balancer**.

Create Application Load Balancer [Info](#)

The Application Load Balancer distributes incoming HTTP and HTTPS traffic across multiple targets such as Amazon EC2 instances, microservices, and containers, based on request attributes. When the load balancer receives a connection request, it evaluates the listener rules in priority order to determine which rule to apply, and if applicable, it selects a target from the target group for the rule action.

► How Application Load Balancers work

Basic configuration

Load balancer name

Name must be unique within your AWS account and can't be changed after the load balancer is created.

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

Scheme [Info](#)

Scheme can't be changed after the load balancer is created.

☒ Internet-facing

An internet-facing load balancer routes requests from clients over the internet to targets. Requires a public subnet. [Learn more](#) [↗](#)

☐ Internal

An internal load balancer routes requests from clients to targets using private IP addresses. Compatible with the IPv4 and Dualstack IP address types.

Load balancer IP address type [Info](#)

Step 2: Create a Load Balancer

1. Navigate to the EC2 Dashboard:

- Click on **Load Balancers** under the Load Balancing section.
- Click on **Create Load Balancer**.
- Choose **Application Load Balancer**.
- Configure the load balancer:
 - Name: my-load-balancer.
 - Scheme: Internet-facing.
 - Listeners: HTTP (port 80).
 - Availability Zones: Select the VPC and subnets.

2. Configure Security Groups for the load balancer:

- Ensure it allows HTTP traffic.

3. Configure Routing:

- Create a target group:
 - Name: my-target-group.
 - Target type: Instances.
 - Protocol: HTTP.
 - Port: 80.
 - Health checks: HTTP.

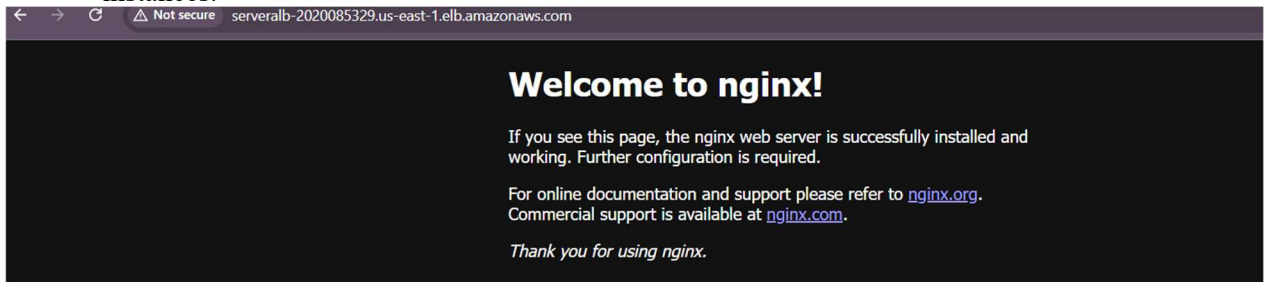
- Register your instances in the target group.
4. **Configure Routing:**
 - Create a new target group.
 - Select Instance as the target type.
 - Register the same EC2 instances you used with the CLB.
 5. **Review and Create** the load balancer.

Step 4: Test the Application Load Balancer

1. **Get the DNS name of the ALB** from the Load Balancers dashboard.

Load balancer ARN ⓘ arn:aws:elasticloadbalancing:us-east-1:016877529802:loadbalancer/app/ServerALB/c1775d93389d35d8	DNS name Info ⓘ ServerALB-2020085329.us-east-1.elb.amazonaws.com (A Record)
--	--

2. Access the DNS name in your web browser to ensure the ALB is routing traffic correctly to your instances.



Step 5: (Optional) Delete the Classic Load Balancer

- Once you have confirmed the ALB is working correctly, you can delete the Classic Load Balancer if it is no longer needed.

Note

- Ensure that your security groups allow inbound traffic on port 80 for the ALB and instances. This process will set up a Classic Load Balancer with registered EC2 instances and then migrate to an Application Load Balancer while ensuring the web pages remain accessible.