Demo – Enable Load Balancing

Nicolas El Khoury

Introduction

To better understand load balancing, we are going to:

- Create three Ubuntu EC2 machines on AWS.
- Create two simple HTML pages.
- Deploy each application on one VM.
- Create and configure a load balancer on the third VM, and instruct it to distribute the requests on the two machines.

The first application is the following HTML Page:

```
<!DOCTYPE html>
<html>
    <head>
        <title>My First Application</title>
    </head>
    <body>
        <p>I have no idea what I'm doing.</p>
```

```
    </body>
</html>
```

The second application is the following HTML Page:

```
<!DOCTYPE html>
<html>
    <head>
        <title>My Second Application</title>
    </head>
    <body>
        <p>Neither do I.</p>
    </body>
</html>
```

Solution

AWS Networking and Compute Resources

SSH Keypair

1. Navigate to the **EC2** service, **Key Pairs** option from the left menu.
2. Create a Keypair.
3. The key will be automatically downloaded. Move it to a hidden directory.
4. Modify the permissions to read only: `chmod 400 <keyName>.pem`

Security Group

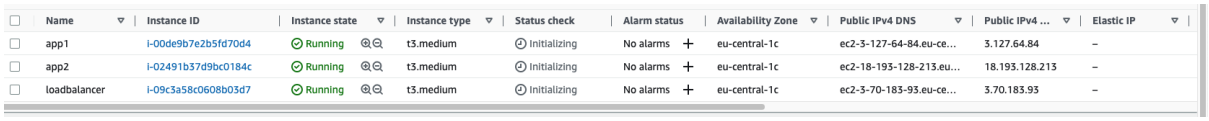
1. Navigate to the **Security group** option from the left menu.
2. Specify a name.
3. Attach it to the default VPC.
4. Enable ports **22** and **80** to all IPv4 addresses.

EC2 Instances

Create three `Ubuntu 20.04` VMs:

- Navigate to **AWS EC2** —> **instances** —> **Launch instances**, with the following parameters:

- **Name**: app1 | app2 | loadbalancer *(each name corresponds to one VM)*
- **AMI**: Ubuntu Server 20.04 LTS (HVM), SSD Volume Type
- **Instance Type**: t3.medium *(Or any type of your choice)*
- **Key pair name**: aws-demo
- **Network Settings**:
  - **Select existing security group**: aws-demo
- **Configure storage**: 1 x 25 GiB gp2 Root volume



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
app1	i-00de9b7e2b5fd70d4	Running	t3.medium	Initializing	No alarms	eu-central-1c	ec2-3-127-64-84.eu-ce...	3.127.64.84	–
app2	i-02491b37d9bc0184c	Running	t3.medium	Initializing	No alarms	eu-central-1c	ec2-18-193-128-213.eu...	18.193.128.213	–
loadbalancer	i-09c3a58c0608b03d7	Running	t3.medium	Initializing	No alarms	eu-central-1c	ec2-3-70-183-93.eu-ce...	3.70.183.93	–

App1 Server configuration

Apache2 Installation

1. Update the local package index to reflect the latest upstream changes: `sudo apt-get update`
2. Install the **Apache2** package: `sudo apt-get install -y apache2`

Application Deployment

a. Perform the following steps to deploy the first application:

```
# Create a directory
sudo mkdir /var/www/myapp
# Change the owership to www-data
sudo chown -R www-data:www-data /var/www/myapp
# Change the directory permissions
sudo chmod -R 755 /var/www/myapp
# Create the index.html file and paste the code of the first app in it
sudo nano /var/www/myapp/index.html
# Create the log directory
sudo mkdir /var/log/myapp
# Change the ownership of the directory
sudo chown -R www-data:www-data /var/log/myapp/
```

Virtualhost Configuration

- Create the virtual host file: `sudo nano /etc/apache2/sites-available/myapp.conf`
- Paste the following configuration:

```
<VirtualHost *:80>
    DocumentRoot /var/www/myapp
    ErrorLog /var/log/myapp/error.log
    CustomLog /var/log/myapp/requests.log combined
</VirtualHost>
```

- Enable the configuration:

```
# Enable the site configuration
sudo a2ensite myapp.conf
# Disable the default configuration
sudo a2dissite 000-default.conf
# Test the configuration
sudo apache2ctl configtest
# Restart apache
sudo systemctl restart apache2
```

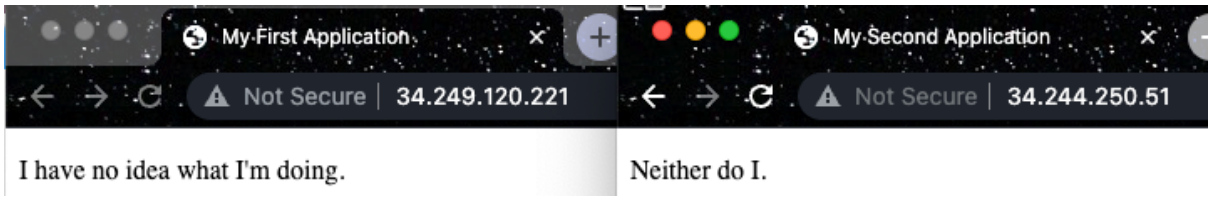
- Perform a request on the server to ensure that the configuration is done properly.

App2 Server configuration

- Repeat the same steps exactly on the second server to:
  - Install Apache2.
  - Deploy the application.
  - Create the virtual host.
  - Test a request

Server testing

Perform a request on the server to ensure that the configuration is done properly.



Load Balancer Configuration

Perform the following on the **loadbalancer** VM:

Apache2 Installation

1. Update the local package index to reflect the latest upstream changes: `sudo apt-get update`
2. Install the **Apache2** package: `sudo apt-get install -y apache2`

Apache2 Configuration

- Install the required modules

```
sudo a2enmod proxy
sudo a2enmod proxy_http
sudo a2enmod proxy_balancer
sudo a2enmod lbmethod_byrequests
sudo a2enmod headers
```

- Create the virtual host: `sudo nano /etc/apache2/sites-available/lbmanager.conf`
- Paste the following configuration:

```
<VirtualHost *:80>
    <Proxy balancer://myservers>
        BalancerMember http://<APP1 IP>:80
        BalancerMember http://<APP2 IP>:80
    </Proxy>

    ProxyPass "/" "balancer://myservers/"
    ProxyPassReverse "/" "balancer://myservers/"
</VirtualHost>
```

- Enable the configuration:

```
# Disable the default configuration
sudo a2dissite 000-default.conf
# Enable the lbmanager
sudo a2ensite lbmanager.conf
# Test the configuration
sudo apache2ctl configtest
# Restart apache
sudo systemctl restart apache2
```

- Restart Apache2: `sudo service apache2 restart`

Application Testing

- Entering the IP of the load balancer should balance the load between the two machines: