**COMANDOS MICRO PROYECTO 1**

**Vagrantfile:**

# -\*- mode: ruby -\*-

# vi: set ft=ruby :

Vagrant.configure("2") do |config|

if Vagrant.has\_plugin? "vagrant-vbguest"

config.vbguest.no\_install = true

config.vbguest.auto\_update = false

config.vbguest.no\_remote = true

end

config.vm.define :nodo1 do |nodo1|

nodo1.vm.box = "bento/ubuntu-22.04"

nodo1.vm.network :private\_network, ip: "192.168.100.5"

nodo1.vm.hostname = "nodo1"

nodo1.vm.provision "shell", path: "provision-consul-node1.sh"

# Configuración de Directorio Sincronizado para el Servidor

# nodo1.vm.synced\_folder "C:\\Users\\SkeltonPC\\Documents\\CURSOS 2024 ESP IA\\Vagrant's Files\\Actividad 1\\directorioSincronizadoWindows", "/home/vagrant/directorioSincronizado"

end

config.vm.define :nodo2 do |nodo2|

nodo2.vm.box = "bento/ubuntu-22.04"

nodo2.vm.network :private\_network, ip: "192.168.100.6"

nodo2.vm.hostname = "nodo2"

nodo2.vm.boot\_timeout = 300 # Añadir tiempo de espera

nodo2.vm.provision "shell", path: "provision-consul-node2.sh"

# Configuración de Directorio Sincronizado para el servidor

# nodo2.vm.synced\_folder "C:\\Users\\SkeltonPC\\Documents\\CURSOS 2024 ESP IA\\Vagrant's Files\\Actividad 1\\directorioSincronizadoWindows", "/home/vagrant/directorioSincronizado"

end

config.vm.define :balanceadorCarga do |balanceadorCarga|

balanceadorCarga.vm.box = "bento/ubuntu-22.04"

balanceadorCarga.vm.network :private\_network, ip: "192.168.100.7"

balanceadorCarga.vm.hostname = "balanceadorCarga"

balanceadorCarga.vm.provision "shell", path: "provision-balanceadorCarga.sh"

# Configuración de Directorio Sincronizado para el servidor

# balanceadorCarga.vm.synced\_folder "C:\\Users\\SkeltonPC\\Documents\\CURSOS 2024 ESP IA\\Vagrant's Files\\Actividad 1\\directorioSincronizadoWindows", "/home/vagrant/directorioSincronizado"

end

end

**Crear Archivo script sh para nodo1 (provision-consul-node1.sh) en la ruta raíz donde se encuentra el Vagrantfile:**

# provision-consul.sh

# Este script instala Consul en el nodo y configura su archivo de configuración

# Instalación de Consul

sudo apt-get update -y

sudo apt-get install net-tools -y

sudo apt-get install nano -y

sudo apt-get install vim -y

sudo apt install git -y

# Instalación de Consul

wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg

echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb\_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list

sudo apt update -y

sudo apt install consul -y

sudo consul -v

sudo echo '

# Copyright (c) HashiCorp, Inc.

# SPDX-License-Identifier: BUSL-1.1

# Full configuration options can be found at https://www.consul.io/docs/agent/config

# datacenter

# This flag controls the datacenter in which the agent is running. If not provided,

# it defaults to "dc1". Consul has first-class support for multiple datacenters, but

# it relies on proper configuration. Nodes in the same datacenter should be on a

# single LAN.

#datacenter = "my-dc-1"

# data\_dir

# This flag provides a data directory for the agent to store state. This is required

# for all agents. The directory should be durable across reboots. This is especially

# critical for agents that are running in server mode as they must be able to persist

# cluster state. Additionally, the directory must support the use of filesystem

# locking, meaning some types of mounted folders (e.g. VirtualBox shared folders) may

# not be suitable.

data\_dir = "/opt/consul"

enable\_local\_script\_checks = true

server = false

bind\_addr = "192.168.100.XX"

retry\_join = ["192.168.100.5", "192.168.100.6"]

enable\_syslog = true

log\_level = "INFO"

# client\_addr

# The address to which Consul will bind client interfaces, including the HTTP and DNS

# servers. By default, this is "127.0.0.1", allowing only loopback connections. In

# Consul 1.0 and later this can be set to a space-separated list of addresses to bind

# to, or a go-sockaddr template that can potentially resolve to multiple addresses.

#client\_addr = "0.0.0.0"

# ui

# Enables the built-in web UI server and the required HTTP routes. This eliminates

# the need to maintain the Consul web UI files separately from the binary.

# Version 1.10 deprecated ui=true in favor of ui\_config.enabled=true

#ui\_config{

# enabled = true

#}

# server

# This flag is used to control if an agent is in server or client mode. When provided,

# an agent will act as a Consul server. Each Consul cluster must have at least one

# server and ideally no more than 5 per datacenter. All servers participate in the Raft

# consensus algorithm to ensure that transactions occur in a consistent, linearizable

# manner. Transactions modify cluster state, which is maintained on all server nodes to

# ensure availability in the case of node failure. Server nodes also participate in a

# WAN gossip pool with server nodes in other datacenters. Servers act as gateways to

# other datacenters and forward traffic as appropriate.

#server = true

# Bind addr

# You may use IPv4 or IPv6 but if you have multiple interfaces you must be explicit.

#bind\_addr = "[::]" # Listen on all IPv6

#bind\_addr = "0.0.0.0" # Listen on all IPv4

#

# Advertise addr - if you want to point clients to a different address than bind or LB.' > /etc/consul.d/consul.hcl

sudo echo '{

"service": {

"Name": "web",

"Port": 80,

"check": {

"args": ["curl", "192.168.100.5"],

"interval": "3s"

}

}

}' > /etc/consul.d/web-service.json

# Instalar componentes para microservicios

sudo apt update -y

sudo apt upgrade -y

sudo apt install apache2 -y

sudo systemctl enable apache2

sudo chmod 777 /var/www/html/index.html

sudo echo "Hello from nodo1" > /var/www/html/index.html

# Arrancar agente de Consul

sudo consul agent -ui -server -bootstrap-expect=1 -node=agent-one -bind=192.168.100.5 -data-dir=. -client=0.0.0.0 -config-dir=/etc/consul.d

**Crear Archivo script sh para nodo2 (provision-consul-node2.sh) en la ruta raíz donde se encuentra el Vagrantfile:**

# provision-consul.sh

# Este script instala Consul en el nodo y configura su archivo de configuración

# Instalación de Consul

sudo apt-get update -y

sudo apt-get install net-tools -y

sudo apt-get install nano -y

sudo apt-get install vim -y

sudo apt install git -y

# Instalación de Consul

wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg

echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb\_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list

sudo apt update -y

sudo apt install consul -y

sudo consul -v

sudo echo '# Copyright (c) HashiCorp, Inc.

# SPDX-License-Identifier: BUSL-1.1

# Full configuration options can be found at https://www.consul.io/docs/agent/config

# datacenter

# This flag controls the datacenter in which the agent is running. If not provided,

# it defaults to "dc1". Consul has first-class support for multiple datacenters, but

# it relies on proper configuration. Nodes in the same datacenter should be on a

# single LAN.

#datacenter = "my-dc-1"

# data\_dir

# This flag provides a data directory for the agent to store state. This is required

# for all agents. The directory should be durable across reboots. This is especially

# critical for agents that are running in server mode as they must be able to persist

# cluster state. Additionally, the directory must support the use of filesystem

# locking, meaning some types of mounted folders (e.g. VirtualBox shared folders) may

# not be suitable.

data\_dir = "/opt/consul"

enable\_local\_script\_checks = true

# client\_addr

# The address to which Consul will bind client interfaces, including the HTTP and DNS

# servers. By default, this is "127.0.0.1", allowing only loopback connections. In

# Consul 1.0 and later this can be set to a space-separated list of addresses to bind

# to, or a go-sockaddr template that can potentially resolve to multiple addresses.

#client\_addr = "0.0.0.0"

# ui

# Enables the built-in web UI server and the required HTTP routes. This eliminates

# the need to maintain the Consul web UI files separately from the binary.

# Version 1.10 deprecated ui=true in favor of ui\_config.enabled=true

#ui\_config{

# enabled = true

#}

# server

# This flag is used to control if an agent is in server or client mode. When provided,

# an agent will act as a Consul server. Each Consul cluster must have at least one

# server and ideally no more than 5 per datacenter. All servers participate in the Raft

# consensus algorithm to ensure that transactions occur in a consistent, linearizable

# manner. Transactions modify cluster state, which is maintained on all server nodes to

# ensure availability in the case of node failure. Server nodes also participate in a

# WAN gossip pool with server nodes in other datacenters. Servers act as gateways to

# other datacenters and forward traffic as appropriate.

#server = true

# Bind addr

# You may use IPv4 or IPv6 but if you have multiple interfaces you must be explicit.

#bind\_addr = "[::]" # Listen on all IPv6

#bind\_addr = "0.0.0.0" # Listen on all IPv4

#

# Advertise addr - if you want to point clients to a different address than bind or LB' > /etc/consul.d/consul.hcl

sudo echo '{

"service": {

"Name": "web",

"Port": 80,

"check": {

"args": ["curl", "192.168.100.6"],

"interval": "3s"

}

}

}' > /etc/consul.d/web-service.json

# Instalar componentes para microservicios

sudo apt update -y

sudo apt upgrade -y

sudo apt install apache2 -y

sudo systemctl enable apache2

sudo chmod 777 /var/www/html/index.html

sudo echo "Hello from nodo2" > /var/www/html/index.html

**Crear Archivo script sh para balanceador de carga (provision-balanceadorCarga.sh) en la ruta raíz donde se encuentra el Vagrantfile:**

# Este script instala HAPROXY y lo configura

# Instalación de Consul

sudo apt-get update -y

sudo apt-get install net-tools -y

sudo apt-get install nano -y

sudo apt-get install vim -y

sudo apt install git -y

#Instalacion de HAProxy

sudo apt update -y

sudo apt upgrade -y

sudo apt install haproxy -y

sudo systemctl enable haproxy

sudo systemctl start haproxy

# Definir el archivo a modificar

echo "Modificando haproxy.cfg"

sudo echo 'global

log /dev/log local0

log /dev/log local1 notice

chroot /var/lib/haproxy

stats socket /run/haproxy/admin.sock mode 660 level admin expose-fd listeners

stats timeout 30s

user haproxy

group haproxy

daemon

# Default SSL material locations

ca-base /etc/ssl/certs

crt-base /etc/ssl/private

# Default ciphers to use on SSL-enabled listening sockets.

# For more information, see ciphers(1SSL). This list is from:

# https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/

# An alternative list with additional directives can be obtained from

# https://mozilla.github.io/server-side-tls/ssl-config-generator/?server=haproxy

ssl-default-bind-ciphers ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:RSA+AESGCM:RSA+AES:!aNULL:!MD5:!DSS

ssl-default-bind-options no-sslv3

defaults

log global

mode http

option httplog

option dontlognull

timeout connect 5000

timeout client 50000

timeout server 50000

errorfile 400 /etc/haproxy/errors/400.http

errorfile 403 /etc/haproxy/errors/403.http

errorfile 408 /etc/haproxy/errors/408.http

errorfile 500 /etc/haproxy/errors/500.http

errorfile 502 /etc/haproxy/errors/502.http

errorfile 503 /etc/haproxy/errors/503.http

errorfile 504 /etc/haproxy/errors/504.http

backend web-backend

balance roundrobin

server-template consul\_node 2 \_web.\_tcp.service.consul resolvers consul check

stats enable

stats auth admin:admin

stats uri /haproxy?stats

frontend http

bind \*:80

default\_backend web-backend

resolvers consul

nameserver consul 192.168.100.5:8600

nameserver consul2 192.168.100.6:8600

accepted\_payload\_size 8192

hold valid 5s' > /etc/haproxy/haproxy.cfg

echo 'HTTP/1.0 503 Service Unavailable

Cache-Control: no-cache

Connection: close

Content-Type: text/html

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Error 503 - Servicio No Disponible</title>

<style>

body {

font-family: 'Arial', sans-serif;

background-color: #f8f8f8;

color: #333;

margin: 0;

padding: 0;

display: flex;

align-items: center;

justify-content: center;

height: 100vh;

}

.error-container {

text-align: center;

}

h1 {

font-size: 2em;

color: #dc3545;

margin-bottom: 10px;

}

p {

font-size: 1.2em;

color: #555;

margin-top: 0;

}

</style>

</head>

<body>

<div class="error-container">

<h1>Lo sentimos, estamos experimentando problemas</h1>

<p>En este momento, no podemos atender tu solicitud. Por favor, inténtalo de nuevo más tarde.</p>

</div>

</body>

</html>

' > /etc/haproxy/errors/503.http

echo "Archivo de configuración actualizado con éxito."

sudo systemctl restart haproxy

sudo systemctl status haproxy

**NOTA: LOS ARCHIVOS DE APROVISIONAMIENTO LEVANTAN TODO AUTOMATIZADO ENTONCES ES NECESARIO QUE SE DEBA DE SEGUIR TODOS LOS PASOS SIGUIENTES, SOLAMENTE ES PARA ENTENDER PASO A PASO PARA TODO, PERO UNA VEZ FINALICE EL APROVISIONAMIENTO SE PUEDE ACCEDER A:**  
<http://192.168.100.5:8500/ui/dc1/overview/server-status>  
<http://192.168.100.6:8500/ui/dc1/overview/server-status>

<http://192.168.100.7/haproxy?stats>

**para haproxy stats**

**usuario: admin**

**password: admin**

* sudo systemctl stop apache2
* sudo systemctl start apache2
* sudo systemctl restart apache2
* consul leave
* consul members

**TENER EN CUENTA QUE YA TENIENDO LAS MAQUINAS ARRIBA PUEDO ABRIR VENTANAS CMD, ACCEDER POR SSH Y SE PUEDE UTILIZAR LOS COMANDOS QUE REQUIERA**

**COMANDOS Maquina NODO 1**

* **vagrant ssh nodo1**
* **sudo apt update -y**
* **sudo apt upgrade -y**
* **sudo apt install apache2 -y**
* **sudo systemctl enable apache2**
* **sudo chmod 777 /var/www/html/index.html**
* **sudo echo "Hello from nodo1" > /var/www/html/index.html**

sudo consul agent -ui -server -bootstrap-expect=1 -node=agent-one -bind=192.168.100.5 -data-dir=. -client=0.0.0.0 -config-dir=/etc/consul.d

**ABRIMOS OTRA VENTANA CMD Y EJECUTAMOS LOS COMANDOS:**

* **vagrant ssh nodo1**
* consul members
* Se puede acceder desde el navegador: <http://192.168.100.5:8500/ui/dc1/overview/server-status>
* cd /etc/consul.d
* **sudo nano web-service.json** y colocamos la siguiente config:  
  {

"service": {

"Name": "web",

"Port": 80,

"check": {

"args": ["curl", "192.168.100.5"],

"interval": "3s"

}

}

}

* **sudo nano consul.hcl** y colocamos la siguiente config:

# Copyright (c) HashiCorp, Inc.

# SPDX-License-Identifier: BUSL-1.1

# Full configuration options can be found at https://www.consul.io/docs/agent/config

# datacenter

# This flag controls the datacenter in which the agent is running. If not provided,

# it defaults to "dc1". Consul has first-class support for multiple datacenters, but

# it relies on proper configuration. Nodes in the same datacenter should be on a

# single LAN.

#datacenter = "my-dc-1"

# data\_dir

# This flag provides a data directory for the agent to store state. This is required

# for all agents. The directory should be durable across reboots. This is especially

# critical for agents that are running in server mode as they must be able to persist

# cluster state. Additionally, the directory must support the use of filesystem

# locking, meaning some types of mounted folders (e.g. VirtualBox shared folders) may

# not be suitable.

data\_dir = "/opt/consul"

enable\_local\_script\_checks = true

server = false

bind\_addr = "192.168.100.XX"

retry\_join = ["192.168.100.5"]

enable\_syslog = true

log\_level = "INFO"

# client\_addr

# The address to which Consul will bind client interfaces, including the HTTP and DNS

# servers. By default, this is "127.0.0.1", allowing only loopback connections. In

# Consul 1.0 and later this can be set to a space-separated list of addresses to bind

# to, or a go-sockaddr template that can potentially resolve to multiple addresses.

#client\_addr = "0.0.0.0"

# ui

# Enables the built-in web UI server and the required HTTP routes. This eliminates

# the need to maintain the Consul web UI files separately from the binary.

# Version 1.10 deprecated ui=true in favor of ui\_config.enabled=true

#ui\_config{

# enabled = true

#}

# server

# This flag is used to control if an agent is in server or client mode. When provided,

# an agent will act as a Consul server. Each Consul cluster must have at least one

# server and ideally no more than 5 per datacenter. All servers participate in the Raft

# consensus algorithm to ensure that transactions occur in a consistent, linearizable

# manner. Transactions modify cluster state, which is maintained on all server nodes to

# ensure availability in the case of node failure. Server nodes also participate in a

# WAN gossip pool with server nodes in other datacenters. Servers act as gateways to

# other datacenters and forward traffic as appropriate.

#server = true

# Bind addr

# You may use IPv4 or IPv6 but if you have multiple interfaces you must be explicit.

#bind\_addr = "[::]" # Listen on all IPv6

#bind\_addr = "0.0.0.0" # Listen on all IPv4

#

# Advertise addr - if you want to point clients to a different address than bind or LB.

**COMANDOS Maquina NODO 2**

* **vagrant ssh nodo2**
* **sudo apt update -y**
* **sudo apt upgrade -y**
* **sudo apt install apache2 -y**
* **sudo systemctl enable apache2**
* **sudo chmod 777 /var/www/html/index.html**
* **sudo echo "Hello from nodo2" > /var/www/html/index.html**
* sudo consul agent -ui -node=agent-two -bind=192.168.100.6 -enable-script-checks=true -client=0.0.0.0 -data-dir=. -config-dir=/etc/consul.d
* consul join 192.168.100.5
* desde el navegador podemos acceder: http://192.168.100.6:8500/ui/dc1/nodes/agent-one/health-checks

**ABRIMOS OTRA VENTANA CMD Y EJECUTAMOS LOS COMANDOS:**

* **vagrant ssh nodo2**
* consul members
* cd /etc/consul.d
* **sudo nano web-service.json** y colocamos la siguiente config:  
  {

"service": {

"Name": "web",

"Port": 80,

"check": {

"args": ["curl", "192.168.100.6"],

"interval": "3s"

}

}

}

* **sudo nano consul.hcl** y colocamos la siguiente config:

# Copyright (c) HashiCorp, Inc.

# SPDX-License-Identifier: BUSL-1.1

# Full configuration options can be found at https://www.consul.io/docs/agent/config

# datacenter

# This flag controls the datacenter in which the agent is running. If not provided,

# it defaults to "dc1". Consul has first-class support for multiple datacenters, but

# it relies on proper configuration. Nodes in the same datacenter should be on a

# single LAN.

#datacenter = "my-dc-1"

# data\_dir

# This flag provides a data directory for the agent to store state. This is required

# for all agents. The directory should be durable across reboots. This is especially

# critical for agents that are running in server mode as they must be able to persist

# cluster state. Additionally, the directory must support the use of filesystem

# locking, meaning some types of mounted folders (e.g. VirtualBox shared folders) may

# not be suitable.

data\_dir = "/opt/consul"

enable\_local\_script\_checks = true

# client\_addr

# The address to which Consul will bind client interfaces, including the HTTP and DNS

# servers. By default, this is "127.0.0.1", allowing only loopback connections. In

# Consul 1.0 and later this can be set to a space-separated list of addresses to bind

# to, or a go-sockaddr template that can potentially resolve to multiple addresses.

#client\_addr = "0.0.0.0"

# ui

# Enables the built-in web UI server and the required HTTP routes. This eliminates

# the need to maintain the Consul web UI files separately from the binary.

# Version 1.10 deprecated ui=true in favor of ui\_config.enabled=true

#ui\_config{

# enabled = true

#}

# server

# This flag is used to control if an agent is in server or client mode. When provided,

# an agent will act as a Consul server. Each Consul cluster must have at least one

# server and ideally no more than 5 per datacenter. All servers participate in the Raft

# consensus algorithm to ensure that transactions occur in a consistent, linearizable

# manner. Transactions modify cluster state, which is maintained on all server nodes to

# ensure availability in the case of node failure. Server nodes also participate in a

# WAN gossip pool with server nodes in other datacenters. Servers act as gateways to

# other datacenters and forward traffic as appropriate.

#server = true

# Bind addr

# You may use IPv4 or IPv6 but if you have multiple interfaces you must be explicit.

#bind\_addr = "[::]" # Listen on all IPv6

#bind\_addr = "0.0.0.0" # Listen on all IPv4

#

# Advertise addr - if you want to point clients to a different address than bind or LB.

**COMANDOS BALANCEADOR DE CARGA**

**Accedemos mediante ssh a la máquina de balanceo de carga:**

* **vagrant ssh balanceadorCarga**
* sudo apt-get install lxd-installer -y
* newgrp lxd
* lxd init --auto
* lxc launch ubuntu:18.04 haproxy
* lxc exec haproxy /bin/bash
* apt update && apt upgrade
* apt install haproxy
* systemctl enable haproxy

**Vamos a editar a el archivo haproxy.cfg:**

* nano /etc/haproxy/haproxy.cfg

**añadimos toda esta configuración:**

global

log /dev/log local0

log /dev/log local1 notice

chroot /var/lib/haproxy

stats socket /run/haproxy/admin.sock mode 660 level admin expose-fd listeners

stats timeout 30s

user haproxy

group haproxy

daemon

# Default SSL material locations

ca-base /etc/ssl/certs

crt-base /etc/ssl/private

# Default ciphers to use on SSL-enabled listening sockets.

# For more information, see ciphers(1SSL). This list is from:

# https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/

# An alternative list with additional directives can be obtained from

# https://mozilla.github.io/server-side-tls/ssl-config-generator/?server=haproxy

ssl-default-bind-ciphers ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:RSA+AESGCM:RSA+AES:!aNULL:!MD5:!DSS

ssl-default-bind-options no-sslv3

defaults

log global

mode http

option httplog

option dontlognull

timeout connect 5000

timeout client 50000

timeout server 50000

errorfile 400 /etc/haproxy/errors/400.http

errorfile 403 /etc/haproxy/errors/403.http

errorfile 408 /etc/haproxy/errors/408.http

errorfile 500 /etc/haproxy/errors/500.http

errorfile 502 /etc/haproxy/errors/502.http

errorfile 503 /etc/haproxy/errors/503.http

errorfile 504 /etc/haproxy/errors/504.http

backend web-backend

balance roundrobin

server-template consul\_node 2 \_web.\_tcp.service.consul resolvers consul check

stats enable

stats auth admin:admin

stats uri /haproxy?stats

frontend http

bind \*:80

default\_backend web-backend

resolvers consul

nameserver consul 192.168.100.5:8600

accepted\_payload\_size 8192

hold valid 5s

* systemctl restart haproxy
* systemctl status haproxy.service
* lxc config device add haproxy http proxy listen=tcp:0.0.0.0:80 connect=tcp:127.0.0.1:80

El estado debería ser 200 y verde.

* Abrir url en navegador <http://192.168.100.7/haproxy?stats>

**login: admin**

**password: admin**

* Y si abris solito el http://192.168.100.7 y se le da varias veces enter se va viendo las distintas peticiones a los distintos servidores web
* Se puede realizar **curl 192.168.100.7:80**

**COMANDOS PARA ARTILLERY**

* Primero instalar node js desde la pagina oficial en el host <https://nodejs.org/en/download/current>

**ABRIMOS UN CMD EN MODO ADMINISTRADOR PARA EJECUTAR LOS COMANDOS:**

* npm install -g artillery

**Verificamos la instalación**

* npx artillery dino
* artillery versión

**Creamos el archivo “pruebas-de-carga.yml” con el contenido:**

config:

target: 'http://192.168.100.7'

phases:

- duration: 60

arrivalRate: 1

scenarios:

- flow:

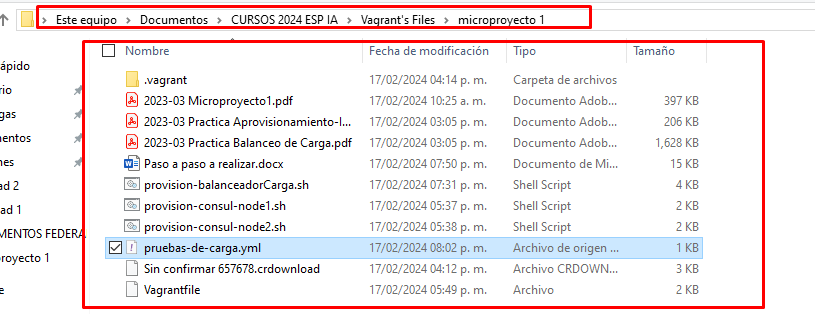
- get:

url: 'http://192.168.100.7'

- get:

url: 'http://192.168.100.7'

**En la ruta raíz donde tenemos nuestros archivos creamos nuestro archivo “pruebas-de-carga.yml”**



**Desde el cmd puesto en la ruta raíz donde esta nuestro “prueba-de-carga.yml” ejecutamos:**

* artillery run "pruebas-de-carga.yml"

**COMANDOS PARA TUMBAR APACHE EN CUALQUIERA DE LOS DOS NODOS Y VISUALIZAR LAS STATS:**

* sudo systemctl stop apache2
* sudo systemctl start apache2
* sudo systemctl restart apache2