Buatlah Vagrantfile sekaligus provisioning-nya untuk menyelesaikan kasus tersebut.

### Membuat 3 Vagrant (1 Load Balancer dan 2 Worker)

#### Membuat folder untuk ketiga Vagrant

##### # loadbalancer : loadbalancer | worker : worker-1 & worker-2

##### mkdir loadbalancer worker-1 worker-2

### Inisiasi Project Vagrant pada setiap folder Vagrant

vagrant init

Menambahkan box Ubuntu 16.04 Xenial ke Vagrant

vagrant box add ubuntu/xenial64

Konfigurasi file Vagrantfile

nano Vagrantfile

Menggunakan box Ubuntu 16.04 Xenial

config.vm.box = "ubuntu/xenial64"

Mengaktifkan private networking pada setiap Vagrantfile

config.vm.network "private\_network", ip: "xxx.xxx.xxx.xxx"

**Keterangan:**

Loadbalancer (192.168.0.2)

Worker 1 (192.168.0.3)

Worker 2 (192.168.0.4)

Mengaktifkan Provisioning

config.vm.provision :shell, path: "bootstrap.sh"

#### Konfigurasi Provisioning

#### *(menggunakan file bootstrap.sh)*

nano bootstrap.sh

**Contoh**

**Load Balancer**

apt-get update

#instalasi PHP 7.0

apt-get install -y php7.0 php7.0-fpm php7.0-cgi

#instalasi Nginx

apt-get install -y nginx

**Worker 1 & 2**

apt-get update

#instalasi PHP 7.0

apt-get install -y php7.0 php7.0-fpm

#instalasi Web Server Apache

apt-get install -y libapache2-mod-php7.0 apache2

#menjalankan Service Apache

service apache2 start

Menjalankan vagrant Load Balancer dan Kedua Worker

vagrant up --provision

### Masuk ke Dalam Vagrant

### vagrant ssh

### Konfigurasi Vagrant

### Konfigurasi Nginx pada Load Balancer

sudo nano etc/nginx/sites-available/default

Menambahkan upstream, sesuai dengan Algoritma yang dibutuhkan

#Round Robin – default algorithm configuration

upstream worker :

server 192.168.0.3:9000;

server 192.168.0.4:9000;

}

#Least Connection

upstream worker {

least\_conn;

server 192.168.0.3:9000;

server 192.168.0.4:9000;

}

#IP Hash

upstream worker {

ip\_hash;

server 192.168.0.3:9000;

server 192.168.0.4:9000;

}

Konfigurasi Indexing dan Lainnya

# Menambahkan index.php pada list

index index.php index.html index.htm index.nginx-debian.html;

# pass PHP scripts pada FastCGI server listening on "worker"

location ~ \.php$ {

include snippets/fastcgi-php.conf;

fastcgi\_pass worker;

}

Restart Nginx

sudo service nginx restart

**Membuat file PHP pada Load balancer dan Worker**

nano /var/www/html/index.php

**Konfigurasi PHP-fpm pada Worker 1 & 2**

sudo nano /etc/php/7.0/fpm/pool.d/www.conf

Mengubah variabel “**listen”**

dari

listen = /run/php/php7.0-fpm.sock

menjadi

listen = 9000

Restart PHP-fpm

sudo service php7.0-fpm restart

**Konfigurasi DocumentRoot pada Worker 1 & 2**

cd /etc/apache2/sites-available/

sed -i "s/DocumentRoot \/var\/www/DocumentRoot \/var\/www\/html/g" default

**Restart Apache2**

service apache2 restart

Test menggunakan browser

Gunakan Alamat IP dari *load balancer*

### Perbedaan Algoritma IP Hash, Round Robin, dan Least Connection

#### Round Robin

Load Balancing mendistribusikan beban kerja ke Web Server secara merata, sesuai dengan antrian. Antrian tersebut didasarkan pada urutan array IP Address, yang didefinisikan pada Load-Balancer Server. Pendistribusian tersebut dilakukan secara berulang-ulang, sehingga tidak ada beban kerja yang terhambat. Alokasi koneksi secara bergantian. Tidak cocok untuk dynamic Website.

#### Least Connection

Load Balancing mendistribusikan beban kerja ke Web Server, yang mempunyai beban kerja yang paling rendah, yaitu banyaknya **active session** yang sedang dilayani oleh sebuah server. Dynamic Schedule. Tidak cocok untuk dynamic Website.

#### IP Hash

Load Balancing mendistribusikan beban kerja, lewat pemetaan berdasarkan IP Address dari Client. Hash key digenerate dari IP Client dan Server, untuk menempatkan request Client ke Web Server tertentu, sehingga pada pergantian session selanjutnya, request langsung diarahkan ke server yang telah digunakan sebelumnya dan key akan di generate ulang. Algoritma ini berguna untuk sistem yang membutuhkan Client terhubung ke suatu session yang sama, setelah terjadi pergantian session. Parameter subnet mask juga diperhitungkan. Paling cocok untuk dynamic Website.

### Cara mengatasi masalah session, ketika kita melakukan load balancing

Sticky Session. Persistent session method dalam load balancing. Pada hit pertama, Client akan menerima penanda, berbentuk cookie. Pada hit berikutnya, load balancer akan melihat cookie-nya dan mengarahkan ke server yang sebelumnya telah meng-handle Client tersebut. Data sessionnya persisten dan tidak berpencar-pencar.