Universitat de les Illes Balears

Escola Politècnica Superior

Memòria del Treball de Fi de Grau

Proyecto de automatización de una infraestructura web.

Rafael Bosch Lladó

Grau d'Enginyeria Informàtica.

Any acadèmic 2019-20

DNI de l'alumne: 43198591Z

Treball tutelat per Isaac Lera Castro
Departament de Matemàtiques i informàtica

S'autoritza la Universitat a incloure aquest treball en el Repositori Institucional per a la seva consulta en accés obert i difusió en línia, amb finalitats exclusivament acadèmiques i d'investigació	Autor		Tutor	
	Sí	No	Sí	No
	X		X	

Paraules clau del treball:
Automatización, Virtualización, Balanceador, Servidores, Ansible, VMWARE

Agradecimientos

Me gustaría mostrar mi gratitud a mi tutor el Dr. Isaac Lera Castro por su implicación durante todo el transcurso de mi proyecto y por su impecable trato tanto personal como profesional.

A Carlos Triviño Manjón por ser mi mentor en la empresa donde se realizó este proyecto. Siempre tuvo paciencia contestándome las dudas que me surgieron y trabajar junto a él fue un gran placer.

A mi familia por su ayuda siempre desde el inicio de la carrera y por no haber dejado que nunca baje los brazos.

Y por último, quisiera hacer mención especial a Laura Casanueva Reimon. Me ha ayudado en todo momento, me ha escuchado y ha entendido en lo que estaba trabajando, incluso sorprendiéndome hablando de forma técnica en algunas de las conversaciones. Pero, por encima de todo, por indicarme la mejor manera de expresarme para que el contenido se entendiese.

Resumen

El proyecto que tenía como objetivo final montar un sistema de balanceo de forma automática. Se ha usado un modelo cliente-servidor con balanceador para distribuir la carga entre los servidores web. Las herramientas utilizadas han sido herramientas de automatización, virtualización y monitorización. Siendo la herramienta Ansible, de automatización, la que ha tenido más peso dentro del proyecto. El uso de las tres herramientas ha sido indispensable para poder lograr el objetivo del proyecto. Los servidores usados han sido servidores de aplicaciones y servidores webs. Ambos, han sido máquinas virtuales con el sistema operativo CentOS. El balanceador que se instaló entre los servidores de aplicaciones y los servidores web fue un balanceador por software. Finalmente añadir que el sistema puede ser escalado, con la ayuda de la persona responsable, para adaptarse a las necesidades de carga sobre el sistema.

Tabla de contenido

1. Introducción

Hoy en día, la evolución tecnológica gira en torno al paradigma de internet ya que gran parte de los dispositivos existentes, ya sean teléfonos móviles, tabletas o raspberries, lo usan para ofrecer innumerables servicios. Estos servicios suelen ser computados en servidores y a causa de su demanda, están sometidos a mayores cargas de trabajo. Una manera de poder atender este exceso de carga de trabajo entre múltiples servidores es usando balanceadores de carga. Gracias a estos balanceadores se evita que los servidores se vean sometidos a cargas que puedan llegar a saturarlos. Por este motivo, la arquitectura *cliente-servidor básica* ha evolucionado hasta convertirse en una *arquitectura cliente-servidor con balanceador* como un elemento extra.

En la Figura 1(A) observamos una representación esquemática como ejemplo de una posible implementación que presenta una arquitectura *cliente-servidor* básica. En esta implementación, existe un dispositivo que puede acceder a internet y toda la comunicación recae sobre un solo servidor, su caída podría provocar una caída del sistema. En cambio, en la Figura 1(B) observamos una representación esquemática de cómo es una implementación cliente-servidor con un balanceador y varios servidores. En este caso, se consigue repartir las peticiones de los dispositivos sobre los servidores y, por ende, reducir el riesgo de caída de sistema.

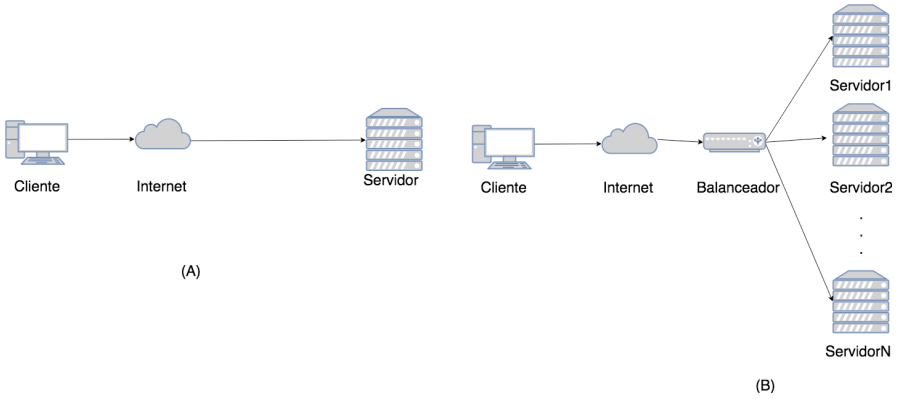

Figura 1: Modalidad de arquitectura cliente-servidor A) Modelo de Arquitectura cliente-servidor básica. B) Modelo de arquitectura cliente-servidor con balanceador

El hándicap proviene del hecho de desconocer el comportamiento del tráfico de llegadas. Por lo general, este tráfico web suele recibir el nombre de ráfagas (burstiness) [1]. Para adaptar toda la arquitectura del sistema al tráfico variable de llegadas es necesario crearla y configurarla dinámicamente. Esto se puede conseguir utilizando mecanismos de automatización [2], obteniendo así un sistema adaptable a la carga bajo las indicaciones de la persona responsable.

El objetivo del proyecto consiste en implementar un sistema de balanceo automatizado; en el cual se tienen que usar herramientas de automatización y virtualización [3]. El sistema estará formado por uno o varios servidores webs encargados de responder las peticiones, un balanceador de software que se encargará de distribuir la carga y uno o varios servidores de aplicaciones que van a contener el servicio al cual se quiere acceder.

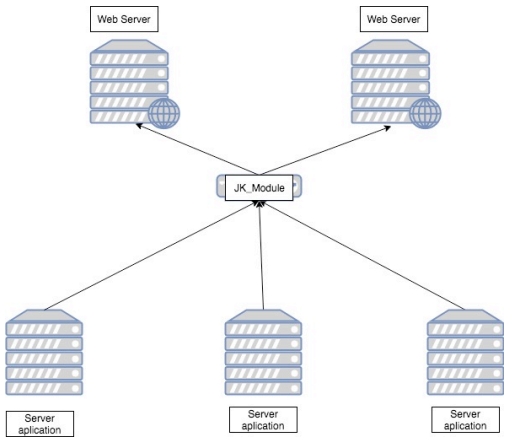

Figura 2: Objetivo del proyecto: Arquitectura cliente servidor con balanceador

En la Figura 2 podemos observar el diseño de nuestro proyecto esquematizado. Los servidores de aplicaciones y los servidores web se pueden escalar en función de los requisitos del proyecto. Como muestra el esquema, el balanceador está en conexión con los dos tipos de servidores para así poder realizar su función de balancear.

Este proyecto se ha realizado para una empresa del sector turístico donde el objetivo final era automatizar el despliegue de un sistema para asegurar la disponibilidad de una página web. Las herramientas han sido seleccionadas por la empresa en base a su

trayectoria y su proyección del negocio. En este proyecto se han utilizado herramientas automatización, virtualización, servidores y un módulo de software que funcionaría como balanceador. Concretamente las herramientas escogidas han sido Ansible, como herramienta de Automatización, la herramienta de virtualización VMWARE, servidores web Apache, servidores de aplicaciones Tomcat y como balanceador un módulo por software llamado JK_Module. El uso de estas herramientas permite conseguir un sistema automatizado en el cual se logra montar el proyecto entero desde cero pero en el que también se puedan agregar o eliminar elementos desde un sistema ya en funcionamiento. El uso de herramientas de virtualización ha permitido tener los diferentes servidores web y los servidores de aplicaciones en el hardware.

Los diferentes tipos de balanceadores existentes hoy en día son balanceadores basados en DNS, balanceadores de software y balanceadores de carga dedicado. Cada uno de ellos balancea la carga de forma diferente [8].

1.1 Tareas del proyecto

El proyecto consiste en una serie de tareas:

1. Obtener los requerimientos del proyecto: Con la empresa cliente, que en este caso pertenece al sector turístico, se estableció que querían como objetivo final y como tenía que funcionar para ellos.
2. Diseñar la arquitectura: Al tener como objetivo final un sistema automatizado con uso de balanceador se optó por usar un modelo *cliente-servidor* con balanceador como hemos observado anteriormente en la Figura 1(B).
3. Diseñar el sistema: El sistema consta de *n* servidores para tratar las peticiones de los clientes, *x* servidores de aplicaciones y un balanceador por software. En la Figura 3 se muestra como fue el diseño del sistema. La mayor parte de las

pruebas se han realizado sobre 2 servidores de peticiones y 3 o 4 servidores de aplicaciones.

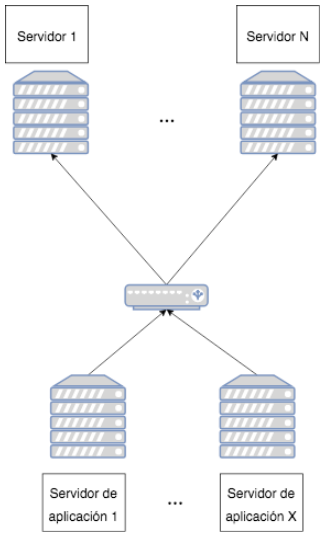

Figura 3: Diseño de la estructura del sistema utilizado en nuestro proyecto

4. Conocer y aprender las herramientas: Se realizó una búsqueda en profundidad de como funcionaban las principales herramientas Ansible, Consul, VMWARE, Apache y Tomcat. Seguido de un estudio sobre cómo se podían implementar estas herramientas en nuestro proyecto y del aprendizaje de sus respectivos lenguajes de programación.
5. Implementación del sistema: La implementación del proyecto se realizó usando la herramienta de automatización Ansible, VMWARE como herramienta de virtualización, Apache como servidor web, Tomcat como servidor de aplicación y el módulo JK_Module como balanceador por software.
6. Realización de pruebas: Las pruebas realizadas consistieron en verificar que las funciones que se implementaban funcionasen correctamente. Por ejemplo, la creación de nuevas máquinas mediante Ansible y que se incorporasen en el entorno virtualizado. La IP asignada a cada máquina tenía que ser el identificador que se usaba en la herramienta Consul para tener las máquinas

controladas y la comprobación de que los servidores Apache y Tomcat se instalaran correctamente.

En la Figura 4 se puede observar el diagrama de Gantt que muestra el tiempo estimado para cada una de las tareas del proyecto. La tarea que más tiempo ha conllevado ha sido la tarea 5: Implementación del sistema.

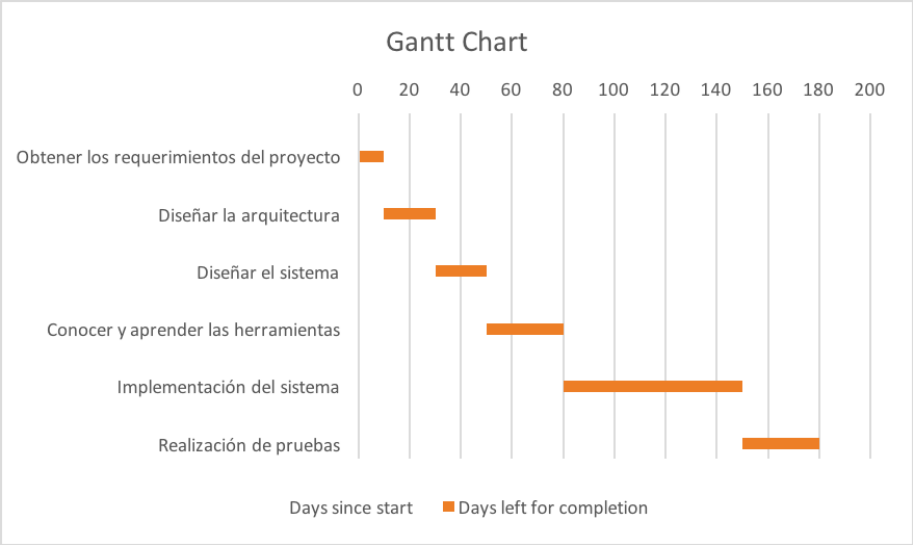

Figura 4: Gantt Chart para el desarrollo de las tareas

2. Background Tecnológico

Este capítulo se estructura de la siguiente forma. Primero, definiremos el concepto de balanceador de carga, los tipos y como gestionan la carga de trabajo. Esto nos permitirá entender la necesidad de esta tecnología para gestionar las ráfagas de tráfico y evitar que el sistema se colapse.

En el segundo punto, definiremos el concepto de virtualización y el papel que desempeña en la gestión de servicios. Comentaremos algunas de las herramientas más usuales para la virtualización pero haremos hincapié en la herramienta utilizada en la empresa, VMWARE. Cuando disponemos de múltiples máquinas, ya sean físicas o virtuales, si queremos conocer su estado de funcionamiento para tareas de automatización será necesario utilizar herramientas que nos proporcionen dicha información. Así, en el tercer punto trataremos el control de dispositivos.

Finalmente, el nexo común de todas estas tecnologías y en las que se basa este trabajo es el proceso de automatización. En la última sección, se expondrán los detalles más generales que tienen las herramientas de automatización, como por ejemplo su organización o su manejo. Además, se mencionarán las herramientas más comunes y sus principales ventajas y desventajas.

Durante muchos años la arquitectura *cliente-servidor* ha sido la más utilizada. Esta arquitectura *cliente-servidor* consiste en un servidor que responde a las peticiones realizadas por los clientes. Debido al exceso de carga sobre los servidores, la arquitectura *cliente-servidor* ha evolucionado hasta complementarse con el uso de un balanceador entre los clientes y los servidores. Este balanceador permite repartir el nivel de carga entre los diferentes servidores. La carga de trabajo se repartirá según el modelo de distribución de carga que use el balanceador. Existen balanceadores basados en DNS, balanceadores de software y balanceadores de carga dedicado. Cada uno de ellos balancea la carga de forma diferente [4]. Además de por el tipo de balanceadores que use el sistema, también podemos clasificarlos según los métodos

de balanceo de carga empleados, entre los cuales destacan Round-Robin, Weighted round-robin y LeastConnection [5, 6, 7].

Para aprovechar al máximo el hardware, se virtualizó uno de los servidores de la empresa. La virtualización permite crear múltiples entornos simulados o recursos dedicados desde un solo sistema de hardware físico. En este proyecto, utilizamos un software conocido como "hipervisor". El software hipervisor [8] se conecta con el hardware y permite dividir un sistema en diferentes entornos separados, distintos y seguros, conocidos como máquinas virtuales.

El uso de sistemas de virtualización ha ido incrementando con el paso del tiempo. Otro concepto que a día de hoy se utiliza con frecuencia es el de automatización. La automatización consiste en procesos y tareas que actúan de forma automática para así reducir el tiempo de ejecución y los costes, además de mejorar el mantenimiento y la flexibilidad de un sistema. Las tareas que se automatizan suelen ser tareas repetitivas como por ejemplo, realizar la instalación de un software en diferentes servidores. Por este motivo, si lo que se busca es automatizar tareas y monitorizar estados finales en servidores se usarán las herramientas de automatización.

Existen diferentes herramientas de automatización como pueden ser Chef [9], Puppet [10] o Ansible [11], entre otras. Estas herramientas son similares entre ellas, sobretodo Puppet y Chef, las cuales utilizan el lenguaje de programación *ruby on rails*[1] y una arquitectura maestro-esclavo [12] para el desarrollo de las tareas. Chef tiene incorporado *git* [13] lo que favorece que sea más robusto en el control de versiones. En cambio, Ansible presenta una arquitectura de tipo *controlador-nodo*. Esta última

[1] Es un framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby.

arquitectura permite que desde el controlador (*host* principal) se indique a los nodos destino los procesos que deben ejecutar. El lenguaje que usa Ansible es YAML [14] (un formato de serialización fácil de interpretar) y se utiliza para desarrollar los roles [15] y los playbooks [16]. Los roles son los archivos que contienen las tareas que se ejecutarán en las máquinas nodos y realizarán la instalación del código programado. Los playbooks son los archivos que determinan a que hosts y que roles se van a ejecutar en la máquinas nodos.

La tecnología de virtualización y la herramienta Ansible han sido imprescindibles en la realización de nuestro proyecto. Éstas junto con el balanceador nos han permitido desarrollar el proyecto que expongo en este informe.

2.1 Balanceadores

Como hemos dicho anteriormente, el balanceador es un dispositivo que permite distribuir la carga de un sistema para que los servidores no se sobresaturen. Existen diferentes tipos de balanceadores, entre los cuales podemos nombrar:

- Balanceadores basados en DNS: para balancear la carga usan identificadores que clasificarán a donde se va dirigir la carga. Estos identificadores son las direcciones IP's de las máquinas.
- Balanceadores de software: se usa una aplicación para realizar el balanceo de la carga. No se necesitan usar un dispositivo físico como sí usan los balanceadores de carga dedicados por hardware.
- Balanceadores de carga dedicado: usan un dispositivo físico que permite repartir la carga hacia los servidores. Este dispositivo físico puede contener una aplicación de balanceo tanto de código libre, es decir, un código por el que no se tiene que pagar para usarlo, como de un código de pago.

Para no sobrecargar los servidores, se necesitan algoritmos que dividan, repartan y envíen la carga a distintos servidores. Según el algoritmo que se use se realizará la

distribución de carga de manera diferente. Las técnicas más usadas para gestionar las cargas son:

- Round-Robin: consiste en asignar la primera petición al primer elemento, en nuestro caso el primer elemento es el servidor; mientras que la segunda petición llegará al segundo servidor y así sucesivamente hasta llegar al último servidor. En este momento volverá a empezar en el primer servidor. Con este método se consigue una aplicación circular tratando todas las peticiones recibidas de la misma manera.
- Weighted round-robin: consiste en dotar de pesos a los servidores según su capacidad de procesamiento por lo que los servidores con mayor capacidad obtendrían un peso mayor. De esta manera, al repartir la carga, los servidores con mayor peso reciben las primeras peticiones y, también, más volumen de carga que los otros servidores con menor peso.
- LeastConnection: se ocupa principalmente de contar el número de conexiones que tiene cada servidor y así enviar las peticiones al servidor que tiene menos conexiones en ese momento.

Por lo tanto, en la elección de un balanceador se deben considerar varios factores como el tipo de balanceador, el método que se va a utilizar y el coste de cada uno de ellos.

2.2 Virtualización de servidores con VMWARE

Otra tecnología utilizada en nuestro proyecto ha sido la virtualización. La virtualización es el proceso que permite crear una representación basada en software tanto en servidores, redes, almacenamiento como en aplicaciones. Existen varios tipos de virtualización [17]:

1. Virtualización de servidores
2. Virtualización de datos
3. Virtualización de escritorios
4. Virtualización de redes

Cada tipo de virtualización tiene funciones diferentes. La virtualización por servidores se hace mediante una aplicación de control conocida como Hipervisor de manera que cada máquina virtual puede funcionar independiente como si fuese un hardware físico diferente. Además, cada una de las máquinas puede llevar sistemas operativos distintos y correr aplicaciones diferentes. En nuestro caso se ha usado la virtualización por servidores mediante la herramienta VMWARE vSphere [18].

VMWARE es un software especializado en virtualización. Ofrece una serie de productos para diferentes tipos de virtualización. En este proyecto se usan los específicos para virtualización de servidores. Las principales ventajas de VMWARE son: reducción de costes en hardware y mejora de la eficacia, flexibilidad, mantenimiento y rendimiento de trabajo. El entorno de virtualización en nuestro proyecto se hizo con la instalación de un Hipervisor tipo 1 llamado ESXi. Los Hipervisores de tipo 1 se ejecutan directamente sobre el hardware y no sobre un sistema operativo ya instalado como son los de tipo 2.

En cuanto a las desventajas que presenta esta tecnología de VMWARE podemos mencionar principalmente el coste que tiene usar sus servicios. Crear y mantener una infraestructura virtualizada no es sencillo y se necesita de un buen poder económico para poder hacer frente a este gasto. También se puede añadir como desventaja la dificultad de aprender el funcionamiento de las aplicaciones.

En la Figura 5 vemos un ejemplo de una arquitectura de virtualización con un Hipervisor VMWARE tipo 1. El ESXi es la capa inmediatamente superior a la capa física y nos permite controlar el servidor de forma remota. Se accede a esta capa a través de un navegador web(IP proporcionada al finalizar la instalación) o desde otro producto de VMWARE, llamado VSPHERE, que facilita la gestión de máquinas virtuales. En la

capa ESXi es donde se pueden instalar y configurar los diferentes sistemas informáticos. Las características de las máquinas podrán ser configuradas y modificadas durante su vida útil. Las principales características modificables son el tipo de red, almacenamiento, memoria RAM y procesador, entre otras. La cantidad de máquinas virtuales que se pueden crear viene determinada por las especificaciones físicas del servidor.

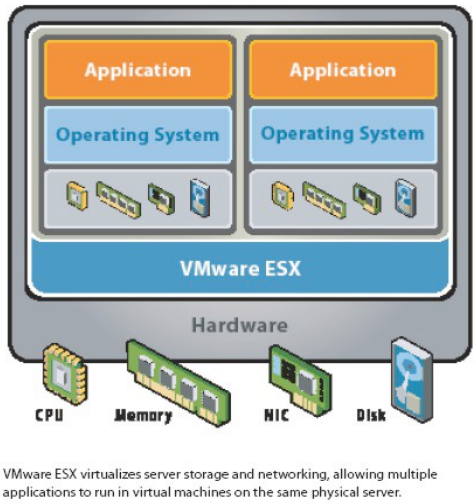

Figura 5: Arquitectura de dispositivo virtualizado con ESXi

2.3 Control de los dispositivos con Consul

Existen diferentes tecnologías que permiten monitorizar el estado de las máquinas como pueden ser: Nagios, Sensu y Consul. Esta última, no solo sirve para monitorizar el estado de las máquinas sino que también puede ser utilizada como descubrimiento de servicio. De hecho, Consul ha sido la tecnología elegida para controlar el estado de las máquinas en nuestro proyecto.

Consul es un servicio que permite organizar la infraestructura de un sistema de forma sencilla. Entre sus diversas funciones destacamos la monitorización del estado de las máquinas. Con el reconocimiento del estatus se puede comprobar si las máquinas están funcionando correctamente o si hay algún error. En el caso de detectar algún

error en el servicio se indicaría con un color rojizo en el estatus de la máquina supervisada por Consul.

Al igual que en una base de datos, las máquinas se pueden añadir y eliminar del sistema; por lo que, si se crea una máquina nueva, automáticamente se da de alta como un servicio en Consul. Consul la reconoce por su dirección IP y se le aplica la configuración previamente establecida.

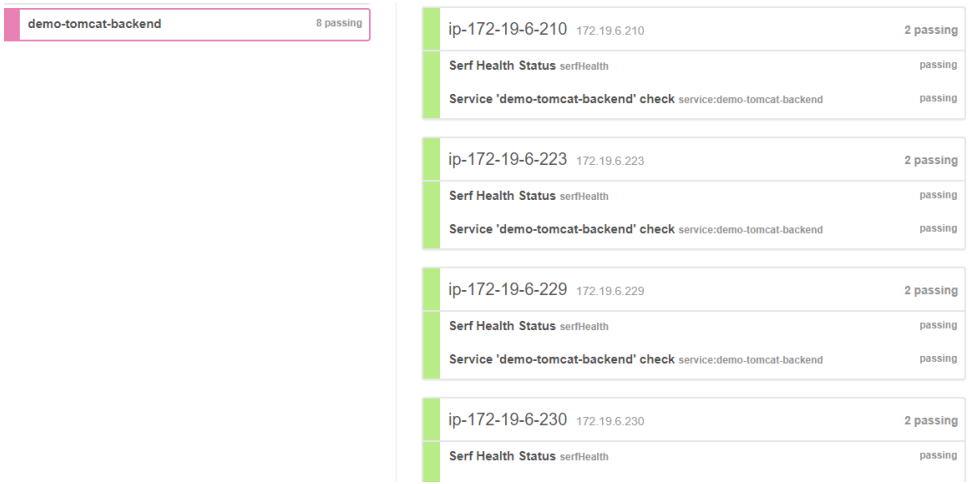

Figura 6: Estatus de las máquinas monitorizadas por la herramienta Consul

La Figura 6 muestra el panel principal de Consul donde nos indica el estado de las máquinas. En este caso vemos que todas las máquinas están en funcionamiento. Además, el panel principal de Consul también nos comunica el tipo de servicio de cada máquina.

Las ventaja de utilizar Consul es la facilidad de comprobar el estado de las máquinas y el ahorro de tiempo que se consigue al no tener que comprobar cada máquina de manera individual. Todo esto facilita la mejora de la productividad dentro de un entorno de trabajo. Las principales desventajas de Consul es la instalación de la aplicación Consul y su configuración.

2.4 Automatización

La automatización es otra de las tecnologías utilizadas para realizar nuestro proyecto. La automatización es el uso de la tecnología para realizar tareas sin la necesidad de un ser humano. Principalmente, se pueden diferenciar dos sectores en el mundo de la automatización dentro de la tecnología: automatización en IT y automatización en decisiones empresariales. En el sector IT se usa la automatización para realizar la repetición de procesos para ejecutar tareas. En cuanto al sector empresarial, la automatización se utiliza para la gestión de procesos y reglas empresariales. Las ventajas que presenta esta tecnología son: mayor productividad, mantenimiento del sistema más sencillo y menor fallo de error. En cuanto las desventajas, tanto el coste como el nivel de aprendizaje son los principales factores a considerar.

Como hemos nombrado al principio del capítulo 2, existen diferentes herramientas para automatizar un sistema. Ansible ha sido la herramienta de automatización elegida para usar en este proyecto.

2.4.1 Herramienta Ansible

La herramienta Ansible es una aplicación para gestionar y administrar sistemas. Se realizarán de manera automática las tareas y procesos repetitivos dentro de un sistema. Una de las ventajas de Ansible es que no usa un agente para realizar la conexión con el nodo destino, sino que usa la conexión ssh para establecer comunicación. El lenguaje que se usa para realizar los archivos de gestión es YAML.

2.4.2 Lenguaje YAML para la gestión de Ansible

El lenguaje más usado para desarrollar este proyecto ha sido YAML. YAML es un lenguaje de marcado que se usa en la herramienta Ansible para definir los roles y los playbooks. YAML se creó con la intención de representar adecuadamente listas, mapeos y valores simples. Siguiendo esta creencia, se creó un formato de fácil

interpretación para los humanos y un código muy estructurado. XML, C y Python son los lenguajes que han servido de referencia para la creación del lenguaje YAML. JSON y XML son los otros lenguajes de marcado más utilizados a día de hoy, pero en Ansible se decidió utilizar YAML por su facilidad de lectura y escritura. Como ejemplo, en la Figura 7 se pueden observar algunas de las diferencias entre XML, JSON y YAML. Por ejemplo, YAML no usa ni llaves ni etiqueta de cierre sino que funciona por tabulación. El argumento que va a contener una serie de reglas es el que esta más a la izquierda seguido de dos puntos como se puede observar con el nombre "Servers" dentro de la Figura 7. Todo lo que este contenido dentro de "Servers", tendrá una tabulación mayor que el propio "Servers".

XML	JSON	YAML

<Servers>
<Server>
<name>Server1</name>
<owner>John</owner>
<created>123456</created>
<status>active</status>
</Server>
</Servers>
	{
	Servers: [
	{
	name: Server1,
	owner: John,
	created: 123456,
	status: active
	}
	]
	}
		Servers:
		- name: Server1
		owner: John
		created: 123456
		status: active

Figura 7: Diferencia entre XML, JSON y YAML

2.5 Servidor Apache

Apache es un servidor web de licencia gratuita que ofrece una gran cantidad de recursos para ejercer de servidor. Su objetivo es ofrecer a los usuarios eficiencia, seguridad y un servidor que se adapte a las necesidades del sistema.

2.6 Servidor Tomcat

Tomcat es un contenedor de servlets que se utiliza en la implementación para Java Servlet[2] y JavaServer Pages. Incluye el compilador Jasper, que compila JSPs[3] convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta con frecuencia en combinación con el servidor web Apache.

2.7 Selección del sistema Operativo

El sistema operativo elegido para nuestras máquinas es CENTOS. Este sistema operativo es la versión gratuita del sistema operativo RED HAT, de la propia empresa RED HAT. Esta empresa ofrece su sistema operativo tanto para servidores como para Workstation, así como diferentes productos, como por ejemplo servidores de aplicaciones (RED HAT JBOSS). La principal diferencia entre RED HAT y CENTOS, obviando la licencia, es el soporte técnico. En el caso de CENTOS existe una gran comunidad que respalda el sistema operativo, como ocurre en otras ocasiones con el software libre. CENTOS se define como un sistema robusto, estable y fácil de instalar. Además, funciona con un hardware básico y no necesita de gran potencia ni de grandes prestaciones para su uso.

[2] Servlet: Son pequeños programas escritos en Java que admiten peticiones a través del protocolo HTTP

[3] JSP: JavaServer Pages es una tecnología que permite generar contenido dinámico con documentos HTML o XML

3. Diseño del sistema

En la empresa se intentó que los servidores de aplicaciones y los servidores web estuvieran conectados entre sí por un balanceador, tal como se observa en la Figura 8. En la Figura 8 apreciamos la arquitectura básica del sistema. Los clientes a través de los servidores de aplicaciones realizan peticiones, el balanceador se encarga de repartir la carga de los servidores de aplicaciones entre los diferentes servidores web instalados y, estos servidores webs tratan las peticiones y devuelven una respuesta al servidor de aplicación. Como se ha comentado en los requerimientos, los servidores webs usados son servidores Apache, los servidores de aplicaciones son Tomcat y el balanceador es un módulo por software llamado JK_Module. La herramienta de automatización que se usó en este proyecto fue Ansible. Ansible se utilizó para configurar el sistema, desde su puesta en marcha hasta el despliegue o eliminación de más servidores webs y aplicaciones.

La persona responsable de controlar el sistema puede escalarlo manualmente en caso de que el sistema lo requiera. Se escala, si es necesario, tanto los servidores webs como los servidores de aplicaciones para poder incrementar la cantidad de clientes que hacen peticiones a los servidores de aplicaciones (contiene una aplicación). Además, aumentar la demanda conlleva un aumento de peticiones que recibe el balanceador y distribuye a los servidores web, cuantos más servidores hayan más rápido se podrá repartir la carga.

A continuación, detallaremos como funciona Ansible para entender la implementación del sistema.

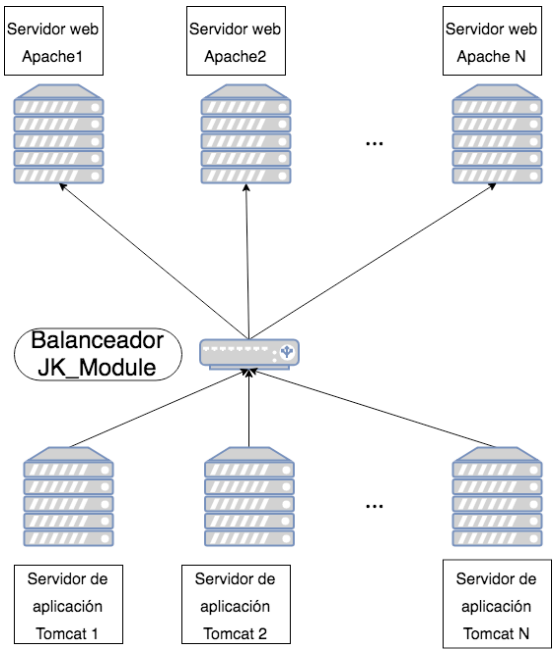

Figura 8: Diseño de la estructura del proyecto a realizar

3.1 Diseño de automatización bajo Ansible

Ansible no usa agente para realizar la conexión con los nodos sino que contiene módulos en los nodos que permiten realizar la conexión mediante conexión ssh. Además, se agrega el archivo de host a las máquinas nodos para poder saber con que dispositivos pueden establecer conexión. Ansible usa una estructura muy definida para su funcionamiento. Por un lado tenemos los archivos internos, como es el archivo ansible.cfg, el cual contiene la configuración de la herramienta en sí. Por otro lado tenemos los roles y los playbooks que son los archivos principales para realizar las instalaciones pertinentes en los nodos destinos.

3.1.1 Ansible.cfg

El archivo ansible.cfg permite configurar personalmente la herramienta Ansible. Es decir, se puede modificar la configuración predeterminada por una configuración más personalizada. Por ejemplo, se podrán modificar las rutas de almacenamiento de los

ficheros, cambiar el aspecto visual y configurar como van a ser las conexiones ssh entre otras opciones.

3.1.2 Roles

Se denomina role a la configuración de la instalación de un servicio o aplicación que se instala en los nodos. Cada role contiene los pasos de configuración necesarios para que la instalación funcione en el nodo destino. La configuración tanto puede ser un role muy sencillo en el cual solo se utilice un archivo básico que contengan líneas de código (por ejemplo, *main*), como puede ser un role más complejo en el que se deban utilizar plantillas y varios archivos junto con el archivo básico para poder proceder a la instalación.

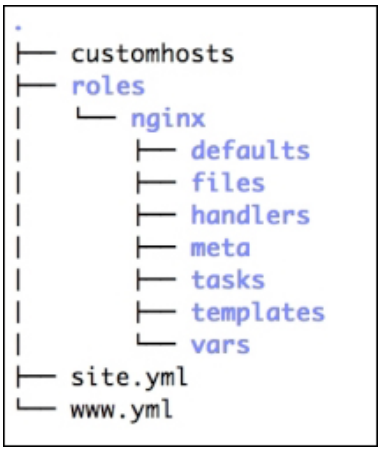

Figura 9: Estructura de los ficheros de los roles en Ansible

Por defecto, los roles están estructurados por el conjunto de ficheros (destacados en color azul en la Figura 9). Como máximo hay un total de siete carpetas. La carpeta "Defaults" contiene un archivo en él que se indicarán los valores de las variables globales. En la carpeta "Files" encontramos los archivos necesarios para el despliegue de la aplicación los cuales se van a copiar exactamente sin poder realizar modificaciones. La carpeta "Templates" contiene archivos muy parecidos a los "Files" pero con la diferencia que sí se pueden modificar, utilizando Jinja2 [21], y son usados como plantillas. La carpeta "Handlers" presenta notificaciones para realizar una acción

una sola vez (no podrá repetirse). La carpeta ”Meta” contiene las dependencias del role y también el entorno, autor, etc. La carpeta “Task” presenta el archivo principal en el que se indican las tareas que se deben ejecutar para la instalación. Por último, la carpeta “Vars” contiene también los valores de las variables con la diferencia que en esta carpeta las variables van a ser más determinantes, es decir, si está la misma variable en “Defaults” y en “Vars” se va a elegir el valor de la variable de la carpeta “Vars” por defecto. A pesar de que es recomendable tener todas las carpetas para realizar los roles en Ansible, no es necesario que estén todas si no se van a utilizar. Para los archivos de la carpeta de “Task” se utiliza la sintaxis YAML, de ahí que la extensión del archivo sea: main.yml.

3.1.3 Playbooks

Los playbooks son archivos encargados de ejecutar los roles ya definidos y de instalar los ficheros (como por ejemplo las dependencias) necesarios en los nodos para que los roles previamente instalados funcionen correctamente. Los playbooks usan también la sintaxis YAML y en estos playbooks se especifican a que nodos destinos van a instalarse los roles. También pueden contener variables y líneas de código, para ejecutar algún proceso sencillo.

En el ejemplo de la Figura 10, se define un archivo playbook que realiza la instalación de un role llamado JBOSS en el host destino. El código del archivo está en formato YAML. Al invocar/llamar el playbook, se especifica la IP o IP’s de las máquinas destinos a través de la variable {{target}}. La línea 4 de la Figura 10 define que va a lanzar la instalación del role en modo root en la máquina destino.

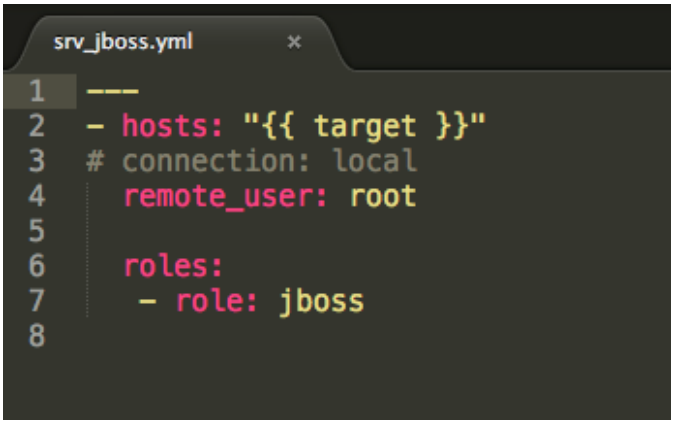

Figura 10: Configuración de un Playbook en Ansible

Para tener una visualización de cómo esta estructurada la herramienta Ansible podemos observar la Figura 11. En el primer nivel observamos el archivo de configuración ansible.cfg, la carpeta de hosts, la carpeta Playbooks y la carpeta Roles. Dentro de la carpeta Playbooks existen los archivos que se pueden ejecutar desde la herramienta Ansible. En este caso, vemos *httpd.yml* el cual hace referencia al role del servidor Apache, *Srv_Consul.yml* para indicar el de Consul y el de *Tomcat.yml* para lanzar el role de Tomcat. En la carpeta roles, existen las carpetas de los diferentes roles. Dentro de cada role, se suelen crear por defecto las siete carpetas ya nombradas en el punto 3.1.2 y dentro de cada una de estas carpetas están los ficheros a ejecutar para la instalación del role.

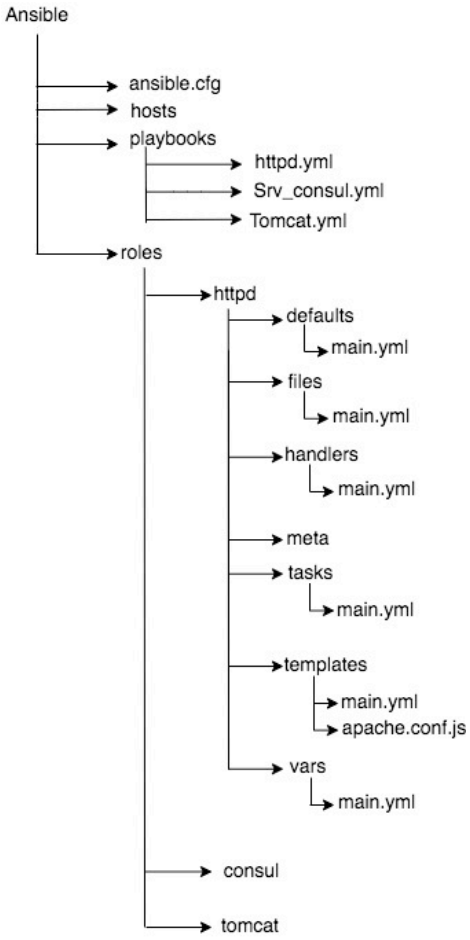

Figura 11: Estructura del contenido de la herramienta Ansible

4. Implementación

Como se ha comentado anteriormente, la implementación gira entorno la herramienta de automatización llamada Ansible. A través de Ansible se ha realizado la configuración del sistema, además, los roles y los playbooks han permitido la instalación en cada una de las máquinas. Se implementaron los roles más destacables para dicha instalación que son los siguientes: los roles de servidores Apache, los roles de Tomcat y el role de Consul. Estos tres roles están presentes en todas las instalaciones que se realizan en las nuevas máquinas. Una vez la instalación está finalizada, utilizamos la herramienta Consul para verificar que todas las máquinas que formarán parte de nuestro sistema están en funcionamiento. Además, se realizan las comprobaciones de la funcionalidad del sistema.

4.1 Configuración de Ansible

La herramienta Ansible permite modificar la configuración por defecto de los diferentes componentes mediante el fichero dev.yml. En nuestro caso, se modificó el fichero dev.yml para poder trabajar conjuntamente con Ansible, VMWARE y Consul. En la Figura 12, se observa cómo se han tratado las modificaciones que se han realizado para poder trabajar con Ansible y VMWARE. En el primer párrafo mostramos la identificación utilizada para poder conectar con el Vcenter [22]: identificador, nombre del datacenter y red en la que se va a trabajar.

En la Figura 12 (líneas 5:6), se indica la dirección del Vcenter y el nombre del CPD(datacenter) al que nos vamos a conectar. Además, en la Figura 12 (líneas 15:20) están incluidas las credenciales del usuario para poder conectar con el Vcenter, que es la carpeta en donde la que se guardarán las nuevas máquinas creadas y la dirección DNS que se utiliza. En el archivo de hosts, encontramos las máquinas que se encuentran al alcance de Ansible, es decir, las máquinas sobre las que Ansible podrá ejecutar algún role sobre ellas. Como este software no utiliza agente, se usa conexión ssh para poder mantener una sesión remota.

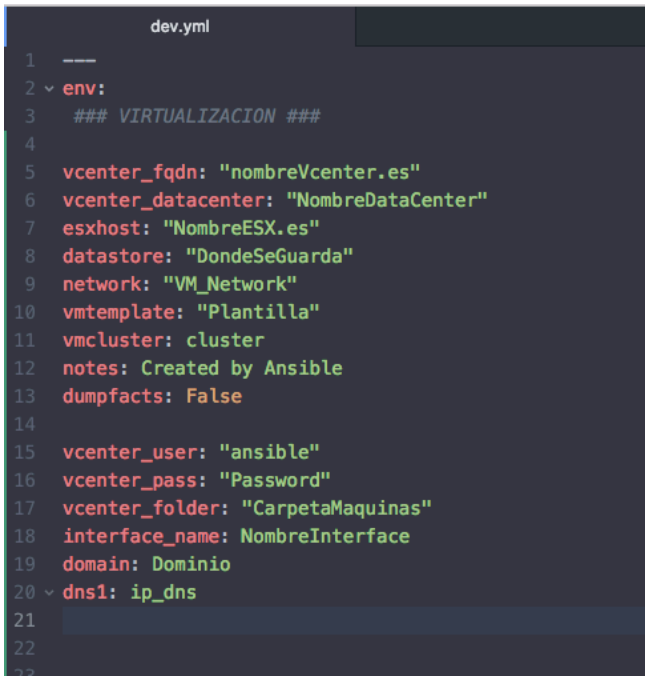

Figura 12: Fichero configuración VMWARE

4.1.1 Implementación de Consul

La implementación del software Consul se ha hecho mediante un role de Ansible llamado “Consul”. En el role se especifica como se va a realizar la instalación y los procesos que se van a llevar cabo. Una vez la instalación del role haya terminado, el software Consul estará en funcionamiento y bien configurado. El role consul contiene tres archivos de configuración. El primer role, la Figura 13, es el “main.yml” y su objetivo es indicar el sistema operativo sobre el cual se realizará la configuración y cual será el siguiente role a ejecutar. El segundo role, la Figura 14, es el “client.yml”. Este role contiene la instalación propia de la herramienta Consul. La instalación que se ejecuta es la que se debe realizar en todas las máquinas que presenten el sistema operativo CentOs. En la Figura 14(líneas 5:26) se observa el proceso de descargar desde la web el software Consul, descomprimirlo en la máquina destino e indicar la carpeta a la cual se va a posicionar. Además, en la Figura 14(líneas 48:54) también se muestra como al finalizar la instalación se establece el servicio Consul “enable” para que este activo dentro de la máquina. En el tercer role, en la Figura 15, hay el archivo “register_service.yml”. Este último role se encarga de registrar el servicio que se va a

crear dentro de la base de datos de Consul. El nombre del nuevo registro, que estará asociado al servicio, se indica por la variable llamada “ítem.name”.

main.yml

```yaml
---
# tasks file for consul

- name: "CONSUL SERVICE DISCOVERY | Check for supported platform (RedHat | Oracle | CentOS Linux 6.6 or 7)"
  assert:
    that:
      - ansible_os_family == 'RedHat'
      - (ansible_distribution_version | version_compare('6.6', '=') ) or (ansible_distribution_version | version_compare('7.1', '>=') )

- name: "CONSUL SERVICE DISCOVERY | Executing {{ task }}"
  include: "{{ task }}.yml"

```

Figura 13: Archivo main del role Consul

```yaml
---
    - name: Creates directory
      file: path=/tmp/consul state=directory

    - name: download consul
      get_url:
        url: https://releases.hashicorp.com/consul/0.8.1/consul_0.8.1_linux_amd64.zip
        dest: /tmp/
        validate_certs: no

    - name: extract consul
      unarchive:
        src: /tmp/consul_0.8.1_linux_amd64.zip
        dest: "/tmp/consul"
        remote_src: True

    - name: Creates directory
      file: path=/etc/consul/consul.d/client state=directory

    - name: Creates directory
      file: path=/etc/consul/consul.d/bootstrap state=directory

    - name: install consul bin file
      shell: mv consul /usr/local/bin/
      args:
        chdir: /tmp/consul

    - name: get hosts in cluster
      uri:
        url: "[redacted]v1/catalog/service/consul"
        method: GET
        return_content: yes
      register: cluster_members
      #delegate_to: localhost
      become: no

    - set_fact:
        consul_servers_members: []

    - set_fact:
        consul_servers_members: "{{consul_servers_members + [item['Address']]}}"
      with_items: "{{cluster_members.json}}"


    - name: configure consul
      template: src=client.json.j2 dest=/etc/consul/consul.d/client/config.json

    - name: create daemon consul
      copy: src=consul_agent.service dest=/etc/systemd/system/consul.service owner=root group=root mode=0644
      register: consul_service_file
      ignore_errors: True
      notify:
          - start consul
          - enable consul

```

Figura 14: Archivo Client del role de Consul

register_services.yml

```
1  ---
2  - name: Registering new service {{ item.name }}
3    template:
4      src='register_service.json.j2'
5      dest='/etc/consul/consul.d/client/{{ item.name }}.json'
6    with_items: "{{ consul_services }}"
7
```

Figura 15: Archivo Register_service del role Consul

4.2 Servidor Apache y su utilización en Ansible

Como se ha comentado anteriormente, en nuestro proyecto, el servidor Apache funciona junto con un balanceador para mejorar la disponibilidad del sistema dentro de nuestro proyecto. El balanceador distribuye las peticiones que recibe, a través de los servidores Tomcat, a los diferentes servidores Apache que hay en nuestro sistema. Esto servidores Apache tratan la petición que se les asocia y devuelven la respuesta al cliente.

```
19  ###################################################################
20  ##                                                                 #
21  ##  Install hhtpd                                                  #
22  ##                                                                 #
23  ###################################################################
24
25  - name: Install apache with yum
26    yum:
27      name: httpd
28      state: latest
29    notify:
30      - enable apache
31      - start apache
32
33
34  - name: Creates directory
35    file:
36      path: "{{ path_apache }}"
37      state: directory
38      owner: apache
39      group: apache
40      mode: 0775
41
42
43  ###################################################################
44  ##                                                                 #
45  ##  Configure apache file: httpd.conrf                             #
46  ##                                                                 #
47  ###################################################################
48
49  - name: Apache config
50    template:
51      src: apache.conf.j2
52      dest: "{{ path_conf }}/httpd.conf"
53      owner: root
54      group: root
55      mode: 0644
56    notify:
57      - restart apache
58
```

Figura 16: Configuración del role Apache para CentOS en Ansible

La automatización de la instalación de Apache dentro del sistema operativo desplegado en cada una de las máquinas se realiza siguiendo las mismas instrucciones básicas que un usuario utilizaría, pero usando la sintaxis de Ansible. Como el sistema operativo es CentOS, un usuario hace la instalación del servicio Apache a través de la línea de comandos y usa el paquete de instalación yum [23]. Una vez finalizada la instalación se inicia el servicio. En el caso de Ansible, los comandos utilizados son similares, la invocación de yum se especifica en el role (como podemos observar de la línea 25 a la 31 de la Figura 16). Al finalizar esta operación, el comando que contiene notify (líneas 56:57 de la Figura 16) habilita e inicia el servicio de Apache.

```
31  ##                                                                  #
32  ######################################################################
33
34  - name: Download tomcat connnectors
35    get_url:
36      url:
37        "{{url_tomcat_connector}}"
38      dest:
39        "{{dest_path}}"
40
41  - name: Descomprimir
42    unarchive:
43      src: /opt/tomcat-connectors-1.2.42-src.tar.gz
44      dest:
45        "{{dest_path}}"
46      copy: no
47
48  - name: Execute configuration
49    command: sh configure --with-apxs=/usr/bin/apxs --enable-api-compatibility
50    args:
51      chdir: /opt/tomcat-connectors-1.2.42-src/native
52
53  - name: Execute make an make Install
54    command: make
55    args:
56      chdir:  /opt/tomcat-connectors-1.2.42-src/native
57
58  - name: Execute make an make Install
59    command: libtool --finish /usr/lib64/httpd/modules
60    args:
61      chdir: /opt/tomcat-connectors-1.2.42-src/native
62
63  - name: Execute make an make Install
64    command: make install
65    args:
66      chdir: /opt/tomcat-connectors-1.2.42-src/native
67
68  ######################################################################
```

Figura 17: Configuración de Tomcat Connectors para una máquina CentOS

Una vez instalado y configurado el servidor de Apache debemos configurar la automatización de los conectores usando el servidor Tomcat. En la Figura 17 observamos el conjunto de comandos que ejecuta Ansible para instalar los conectores de Tomcat. Estos comandos descargan los conectores desde la web de Apache y guardan el archivo en la dirección destino (dest_path). Posteriormente, se descomprime el archivo y se ejecuta incluyendo los módulos necesarios para que tengan conexión servidores y balanceador.

Para la instalación del modulo JK se usan dos ficheros, uno de configuración del propio módulo (instalación explicada con el ejemplo de la Figura 17), y otro fichero para la configuración del servidor Apache. La Figura 18 contiene los pasos a seguir para direccionar los archivos a la carpeta que le corresponde. En la Figura 18, de la línea 36 a la 46 se redireccionan los ficheros virtualhost.conf y jk.conf2 en la dirección que el servidor apache contiene sus archivos. De la línea 48 a la 53 (Figura 18) se muestra una

modificación en la configuración. Esta modificación sirve para que la conexión de los servidores Apache con el modulo Jk sea funcional y se puedan comunicar entre ellos.

```
30  ##################################################################
31  ##                                                              #
32  ##  Reconfigure files to module jk                              #
33  ##                                                              #
34  ##################################################################
35
36  - name: template virtualhost.conf
37    template:
38      src: virtualhost.conf
39      dest: /etc/httpd/conf.d/
40      force: no
41
42  - name: template jk.conf
43    template:
44      src: jk.conf
45      dest: /etc/httpd/conf.d/
46      force: no
47
48  - name: template workers.properties
49    template:
50      src: "{{ workers_properties }}.j2"
51      dest: "{{ path_workers_properties }}"
52    notify:
53      - reload apache
54
```

Figura 18: Configuración del JK Module para una máquina CentOS con apache instalado

4.3 Tomcat y su configuración dentro del proyecto

Para nuestro proyecto, hemos usado el servidor Tomcat como servidor de aplicación. El proyecto consiste en lanzar una aplicación web o numerosas aplicaciones web en el entorno Tomcat, el cual funciona como un contenedor, y de este modo se puede desplegar la aplicación de una forma controlada. Este sistema permite controlar las aplicaciones que han sido lanzadas dentro del servidor Tomcat. Además, este sistema permite tener la aplicación en un almacenamiento controlado, por lo que las operaciones críticas que puedan surgir afectarán en menor medida al conjunto del sistema.

La instalación de Tomcat dentro del proyecto se realiza en todas las máquinas de forma automática y precisa. Las credenciales son comunes para todas las máquinas que contienen Tomcat. Gracias a esta instalación, se activa el servicio Tomcat y

podemos comprobar si la operación ha sido un éxito accediendo a la dirección IP del Tomcat.

```
- name: install tomcat
  yum:
    name: tomcat
    state: present
    disable_gpg_check: yes

- name: Add line java_opts
  lineinfile:
    dest: /usr/share/tomcat/conf/tomcat.conf
    line: 'JAVA_OPTS="-Djava.security.egd=file:/dev/./urandom -Djava.net.preferIPv4Stack=true -Djava.awt.headless=true

- name: Install webapps tomcat
  yum:
    name: tomcat-webapps, tomcat-admin-webapps, tomcat-docs-webapp, tomcat-javadoc
    state: latest
    disable_gpg_check: yes

- name: users.xml
  template:
    src: tomcat-users.xml.j2
    dest: /usr/share/tomcat/conf/tomcat-users.xml

- name: Enable and start service
  service:
    name: tomcat
    state: started
    enabled: yes
```

Figura 19: Configuración del role Tomcat en Ansible

En la Figura 19 observamos como es el role Tomcat. Primero se hace una instalación sencilla de Tomcat y, posteriormente, se añaden las dependencias necesarias. Además, en este role, se realizarán las modificaciones necesarias para que la instalación funcione. Por ejemplo, en la Figura 19, en el archivo de users.xml usamos una plantilla para que el usuario y la contraseña de los servidores Tomcat sean los mismos. Por último, se inicializa el servicio Tomcat para su uso en las máquinas.

Una vez configurado el entorno de automatización con Ansible, la herramienta Consul, el entorno de los servidores web con apache, los servidores con Tomcat y el modulo JK para hacer las funciones de balanceador, ya habremos alcanzado la automatización que perseguíamos en nuestro objetivo. En el siguiente capítulo veremos los resultados obtenido con todo el proyecto montado.

5. Resultados

Para confirmar la correcta configuración de los ficheros de automatización se decidió realizar una serie de pruebas. A continuación, se específica el entorno donde se realizaron las pruebas y el tipo de pruebas que se realizó.

Las pruebas se realizaron en el entorno que estaba virtualizado con VMWARE el cual fue creado para el proyecto. Utilizamos la herramienta Ansible para crear y configurar todos los dispositivos necesarios para crear un sistema funcional. Una de las pruebas consistía en crear diferentes máquinas con el servicio Tomcat, máquinas con el servidor web Apache y el modulo JK para ver como se repartía la carga. Tanto las máquinas con Tomcat como las máquinas con el servidor Apache formaron parte del servicio Consul.

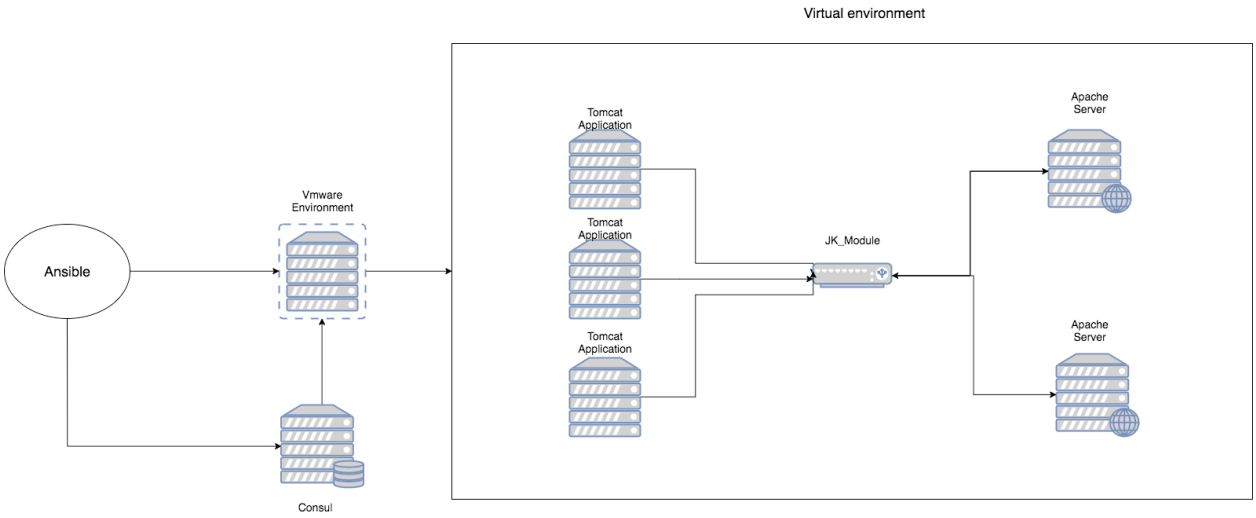

Figura 20: Desarrollo de procesos en el proyecto

En la Figura 20 se puede observar como se realiza la instalación del sistema del proyecto. La primera acción que se ejecuta es lanzar la herramienta Ansible que desencadena una serie de ejecuciones y procesos para montar el sistema. En el servidor virtualizado con VMWARE se crean las máquinas con el sistema operativo CentOS. Todas las máquinas se añaden a la herramienta Consul a través de su IP, para poder mirar el estado de cada máquina. Estas máquinas pueden tener instalado el

servicio Tomcat y se utilizan como servidores de aplicaciones y también el servicio Apache que funcionan como servidores webs. Las máquinas con Tomcat se encargan de realizar peticiones a los servidores web. Además, estas máquinas para poder repartir la carga entre los diferentes servidores web necesitan de un balanceador por software entre las diferentes máquinas. Está representado como ejemplo dentro del recuadro en la Figura 20.

En la Figura 21 observamos una ejecución de la herramienta Ansible en la cual se hace una creación de una máquina nueva. En las primeras 10 líneas se crea el espacio en VMWARE y se le asocia una IP a la máquina. En las siguientes líneas hasta el final de la figura se realiza el aprovisionamiento del espacio en el disco duro. En la Figura 22 se realiza la configuración de red, el DNS y el proceso de inscripción de la máquina en el sistema Consul.

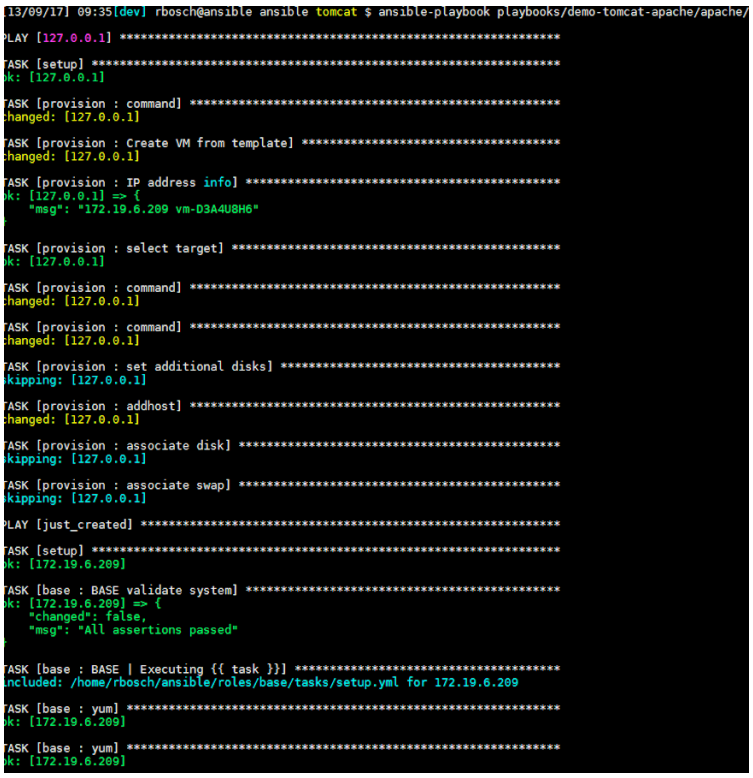

Figura 21: Creación de las nuevas máquinas CentOS

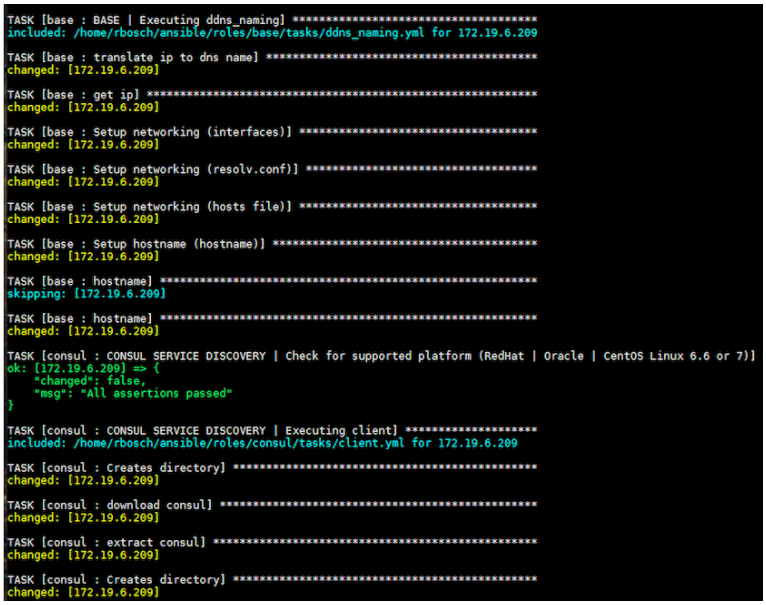

Figura 22: Instalación de la herramienta Consul dentro de una máquina CentOS

En cambio, la Figura 23 muestra entre las líneas 6 y 10 la instalación del servicio Apache y de los conectores Tomcat para comunicarse con las máquinas Tomcat. Finalmente, el sistema nos indica que la instalación se ha ejecutado correctamente y el número de cambios que se han aplicado durante la ejecución.

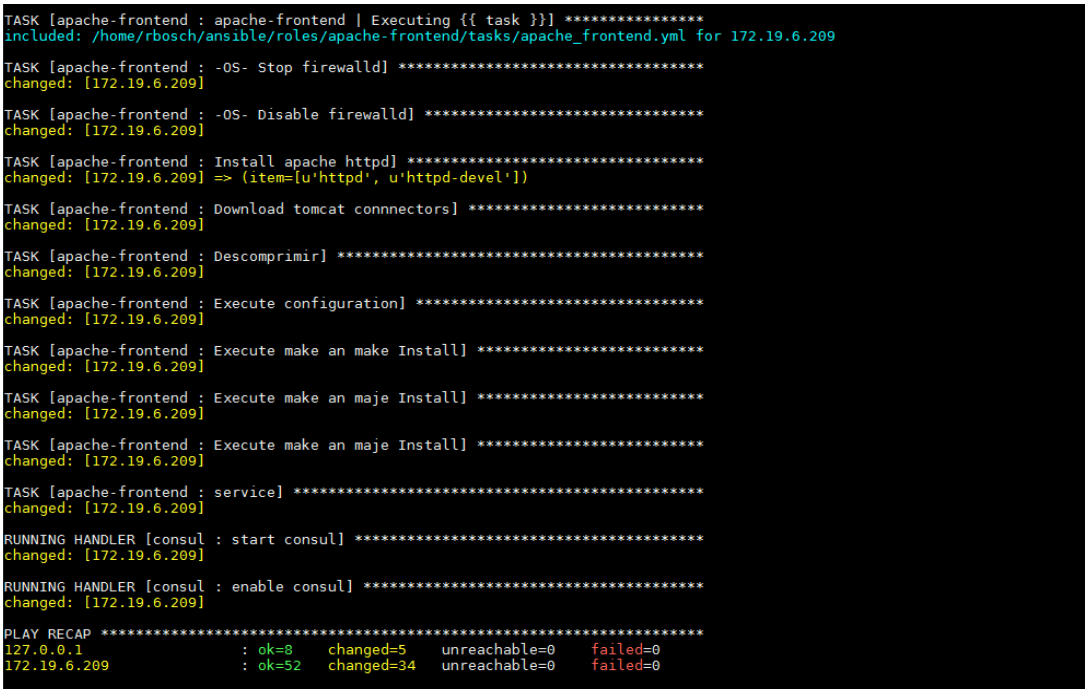

Figura 23: Instalación del role Apache en una de las máquinas CentOs

Una de las formas de comprobar que el servicio Tomcat y del servidor Apache están instalados y funcionando es mediante la búsqueda en el navegador de http://localhost:8080/ para Tomcat y http://localhost/ para Apache. Si se puede observar el contenido como en la Figura 24 y la Figura 25, significa que los servicios Tomcat y Apache están activos y funcionando. Existen otras maneras de verificar que los servicios están activos. En el caso de Tomcat, una de ellas, se puede ejecutar el comando wget http://localhost:8080 en la terminal y ver la respuesta que se obtiene. Para Apache, también en la terminal, se puede usar el comando "service apache2 status".

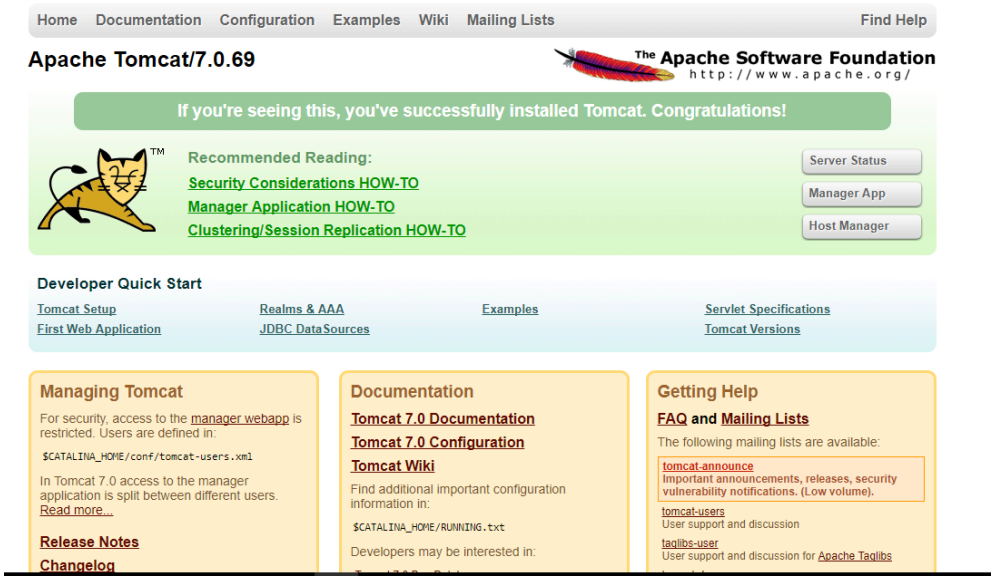

Figura 24: Página inicial Tomcat si está correctamente instalado

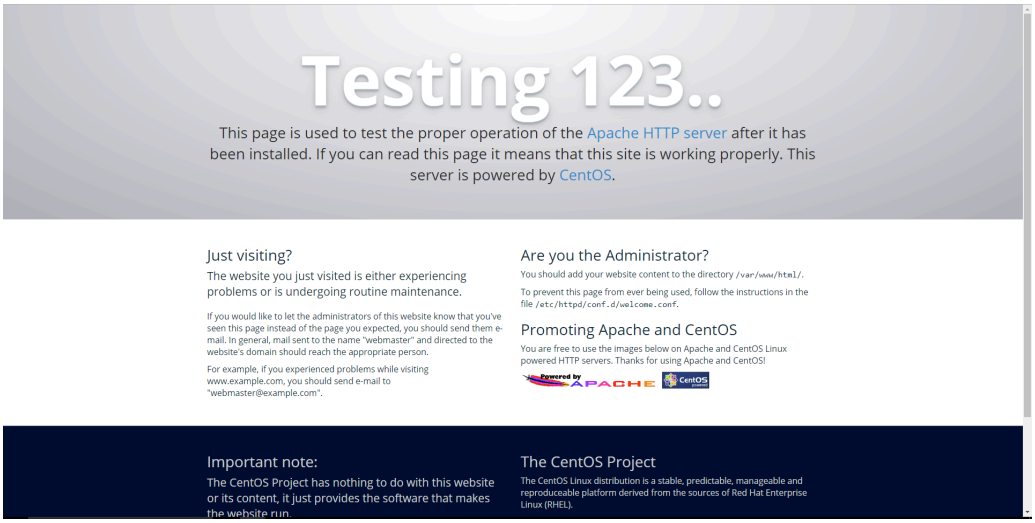

Figura 25: Página de inicio del servidor Apache si está correctamente instalado

La tercera prueba consistía en comprobar el correcto funcionamiento de Consul. Para ello se procedió a eliminar máquinas que ya se habían creado previamente.

En la Figura 26 observamos la página inicial de Consul donde se muestran las máquinas que presentan Tomcat instalado. Si las máquinas están en verde significa que el servicio está activo y funcionando, en el caso de que estuviera en rojo significaría lo contrario. En esta página web de Consul (Figura 26) se van agregando o eliminando las máquinas en función si se modifican en el entorno VMWARE, como se observa en la Figura 27. En el entorno VMWARE, las máquinas están identificadas por un nombre dado por parámetro o por una variable interna al lanzar la acción de Ansible. En cambio, en el Consul tenemos las máquinas identificadas por su dirección IP.

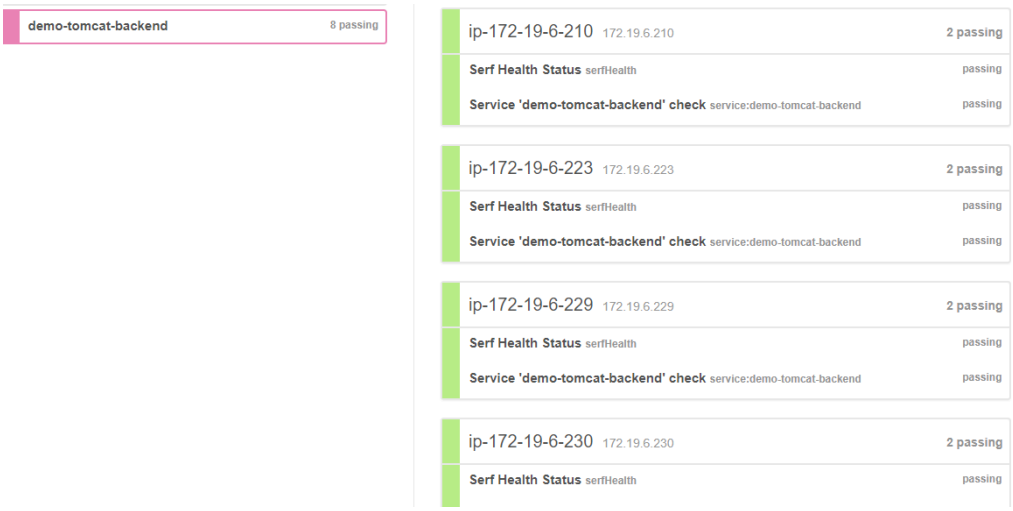

Figura 26: Página inicial de la herramienta Consul

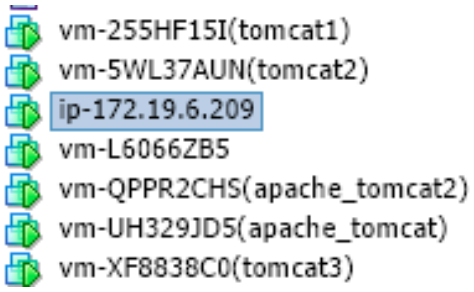

Figura 27: Máquinas virtuales dentro del entorno VMWARE

Por último, se comprobó si el sistema funcionaba. Para está comprobación, se creó otra máquina con el servicio Apache. Junto con dos máquinas Tomcat que ya teníamos activas, pudimos verificar si balanceaba la carga correctamente. Para la verificación se utilizó el comando "*ping*" en la terminal y se obtuvo la respuesta del servicio. Si la respuesta que obteníamos hacía referencia a dos IP diferentes, significaba que el balanceador funcionaba y se distribuía la carga. En la Figura 28 se puede ver como al llamar a una máquina con el servicio Tomcat, se obtiene la primera vez una respuesta desde la IP: 172.19.6.211. La segunda vez que se llama al mismo servicio Tomcat, la respuesta viene desde la IP: 172.19.6.222. En la tercera llamada al mismo servicio, la respuesta vuelve a venir dada por la primera IP, la IP: 172.19.6.211.

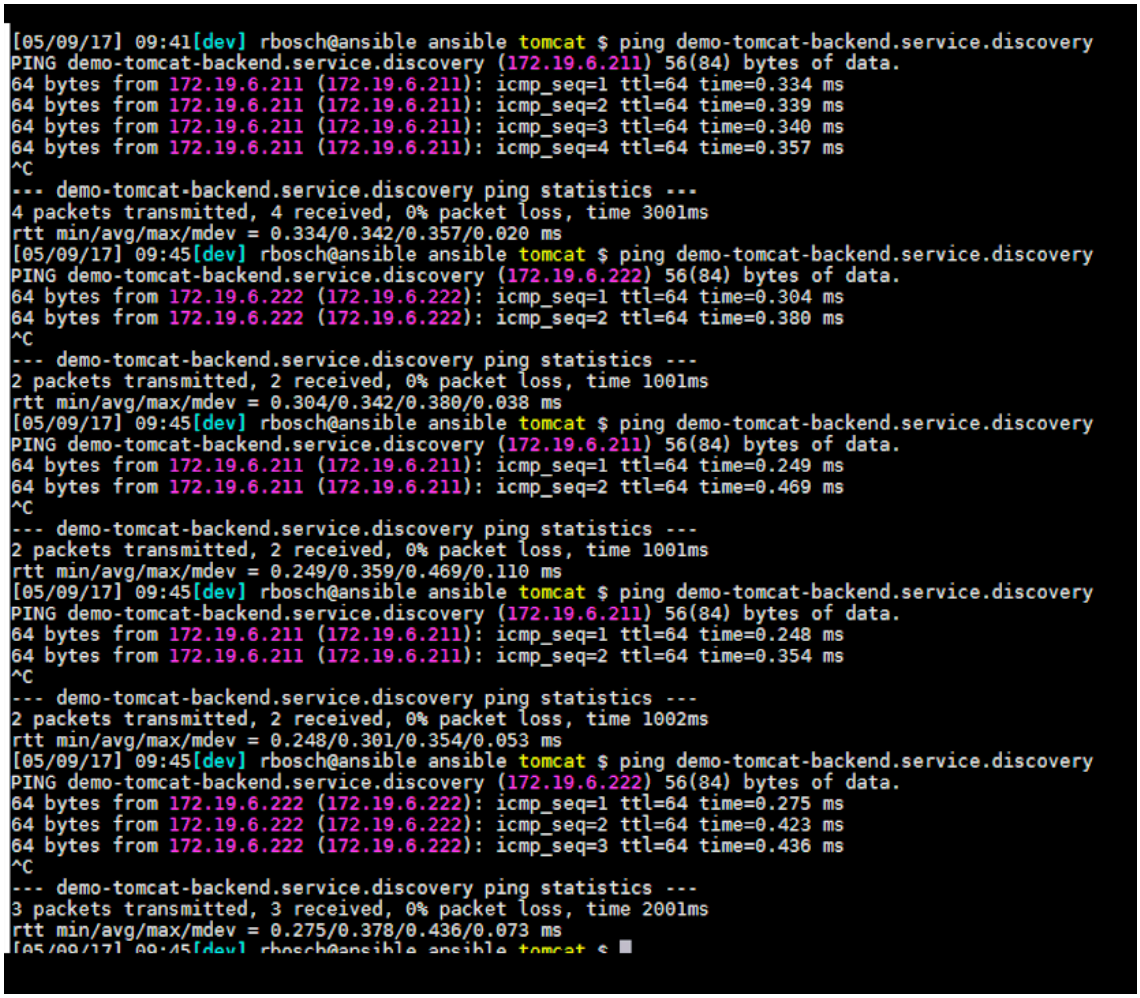

Figura 28: Ejemplo de balanceo de peticiones web entre dos máquinas.

Al obtener respuestas diferentes al llamar al mismo servicio llegamos a la conclusión que el balanceador de carga funcionaba. Después de verificar que todo el sistema estuviera montado y funcionando correctamente se realizó la verificación de si realmente funcionaba cumpliendo el propósito del proyecto, obteniendo un resultado satisfactorio como ya hemos comentado unas líneas más arriba.

6. Conclusiones

Este proyecto puede ayudar notablemente las entidades que busquen automatizar su arquitectura web usando herramientas de automatización para obtener una mejora en su rendimiento, flexibilidad y costes ya que simplifica el trabajo en más de un ámbito. Además, es capaz de ir escalando el diseño (a petición nuestra) según los requisitos del momento.

Personalmente, considero que las tecnologías de automatización como Ansible o de virtualización como VMWARE son herramientas que bien usadas pueden ayudar en gran cantidad a realizar proyectos en menor tiempo de lo común. Ansible y VMWARE son dos herramientas que mejoran el entorno de trabajo, pudiendo llegar a simplificar mucho los posibles problemas en arquitecturas con servidores. En este caso, contamos con dos herramientas que mejoran el hardware a través del software y también el tiempo de ejecución de las tareas a través de una automatización del sistema.

Finalmente, después de haber participado en este proyecto utilizando estas herramientas durante unos seis meses, puedo decir que estoy muy satisfecho con los resultados obtenidos ya que se ha alcanzado el objetivo principal. Al empezar con este proyecto no tenía ningún conocimiento de la herramienta de automatización Ansible. Por este motivo, el hecho de haber alcanzado el objetivo principal cumpliendo con todos los requisitos del proyecto, me permite valorar todo lo aprendido de forma positiva y me proporciona una gran confianza para la realización de futuros proyectos. Además, durante esta estancia en Brújula he demostrado que soy capaz de llevar a cabo un proyecto completo, funcional y con utilidad, incluso a partir de un software totalmente desconocido para mí, por lo que esta experiencia me ha proporcionado una gran motivación para comenzar nuevos proyectos relaciones con sistemas automatizados.

7. Bibliografía

[1] Jiani Guo, L.N. Bhuyan, "Load Balancing in a Cluster-Based Web Server for Multimedia Applications" https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1642451

[2] Red Hat, https://www.redhat.com/es/topics/automation/whats-it-automation

[3] Laura Grit, David Irwin, Aydan Yumerefendi, Jeff Chase, "Virtual Machine Hosting for Networked Clusters: Building the Foundations for "Autonomic" Orchestration", 2006, https://ieeexplore.ieee.org/document/4299352/

[4] Rahul Godha, Sneh Prateek, "Load Balancing in a Network", 2014, http://www.ijsrp.org/research-paper-1014/ijsrp-p3476.pdf

[5] M. Shreedhar and George Varghese, "Efficient Fair Queuing Using Deficit Round-Robin", 1996 [Online]. Available: http://www.ecs.umass.edu/ece/wolf/courses/ECE697J/papers/DRR.pdf

[6] IBM, "Weighted round robin load balancer" [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/SSCVKV_9.1.1/Campaign/Listeners/WeightedRoundRobin.html

[7] "Least-Connection Scheduling" [Online]. Available: http://kb.linuxvirtualserver.org/wiki/Least-Connection_Scheduling

[8] VMWARE, What is a Hypervisor?, https://www.vmware.com/topics/glossary/content/hypervisor

[9] Rishabh Sharma, Mitesh Soni, "Automate your infrastructure using code and leverage DevOps with Chef", 2015

[10] RED HAT, Puppet Guide, 2018, https://access.redhat.com/documentation/en-us/red_hat_satellite/6.1/pdf/puppet_guide/Red_Hat_Satellite-6.1-Puppet_Guide-en-US.pdf

[11] OpenWebinars, https://openwebinars.net/blog/automatizar-tareas-con-ansible-y-puppet/

[12] National Instruments, "Patrones de diseño de aplicaciones: Maestro/Esclavo", 2015, http://www.ni.com/white-paper/3022/es/

[13] Git, https://git-scm.com/

[14] Oren Ben-Kiki, Clark Evans, Ingy döt Net, “YAML Ain’t Markup Language (YAML™) Version 1.2”, 2009, http://yaml.org/spec/1.2/spec.pdf

[15] Ansible, https://docs.ansible.com/ansible/latest/user_guide/playbooks_reuse_roles.html

[16] Ansible, https://docs.ansible.com/ansible/2.6/user_guide/playbooks_intro.html
[17] Red Hat, https://www.redhat.com/es/topics/virtualization/what-is-virtualization
[18] VMWARE, https://www.vmware.com/latam/products/vsphere.html
[19] Apache, https://httpd.apache.org/
[20] Java Code Geeks, “Apache Tomcat Cookbook”, 2016, http://index-of.es/Varios-2/Apache-Tomcat-Cookbook.pdf

[21] Jinja2 Documentation, 2017, https://media.readthedocs.org/pdf/jinja2/latest/jinja2.pdf
[22] Vcenter , https://www.techopedia.com/definition/26818/vmware-vcenter-server
[23] Yum, CentOS, https://wiki.centos.org/es/PackageManagement/Yum