**MINUTES SOLUTION: The new domain was not added in the docker compose file where we were initially added the load balancer url for accessing backend through the frontend**

**Problem Statement:**

Your backend application is hosted using **Nginx** as the web server, and traffic is routed through an **AWS Load Balancer**. While the application is accessible via the **Load Balancer's DNS name**, it is **not accessible through the domain name (DNS)** configured for your application.

**Observed Issue:**

* The domain name resolves but does not direct traffic to the application.
* Backend services or APIs may not respond when accessed via the domain name.

**Possible Causes:**

1. **Incorrect DNS Configuration**:
   * DNS record might not be correctly pointing to the load balancer's DNS name.
   * Misconfigured A or CNAME record in the domain's DNS settings.
2. **DNS Propagation Delay**:
   * Recent DNS changes might not have propagated fully across the internet.
3. **Nginx Configuration**:
   * The server\_name in the Nginx configuration may not match the domain name.
   * Nginx might not be correctly handling requests forwarded by the load balancer.
4. **AWS Load Balancer Configuration**:
   * The load balancer's listener or target group may not be configured correctly.
   * Security group rules for the load balancer or backend instances may be blocking traffic.
5. **SSL/TLS Issues (If Using HTTPS)**:
   * SSL certificates may not be correctly set up on the load balancer or Nginx.
6. **Backend Health**:
   * The load balancer may consider backend instances as unhealthy, causing traffic to fail.

**Goal:**

* Ensure that the application is accessible both via the **Load Balancer DNS name** and the **configured domain name**.
* Verify the seamless routing of traffic to the backend services through the domain.

To give DNS to a load balancer, you'll need to configure a domain name to point to the load balancer's DNS name or IP address using a DNS provider. Here's a step-by-step guide:

**1. Obtain the Load Balancer DNS Name or IP Address**

* **For AWS Elastic Load Balancer (ELB):**  
  In the AWS Management Console:
  1. Navigate to **EC2** > **Load Balancers**.
  2. Select your load balancer.
  3. Copy the **DNS name** (e.g., my-load-balancer-1234567890.us-east-1.elb.amazonaws.com).
* **For Other Load Balancers:**  
  Refer to the respective platform documentation to get the public DNS name or IP address of the load balancer.

**2. Choose a Domain Name**

* Decide the domain name you'd like to associate with your load balancer, such as example.com or app.example.com.

**3. Configure DNS with Your Provider**

1. **Log in to Your Domain Registrar or DNS Hosting Provider**  
   (e.g., GoDaddy, Namecheap, AWS Route 53, Cloudflare).
2. **Add a DNS Record:**
   * For **AWS Route 53**:
     1. Go to the **Route 53 Dashboard**.
     2. Choose your hosted zone for the domain.
     3. Click **Create Record**.
        + **Type:** A or CNAME
        + **Alias:** Choose "Yes" if you're using an AWS Load Balancer and select the load balancer from the dropdown.
        + **Target:** Paste the DNS name of your load balancer.
     4. Save the record.
   * For **Other DNS Providers**:
     1. Add a record to your domain’s DNS settings:
        + **Type:**
          - Use A if the load balancer provides an IP address.
          - Use CNAME if the load balancer provides a DNS name.
        + **Name:** Enter the subdomain (e.g., app) or @ for the root domain.
        + **Value:** Enter the DNS name or IP address of your load balancer.
        + **TTL (Time to Live):** Set a desired TTL (e.g., 300 seconds).

**4. Verify DNS Configuration**

* Test the domain using a tool like dig or nslookup to ensure the DNS resolves to the load balancer:

nslookup example.com

* Ensure that navigating to the domain in a browser or using tools like curl hits the load balancer.

**5. Optional: Use SSL Certificates**

If you want your domain to use HTTPS:

* Configure an SSL certificate for your load balancer.
  + For **AWS ELB**, you can attach an SSL certificate via the **ACM (AWS Certificate Manager)**.
  + For other load balancers, refer to their specific SSL setup guides.

**Common Issues and Tips:**

1. **Propagation Delay:** DNS changes can take time (up to 48 hours) to propagate fully, depending on the TTL settings.
2. **Multiple Subdomains:** Use wildcard DNS records (\*.example.com) if managing many subdomains.
3. **Load Balancer Health Checks:** Ensure your load balancer’s health checks are configured to avoid downtime during DNS propagation.

To configure DNS for a load balancer in **AWS**, you can use **Route 53** or another DNS service. Below is a detailed guide for setting up DNS for an **AWS Load Balancer** using **Route 53**:

### ****1. Prerequisites****

* A load balancer is set up in **AWS Elastic Load Balancing (ELB)** (Classic, Application, or Network Load Balancer).
* A domain name registered (either with **Route 53** or another registrar). If you don't have a domain name yet, you can register one via **Route 53**.

### ****2. Get Your Load Balancer’s DNS Name****

1. Open the **AWS Management Console**.
2. Navigate to **EC2** > **Load Balancers**.
3. Select your load balancer.
4. Copy the **DNS name** (e.g., my-load-balancer-1234567890.us-east-1.elb.amazonaws.com).

### ****3. Configure DNS with AWS Route 53****

If you are using **AWS Route 53** as your DNS provider:

#### Step 1: Go to Route 53

1. Open the **Route 53 Console**.
2. Navigate to **Hosted Zones** and select the hosted zone for your domain (e.g., example.com).

#### Step 2: Create a New Record

1. Click **Create Record**.
2. Configure the record settings:
   * **Record Name:** Enter the subdomain you want (e.g., app for app.example.com) or leave it blank for the root domain.
   * **Record Type:**
     + Select A - IPv4 address and enable **Alias** if you're using an AWS Load Balancer.
     + Select CNAME if Alias is not available.
   * **Alias:** Choose **Yes**.
   * **Target:** Choose the load balancer's DNS name from the dropdown list.
3. Save the record.

### ****4. Configure DNS with a Third-Party Provider****

If your domain is managed by another provider (e.g., GoDaddy, Namecheap):

1. **Log in to Your DNS Provider:**
   * Access the DNS management section of your domain.
2. **Add a DNS Record:**
   * **Record Type:**
     + Use A if the load balancer provides an IP address (for Network Load Balancers).
     + Use CNAME if it provides a DNS name (for Application or Classic Load Balancers).
   * **Name:** Enter the subdomain (e.g., app) or @ for the root domain.
   * **Value:** Paste the DNS name of the load balancer.
   * **TTL:** Set a value (e.g., 300 seconds).
3. **Update Nameservers (Optional):**
   * If you are switching to Route 53 as your DNS provider, update your domain's nameservers to the ones provided by Route 53 in your registrar's dashboard.

### ****5. Test the Configuration****

1. Verify the DNS record using dig or nslookup:

nslookup app.example.com

1. Navigate to your domain in a browser (e.g., http://app.example.com) to ensure it resolves to the load balancer.

### ****6. Set Up HTTPS (Optional but Recommended)****

To secure your domain with HTTPS:

1. Use **AWS Certificate Manager (ACM)** to provision an SSL/TLS certificate for your domain.
   * Validate the certificate via DNS or email.
   * Attach the certificate to your load balancer in the **Listener Settings**.

### ****Example DNS Configurations****

#### Example 1: Alias Record for Route 53

yaml

Type: A

Name: app.example.com

Alias: Yes

Target: my-load-balancer-1234567890.us-east-1.elb.amazonaws.com

#### Example 2: CNAME for Third-Party DNS

yaml

Type: CNAME

Name: app.example.com

Value: my-load-balancer-1234567890.us-east-1.elb.amazonaws.com

TTL: 300

### Common Issues and Troubleshooting:

1. **DNS Propagation Delay:** DNS changes may take time (up to 48 hours).
2. **Health Check Failing:** Ensure the load balancer’s health checks are passing for backend instances.
3. **HTTPS Not Working:** Attach an SSL certificate and configure HTTPS in the load balancer's listener settings.
4. **Third-Party Registrar Issues:** Double-check that the DNS record points to the correct load balancer DNS name.

If your backend isn't accessible after configuring the DNS for the load balancer, the issue might stem from one or more of the following factors. Here's a troubleshooting guide:

**1. Verify DNS Configuration**

* Confirm that the domain is resolving to the load balancer's DNS name:

nslookup your-domain.com

dig your-domain.com

* Ensure the correct DNS name or IP address of the load balancer is set in your DNS record.

**2. Check Load Balancer Configuration**

* **Verify Listener Rules:**
  + Confirm that your load balancer has a listener configured for the backend protocol and port (e.g., HTTP:80 or HTTPS:443).
  + In the **AWS Management Console**, go to **EC2 > Load Balancers > Listeners**.
* **Target Group Health Checks:**
  + Go to **Target Groups** in the AWS Console and check if the backend instances are passing health checks.
  + If instances are marked as **unhealthy**, update the health check path to match a valid endpoint (e.g., /health or /).

**3. Security Group Rules**

* Ensure the security group for the load balancer allows traffic on the necessary ports (e.g., HTTP:80, HTTPS:443).
* Ensure the security group for backend instances allows traffic from the load balancer’s security group:
  + Source: sg-<load-balancer-security-group-id>
  + Port: Your backend's listening port (e.g., 3000 for Node.js).

**4. Backend Instance Accessibility**

* Check that the backend service is running on the expected port:

sudo netstat -tuln | grep <port>

* Ensure the backend application is listening on the private IP or 0.0.0.0 (not localhost).

**5. Route 53 or DNS Issues**

* **TTL (Time-to-Live):**
  + DNS records may take time to propagate. Lower the TTL for quicker updates.
* **Record Type:**
  + Ensure you’re using the correct record type (CNAME for Application Load Balancers, A for Network Load Balancers).

**6. Check Firewall and Network ACLs**

* Ensure the Network ACLs associated with your VPC are not blocking inbound or outbound traffic on required ports.

**7. Debug Using Public IP**

* Directly access the load balancer using its DNS name to confirm it's accessible:

curl http://<load-balancer-DNS>

* + If the DNS name works but your domain does not, the issue is in the DNS configuration.
  + If neither works, troubleshoot the load balancer, backend, or network configuration.

**8. Logs and Metrics**

* **Load Balancer Access Logs:**
  + Enable access logging on the load balancer to view incoming requests.
* **Instance Logs:**
  + Check your application logs to identify errors in handling requests.
  + Common locations for logs:
    - Node.js: Check the application log file or the process output.
    - Apache/Nginx: /var/log/apache2 or /var/log/nginx.
* **CloudWatch Metrics:**
  + Monitor request counts, latencies, and health check metrics for the load balancer and target groups.

**9. Cross-Zone Load Balancing**

* Ensure **Cross-Zone Load Balancing** is enabled if your instances span multiple availability zones.

**10. Backend Service Path Configuration**

* Confirm that the path configured in the load balancer matches the backend application’s endpoints.
  + Example: If your health check path is /status, ensure the backend serves this path.

**Example Checklist**

1. **Load Balancer:** DNS resolves to the load balancer DNS name.
2. **Target Group:** Backend instances are registered and healthy.
3. **Security Groups:** Proper inbound/outbound rules are configured.
4. **Backend:** Service is running and accessible on the correct port.
5. **Logs:** No errors in access logs, backend logs, or CloudWatch metrics.

If you're using **Nginx** as the server for your backend and it's not accessible via the load balancer and DNS, the issue may lie in the **Nginx configuration**, **load balancer setup**, or **networking rules**. Here's how to troubleshoot and resolve the issue:

**1. Verify Nginx Configuration**

Ensure that **Nginx** is configured correctly to handle requests.

* Check if **Nginx** is running:

sudo systemctl status nginx

* + If it's not running, start it:

sudo systemctl start nginx

* Verify the Nginx configuration file (e.g., /etc/nginx/nginx.conf or /etc/nginx/sites-enabled/default):
  + Ensure it's listening on the right port (e.g., 80 for HTTP or 443 for HTTPS).
  + Example for a basic Nginx configuration:

nginx

server {

listen 80;

server\_name your-domain.com;

location / {

proxy\_pass http://localhost:3000; # Backend application port

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

}

}

* Test the Nginx configuration:

sudo nginx -t

* + If there are errors, fix them, then reload Nginx:

sudo systemctl reload nginx

**2. Ensure Backend Service is Running**

Verify that the application behind Nginx is running and accessible:

curl http://localhost:3000

Replace 3000 with the port your backend application uses.

* If the service is not responding:
  + Check logs for errors in the application or Nginx (/var/log/nginx/error.log).
  + Restart the backend service.

**3. Load Balancer Listener Configuration**

Ensure the load balancer forwards traffic to Nginx:

1. Go to **AWS Management Console** > **EC2** > **Load Balancers**.
2. Check the **Listeners** tab:
   * Listener: HTTP (port 80) or HTTPS (port 443).
   * Target Group: Points to your backend instances.

**4. Health Check Configuration**

Ensure the load balancer's health check is properly configured to verify the Nginx server:

1. Go to **Target Groups** in AWS Console.
2. Select the target group associated with your backend.
3. Edit **Health Check Settings**:
   * Protocol: HTTP
   * Path: / (or another path that Nginx serves, like /health).
   * Port: The port where Nginx listens (e.g., 80).
4. Test health checks:
   * Run curl to check the health check path from the instance itself:

curl http://localhost/

**5. Security Groups for Nginx**

Ensure the security groups for the instance running Nginx allow traffic from the load balancer:

* **Inbound Rules:**
  + Source: sg-<load-balancer-security-group-id> (load balancer security group).
  + Port: 80 (HTTP) or 443 (HTTPS).
* **Outbound Rules:**
  + Allow all traffic or specific ports for responses (e.g., 80, 443).

**6. DNS Configuration**

* Ensure the DNS record points to the load balancer's DNS name.
* Verify that DNS changes have propagated:

nslookup your-domain.com

**7. Logs for Debugging**

* Check Nginx logs for errors:
  + Access logs: /var/log/nginx/access.log
  + Error logs: /var/log/nginx/error.log
* Check **Load Balancer** access logs (if enabled) to see incoming requests.

**Example Nginx Configuration for Load Balancer**

Below is an example configuration for an **Nginx backend** receiving traffic from an **AWS Application Load Balancer**:

nginx

server {

listen 80;

server\_name your-domain.com;

# Handle traffic from the load balancer

location / {

proxy\_pass http://localhost:3000; # Backend service

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

}

}

**Common Issues and Solutions**

1. **Health Check Fails:**
   * Ensure Nginx serves the health check path and returns a 200 OK status.
2. **Connection Timeout:**
   * Check backend application status and security groups.
3. **502 Bad Gateway:**
   * Nginx cannot reach the backend service.
   * Verify proxy\_pass URL and backend service status.
4. **DNS Not Resolving:**
   * Ensure DNS record points to the load balancer and has propagated.