**Project: Setting Up HTTP Server and Load Balancer**

**Objective**:

To set up a Python HTTP server on two servers, serve an HTML file, and configure a load balancer to distribute traffic between the two servers.

----------------------

**1. Prerequisites**

**Two Servers**: We are using two Ubuntu servers. Both servers should have Python 3 installed.

**Bastion Host**: A Bastion host for external access to the private servers.

**Load Balancer**: A load balancer for distributing traffic across the two servers.

----------------------

**2. Setting Up HTTP Server on Both Servers**

**Step 1: Install Python 3 (If not already installed)**

On both servers, we ensured that Python 3 was installed. If not, we installed it using the following command:

bash

CopyEdit

sudo apt update sudo apt install python3

**Step 2: Start Python HTTP Server**

To start a simple HTTP server on both servers, we ran the following command:

bash

CopyEdit

python3 -m http.server 8000

This starts an HTTP server on port 8000. Initially, it listens on 127.0.0.1, which means it can only be accessed locally.

**Step 3: Expose the Server Externally**

To make the server accessible externally (from other machines, like the Bastion or the load balancer), we used the following command to bind the server to all network interfaces:

bash

CopyEdit

python3 -m http.server 8000 --bind0.0.0.0

This ensures that the server is accessible via the private IP of the server and also from the Bastion host.

----------------------

**3. Uploading the HTML File**

**Step 1: Upload the HTML File to Both Servers**

I uploaded the index1.html file from the local machine to both servers using scp:

bash

CopyEdit

scp /path/to/index1.html ubuntu@<server-ip>:/home/ubuntu/

After uploading the file, I checked the file’s existence on both servers using:

bash

CopyEdit

ls/home/ubuntu

**Step 2: Fix Permissions**

On the second server, the index1.html file was not being displayed, and the server was returning a directory listing. This was due to incorrect file permissions. I fixed this by running:

bash

CopyEdit

sudo chmod644 /home/ubuntu/index1.html

This ensures that the HTTP server has permission to read the file.

----------------------

**4. Verifying Server and File Accessibility**

**Step 1: Local Testing on Each Server**

I tested the server locally on each machine by running:

bash

CopyEdit

curl http://localhost:8000/index1.html

This should return the content of the index1.html file.

**Step 2: Test from External Machine (Bastion Host)**

I tested the accessibility from the Bastion host using the following curl command:

bash

CopyEdit

curl http://<bastion-ip>:8000/index1.html

This confirmed that the server was correctly serving the HTML file to external clients.

----------------------

**5. Troubleshooting**

**Issue 1: File Not Displaying Correctly**

**Problem**: On one server, the content of index1.html was not being served, and the server was returning a directory listing.

**Resolution**: I checked the file's permissions using ls -l /home/ubuntu/index1.html. I found that the file permissions were incorrect. I resolved this by running:

bash

CopyEdit

sudo chmod644 /home/ubuntu/index1.html

This ensured the server could read the file and serve it correctly.

**Issue 2: Server Not Exposing HTTP Server Externally**

**Problem**: The server was initially bound to 127.0.0.1, making it accessible only from localhost.

**Resolution**: I used the --bind 0.0.0.0 flag when starting the server:

bash

CopyEdit

python3 -m http.server 8000 --bind0.0.0.0

**Issue 3: Connection Refused on Localhost**

**Problem**: When trying to test the server using curl http://localhost:8000/index1.html, I received a "Connection refused" error.

**Resolution**: I verified that the server was running by checking the process list using:

bash

CopyEdit

ps -ef | grep http.server

The server was running, but not bound to the correct network interface. I restarted the server with --bind 0.0.0.0 to fix this.

----------------------

**6. Configuring Load Balancer**

**Step 1: Set Up Load Balancer**

I configured a simple load balancer that forwarded traffic to both servers using their private IP addresses. The load balancer’s job is to balance the traffic between both servers.

**Step 2: Testing Load Balancer**

I tested the load balancer by accessing its DNS or IP from the browser or using curl. The load balancer should distribute requests between the two servers. If properly configured, each server should serve the HTML file.

**Problem**: Initially, only one server responded to the load balancer's traffic.

**Resolution**: I reviewed the load balancer settings and ensured that both servers were registered as healthy backends. I also checked that the load balancer was distributing traffic correctly across both servers.

----------------------

**7. Final Verification**

After resolving the issues with file permissions, server bindings, and load balancing, I conducted the final verification:

Both servers were accessible via their respective private IP addresses.

The index1.html file was correctly served by both servers.

The load balancer was distributing traffic between the servers, and each server was responding as expected.

----------------------

**Problems Faced and Solutions**

1. **Server Not Exposing HTTP Server Externally**:
   * **Cause**: Python HTTP server was bound to 127.0.0.1 by default.
   * **Solution**: Used --bind 0.0.0.0 to bind the server to all available interfaces.
2. **File Not Displaying Correctly**:
   * **Cause**: Incorrect file permissions on index1.html.
   * **Solution**: Used chmod 644 to fix file permissions.
3. **Connection Refused on Localhost**:
   * **Cause**: The server wasn’t listening on the correct network interface.
   * **Solution**: Restarted the HTTP server with the correct bind address (--bind 0.0.0.0).
4. **Load Balancer Not Distributing Traffic Properly**:
   * **Cause**: Incorrect load balancer configuration, with only one server receiving traffic.
   * **Solution**: Reviewed and updated the load balancer configuration to include both servers and balance traffic properly.

----------------------

**Key Learnings**

**Networking**: Ensuring that services listen on the correct interfaces (0.0.0.0) is essential for external accessibility.

**Permissions**: File permissions can cause issues with serving content. Proper file permissions (readable by the server) are a must.

**Load Balancer Configuration**: Correctly setting up the load balancer is key to evenly distributing traffic across servers.

**Troubleshooting Skills**: The process involved a lot of troubleshooting, especially in verifying that the servers were running correctly and that traffic was routed properly.

----------------------

This detailed document covers the entire procedure, the problems faced, and how you solved them. It provides a clear and structured explanation of your approach and will be helpful in explaining the project during an interview.