Setup Guide: Automated HAProxy with Ansible Dynamic Inventory on AWS

1. Project Overview

This guide provides a complete walkthrough for building a high-availability web service architecture on AWS. It features an HAProxy load balancer that uses Ansible's dynamic inventory to automatically discover and configure its backend web server pool. The system adapts to infrastructure changes (like adding or removing servers) in real-time without manual intervention.

2. Prerequisites

Before you begin, ensure you have the following:

- **An AWS Account:** With IAM permissions to create VPCs, EC2 instances, IAM Roles, and Security Groups.
- **A Local Terminal:** A command-line interface like **Git Bash** (for Windows) or any standard terminal (for macOS/Linux).
- **An SSH Key Pair:** An SSH key already generated and available on your local machine. You will need to upload the public key to AWS.

Phase 1: Setting Up the AWS Network Foundation

We will create a secure, isolated network (VPC) for our servers.

1.1 Create the VPC

1. Navigate to the **VPC** service in the AWS Console.
2. Click **"Create VPC"**.
3. Select **"VPC only"**.
4. Configure as follows:
   - **Name tag:** `haproxy-vpc`
   - **IPv4 CIDR block:** `10.0.0.0/16`
5. Click **"Create VPC"**.

1.2 Create a Public Subnet

1. In the VPC dashboard, go to **"Subnets"** and click **"Create subnet"**.
2. Select the `haproxy-vpc`.
3. Configure as follows:
   - **Subnet name:** `haproxy-public-subnet`
   - **Availability Zone:** Choose any one (e.g., `us-east-1a`).

    - **IPv4 CIDR block:** `10.0.1.0/24`
4. Click **"Create subnet"**.
5. After creation, select the subnet, click **"Actions"** -> **"Edit subnet settings"**, and check the box for **"Enable auto-assign public IPv4 address"**. Save changes.

1.3 Create and Attach an Internet Gateway

1. In the VPC dashboard, go to **"Internet Gateways"** and click **"Create internet gateway"**.
2. **Name tag:** `haproxy-igw`
3. Click **"Create internet gateway"**.
4. After creation, select it, click **"Actions"** -> **"Attach to VPC"**, and select `haproxy-vpc`.

1.4 Configure a Route Table

1. In the VPC dashboard, go to **"Route Tables"**.
2. Select the main route table associated with `haproxy-vpc`.
3. Go to the **"Routes"** tab -> **"Edit routes"** -> **"Add route"**.
4. Configure the new route:
    - **Destination:** `0.0.0.0/0`
    - **Target:** Select "Internet Gateway" and choose `haproxy-igw`.
5. Save changes.

1.5 Create a Security Group

1. In the VPC dashboard, go to **"Security Groups"** and click **"Create security group"**.
2. Configure the basic details:
    - **Security group name:** `haproxy-sg`
    - **Description:** `Allows SSH, HTTP, and internal traffic`
    - **VPC:** Select `haproxy-vpc`.
3. Add the following three **Inbound rules**:
    - **Rule 1:** `Type: SSH`, `Source: My IP`
    - **Rule 2:** `Type: HTTP`, `Source: Anywhere-IPv4 (0.0.0.0/0)`
    - **Rule 3:** `Type: All traffic`, `Source: Custom -> haproxy-sg` (select the group itself)
4. Click **"Create security group"**.

Phase 2: Provisioning the EC2 Instances

2.1 Launch the Control Node (Ansible + HAProxy)

1. Navigate to the **EC2** service and click **"Launch instances"**.
2. Configure as follows:
    - **Name:** `haproxy-control-node`

- **AMI:** Ubuntu Server 24.04 LTS
- **Instance type:** `t3.small`
- **Key pair:** Select your pre-existing SSH key pair.
- **Network settings (Edit):**
    - **VPC:** `haproxy-vpc`
    - **Subnet:** `haproxy-public-subnet`
    - **Firewall:** Select existing security group -> `haproxy-sg`

3. Click **"Launch instance"**.

2.2 Launch the Web Servers

Launch **two** identical instances with the following configuration.

1. Navigate to the **EC2** service and click **"Launch instances"**.
2. Configure as follows:
    - **Name:** `web-server-01` (and `web-server-02` for the second one).
    - **AMI:** Ubuntu Server 24.04 LTS
    - **Instance type:** `t3.small`
    - **Tags (Advanced details):** Click "Add tag" and add the following:
        - **Key:** `Role`
        - **Value:** `WebServer`
    - **Key pair:** Select the same SSH key pair.
    - **Network settings (Edit):**
        - **VPC:** `haproxy-vpc`
        - **Subnet:** `haproxy-public-subnet`
        - **Firewall:** Select existing security group -> `haproxy-sg`
3. Click **"Launch instance"**. Repeat for the second server.

Phase 3: Configuring the Control Node

3.1 Connect to the Control Node

From your local terminal, use SSH to connect to your `haproxy-control-node` instance.

```
ssh -i "path/to/your-key.pem" ubuntu@<control-node-public-ip>
```

3.2 Update and Install Software

Run the following commands on the control node:

```
# Update system packages
sudo apt update && sudo apt upgrade -y

# Add Ansible repository and install Ansible
sudo add-apt-repository --yes --update ppa:ansible/ansible
sudo apt install ansible -y

# Install HAProxy
sudo apt install haproxy -y
```

3.3 Place the Private Key on the Control Node

⚠️ **SECURITY WARNING:** This key allows access to your other servers. Protect it carefully.

1. On your **local computer**, copy the entire content of your `.pem` private key file.
2. On the **control node**, create a new file: `nano ~/.ssh/project_key.pem`.
3. **Paste** the key content into the nano editor and save it.
4. Set strict permissions on the key file: `chmod 400 ~/.ssh/project_key.pem`.

Phase 4: Building the Ansible Automation Suite

All the following steps are performed on the **control node**.

4.1 Grant AWS Permissions with an IAM Role

1. In the AWS **IAM** Console, go to **Roles** -> **"Create role"**.
2. **Trusted entity:** AWS service, **Use case:** EC2.
3. **Permissions:** Search for and add `AmazonEC2ReadOnlyAccess`.
4. **Role name:** `ansible-ec2-discovery-role` and create the role.
5. Go back to the **EC2 Console**, select your `haproxy-control-node`, and go to **Actions** -> **Security** -> **Modify IAM role**.
6. Attach the `ansible-ec2-discovery-role`.

4.2 Create the Project Directory and Files

On the control node, create a directory for your project and create the necessary files inside it.

```
mkdir ~/ansible_project
cd ~/ansible_project
```

File 1: `ansible.cfg` (Ansible configuration)

```
nano ansible.cfg
```

Paste the following:

```
[defaults]
inventory = aws_ec2.yml
host_key_checking = False
private_key_file = ~/.ssh/project_key.pem
remote_user = ubuntu
```

```
[inventory]
enable_plugins = aws_ec2
```

File 2: `aws_ec2.yml` (Dynamic Inventory)

```
nano aws_ec2.yml
```

Paste the following, **making sure to change the `regions` value to your AWS region**:

```
---
plugin: aws_ec2
regions:
  - us-east-1 # <-- CHANGE THIS to your AWS region
keyed_groups:
  - key: tags.Role
    prefix: role
compose:
  ansible_host: private_ip_address
```

File 3: `haproxy.cfg.j2` (HAProxy Template)

```
nano haproxy.cfg.j2
```

Paste the following:

```
global
    log /dev/log    local0
    daemon

defaults
    log     global
    mode    http
    timeout connect 5000
    timeout client  50000
    timeout server  50000

frontend http_front
    bind *:80
    default_backend http_back

backend http_back
    balance roundrobin
    {% for host in groups['role_WebServer'] %}
    server {{ hostvars[host].tags.Name }} {{ hostvars[host].ansible_host
}}:80 check
    {% endfor %}
```

File 4: `haproxy_playbook.yml` (HAProxy Playbook)

```
nano haproxy_playbook.yml
```

Paste the following:

```
---
- name: Configure HAProxy Load Balancer
  hosts: localhost
  connection: local
  become: yes

  tasks:
    - name: Generate HAProxy configuration file from template
      ansible.builtin.template:
        src: haproxy.cfg.j2
        dest: /etc/haproxy/haproxy.cfg
      notify:
        - Reload HAProxy

  handlers:
    - name: Reload HAProxy
      ansible.builtin.service:
        name: haproxy
        state: reloaded
```

File 5: `index.html.j2` (Webpage Template)

```
nano index.html.j2
```

Paste the following:

```
<h1>Hello from Web Server</h1>
<h2>Hostname: {{ ansible_hostname }}</h2>
<h3>Private IP: {{ ansible_default_ipv4.address }}</h3>
```

File 6: `deploy_webpage.yml` (Webpage Playbook)

```
nano deploy_webpage.yml
```

Paste the following:

```
---
- name: Deploy custom web page to web servers
  hosts: role_WebServer
  become: yes

  tasks:
    - name: Ensure apache2 is installed
      ansible.builtin.apt:
        name: apache2
        state: present
    - name: Create custom index.html from template
      ansible.builtin.template:
        src: index.html.j2
        dest: /var/www/html/index.html
```

Phase 5: Deployment and Verification

Run the playbooks from your `~/ansible_project` directory on the control node.

1. **Deploy the web pages to your web servers:**

   ```
   ansible-playbook deploy_webpage.yml
   ```

2. **Configure HAProxy:**

   ```
   ansible-playbook haproxy_playbook.yml
   ```

3. **Verify from your local machine:** Open a new local terminal (not the SSH session) and run the `curl` loop. You should see the output alternate between your two web servers.

   ```
   for i in {1..4}; do curl http://<control-node-public-ip>; echo ""; done
   ```

Phase 6: Test Dynamic Scaling

1. Go to the AWS Console and launch a **third** EC2 instance (`web-server-03`) with the tag `Role: WebServer`.
2. Wait for it to be in the "Running" state.
3. On the control node, re-run both playbooks:

   ```
   ansible-playbook deploy_webpage.yml
   ansible-playbook haproxy_playbook.yml
   ```

4. From your local machine, run the `curl` loop again. You will now see traffic being balanced across all three servers.

Conclusion

You have successfully created a self-managing, dynamically scaling web architecture. You can now add or remove backend servers, and the system will adapt automatically by re-running the Ansible playbooks.