Orquestación de entornos productivos mediante aplicación web

Iván Jiménez Valencia

Resumen– La gestión de servidores es un problema general en muchas empresas. No disponer de las herramientas adecuadas puede llegar a suponer el fracaso en el desarrollo de algunos proyectos. El detectar una indisponibilidad de servicio, o un problema de despliegue en una gran infraestructura consume tiempo y recursos y a menudo no se realiza de forma óptima. El desarrollo de este proyecto afronta estos problemas mediante la creación de una aplicación web capaz de orquestar servidores cuya finalidad es la de reducir el tiempo de gestión, reducir la cantidad de errores y obtener una mayor productividad. A nivel técnico, Ansible y Dockers serán las herramientas usadas en la orquestación, mientras que Zabbix se integrará como soporte a la monitorización.
Palabras clave– Docker, Ansible, Orquestación, Nodejs, Vuejs, Zabbix

Abstract– Server management is a general problem in many companies. Not having the right tools can lead to failure in the development of some projects. Detecting a service unavailability, or a deployment problem in a large infrastructure consumes time and resources and is often not performed in an optimal way. The development of this project addresses these problems by creating a web application capable of orchestrating servers whose purpose is to reduce management time, reduce the number of errors and obtain greater productivity. At the technical level, Ansible and Dockers will be the tools used in the orchestration, while Zabbix will be integrated to support the monitoring.
Keywords– Docker, Ansible, Orchestration, Nodejs, Vuejs, Zabbix

1 Introducción

En la actualidad, muchas empresas disponen de un conjunto de servidores que se necesitan gestionar. La gestión de estos servidores causa a menudo problemas al ámbito TIC, en particular si no se dispone de las herramientas adecuadas. Esto se debe a que en particular en grandes infraestructuras el despliegue es una tarea repetitiva que puede generar problemas, una menor productividad y finalmente pérdida de tiempo.

Este proyecto ha sido desarrollado para la empresa Sistemas Informáticos Abiertos (SIA), pero es aplicable para cualquier empresa que requiera gestionar sus servidores.

En este documento se describe de manera detallada el problema mencionado anteriormente y se propone una solución basada en la combinación de una solución de gestión de despliegues y un entorno de monitorización. Seguidamente se explica cuál ha sido el procedimiento para su desarrollo y la planificación que este ha tenido. Finalmente, se valoran los resultados obtenidos y se exponen las conclusiones de este trabajo.

• E-mail de contacto: ivanjimenezvalencia4@gmail.com
• Mención realizada: Tecnologías de la Informació
• Trabajo tutorizado por: Juan Antonio Martínez Carrascal (DEIC)
• Curso 2021/22

2 Descripción del problema

Como se ha comentado en la introducción, es difícil gestionar servidores y realizar tareas repetidamente en medianas y grandes infraestructuras. Una simple tarea, como por ejemplo actualizar el sistema operativo, o simplemente crear un fichero de configuración, a primera vista no parece complicado. Pero si pensamos en empresas que disponen de grandes cantidades de servidores esta gestión se convierte en un problema debido a que tenemos que realizar las tareas repetidas veces, en ocasiones por equipos de personas diferentes, cosa que provoca incrementos de tiempo y esfuerzo, así como mayor número de errores. El origen de este proyecto está en la detección de la vulnerabilidad CVE-2021-44228 que afecta a la librería log4j [1]. La mitigación de la vulnerabilidad requiere un despliegue masivo, pues afecta a todos los servidores. En ese momento se tienen que realizar una gran cantidad de cambios, primeramente, migrando las librerías en los entornos de preproducción con la finalidad de comprobar si los cambios han sido realizados correctamente y, una vez esto es verificado, hay que realizar los cambios en el entorno de producción.

3 Solución propuesta

Frente a este problema, se propone una solución que se describe a continuación.

Se ha desarrollado una aplicación web que es capaz de orquestar todos los servidores de la infraestructura. Esta aplicación permite a los usuarios instalar paquetes de software y gestionar servicios en todos los servidores que se indiquen a través de esta. Por otro lado, dispone de un módulo avanzado, que permite al usuario programar las tareas a realizar de forma personalizada.

En un inicio, la aplicación tenía como objetivo el poder desplegar configuraciones básicas, pero gracias al software que se utiliza para la orquestación, la aplicación es capaz de realizar configuraciones mucho más complejas siempre y cuando se disponga de conocimientos en Ansible [2].

A continuación, se especifican los objetivos en los que se basa la solución implementada, los requisitos funcionales, los requisitos no funcionales y cuáles son las restricciones técnicas.

3.1. Objetivos

- Desplegar una arquitectura similar a la de la empresa que permita la simulación del entorno.
- Desarrollar una aplicación web capaz de orquestar los servidores desplegados.
- Monitorizar los servidores mediante la gestión integrada de una aplicación web.

3.2. Requisitos funcionales

- Poder iniciar y cerrar sesión mediante un panel de login.
- Desplegar ficheros de configuración de forma individual en los servidores.
- Desplegar ficheros de configuración en varios servidores a la vez.
- Cambiar el estado de los servicios en cada servidor.
- Cambiar el estado de los servicios en un conjunto de servidores.
- Comprobar el uso de la CPU de los servidores.
- Comprobar el estado de los servidores (activo/caído).
- Monitorizar la memoria Heap que utilizan los servidores.
- Obtener un gráfico por cada tipo de monitorización.

3.3. Requisitos no funcionales

- La tecnología utilizada para el despliegue debe fundamentarse en soluciones de virtualización [3].
- Toda la infraestructura se desarrollará sobre Linux.
- La tecnología utilizada para la orquestación de servidores es Ansible.
- Los servidores desplegados se monitorizan con la herramienta Zabbix, por ser la existente en la empresa objetivo [4].
- La tecnología utilizada para el desarrollo de la aplicación web es Node.js [5] y Vue.js [6].
- La interfaz web debe ser intuitiva.
- Las contraseñas de todos los servicios y servidores deben de ser seguras. Estas serán formadas por mayúsculas, minúsculas, caracteres especiales y números.
- La aplicación web debe de mostrar mensajes de información/error para guiar al usuario.
- Los servidores Tomcat [7] siempre deben estar disponibles.

3.4. Restricciones técnicas

Debido a que no se podía trabajar directamente sobre la infraestructura de la empresa SIA, el proyecto ha sido desarrollado y desplegado en un solo servidor, sobre el que se simula la infraestructura real, y que dispone de las siguientes características:

- 500GB de memoria ROM.
- 8GB de memoria RAM.
- Procesador Intel i5-7400 de 3.00 GHz.

4 DESARROLLO

Como se ha indicado anteriormente, el problema que se quiere solucionar es disminuir el tiempo en la ejecución de tareas para todos los servidores mediante la aplicación desarrollada. Por otro lado, se quiere monitorizar todos los servidores de la arquitectura mediante gráficos y alertas que indiquen cuando algún servicio no está funcionando correctamente.

En este apartado se explica cómo se ha desarrollado el proyecto y finalmente se mencionan los problemas que se han presentado.

4.1. Despliegue de servidores tomcat

La aplicación debe ejecutarse sobre una infraestructura similar a la de la empresa SIA, por lo que se ha desplegado el entorno que se puede observar en la Figura 1. Por un lado, tenemos dos servidores tomcats que se han desplegado con Docker. Estos servidores son monitorizados por la herramienta Zabbix y, por otro lado, tenemos la aplicación web, que orquestará los servidores bajo el software de Ansible, el cual se conectará a los servidores tomcat mediante el servicio ssh y ejecutará las instrucciones de forma remota. Todo este despliegue se ha realizado sobre un único servidor, en este caso la máquina local.

Existen varias opciones para construir un entorno similar al utilizado en la empresa SIA. En este caso, las opciones valoradas han sido Kubernetes [8], Docker o máquinas virtuales. En un principio la idea principal del proyecto era poder desplegar el entorno con Kubernetes mediante el software MiniKube [9], sin embargo, al realizar los despliegues han surgido limitaciones que no estaban contempladas en un inicio. Por ello, se ha terminado optando por utilizar Docker y máquinas virtuales. Docker nos ofrece la opción

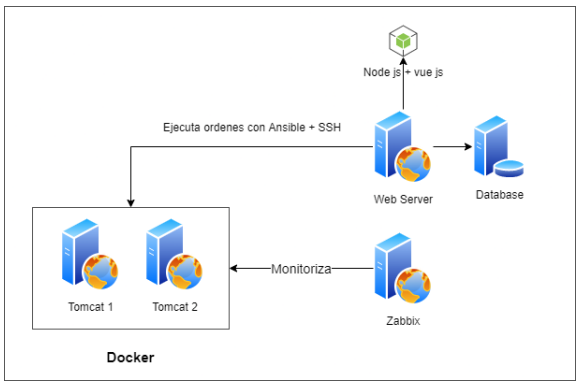

Fig. 1: Infraestructura desplegada

de generar nuestra propia imagen [10]. Para este proyecto se ha utilizado una imagen basada en Tomcat obtenida del repositorio oficial, concretamente la imagen tomcat:8.5-jre8-temurin-focal y ha sido modificada añadiendo algunos paquetes de software necesarios como, por ejemplo, ssh y systemctl. Otra opción que nos permite Docker es la de realizar un fichero de configuración llamado docker-compose.yaml [11], donde se especifican los contendores y servicios que se despliegan con sus correspondientes configuraciones. En este caso, las configuraciones han sido las siguientes:

- En primer lugar, se especifican dos contenedores con los nombres tomcat1 y tomcat2. Para cada uno de ellos se utilizan dos volúmenes. El primer volumen incluye un script que realiza una configuración inicial en cada servidor. Esta configuración únicamente proporciona los permisos necesarios al archivo compartido mediante el segundo volumen e inicia el servidor tomcat. El segundo volumen contiene las claves públicas ssh de la máquina local. Por otro lado, se indica que al iniciar los contenedores se debe ejecutar el script que realiza la configuración inicial y, por último, se indica que red utilizará cada contenedor.

- En segundo lugar, se expone la configuración de la red que utilizan los contenedores. Para realizar esta configuración antes es necesario saber cómo funciona el networking en Docker [12]. En Docker hay diferentes tipos de redes. Por defecto, los contenedores utilizan el tipo bridge, el cual genera una red interna del rango 172.16.0.0/16 y todos los contenedores se pueden ver a través de esa red. Por otro lado, existe el tipo de red local, el cual, como su propio nombre indica, realiza un mapeo de los puertos que utilizan los contenedores con la máquina local. Otro tipo de red es el tipo none, el cual aísla un contenedor y no permite conexiones contra él. Finalmente, existe el tipo de red macvlan que permite crear una subred y asignar una MAC diferente para cada contenedor lo que generará una IP distinta para cada servidor tomcat. Este último ha sido el utilizado para realizar la configuración de red. En este caso, se ha creado la subred 192.168.1.0/24 asignando la dirección ip 192.168.1.1 para el gateway y las direcciones ip 192.168.1.2 y 192.168.1.3 para los contenedores tomcat1 y tomcat2 respectivamente.

Lo ideal utilizando Docker y el tipo de red macvlan sería poder acceder desde el servidor local a los contenedores de Docker, sin embargo, macvlan no permite que el host anfitrión de Docker acceda a los contenedores, por lo que se ha de acceder desde otra máquina. Llegados a este punto se podrían desplegar los contenedores en una máquina virtual o desde otra máquina adicional.

La decisión final ha sido desplegar los contenedores sobre una máquina virtual, obteniendo así el esquema que se muestra en la Figura 2.

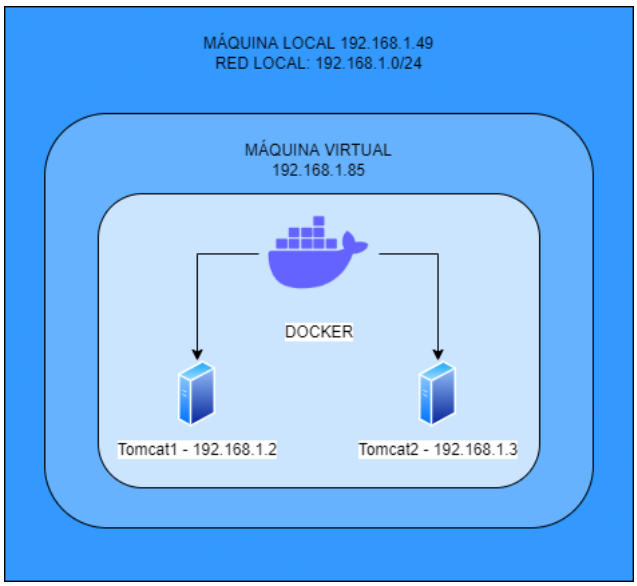

Fig. 2: Despliegue de los servidores tomcats

Gracias a las configuraciones explicadas, desde la máquina local es posible acceder a los servidores tomcat mediante los puertos 8080 o utilizando el puerto 22 del servicio ssh.

4.2. Despliegue de Zabbix

Tal como se indica en el apartado de objetivos, uno de ellos consiste en monitorizar los servidores desplegados con Docker. Nos encontramos con diferentes alternativas para realizar esta monitorización como, por ejemplo, Pandora FMS, Nagios, Zabbix, Monit, etc. En este caso la opción implementada ha sido Zabbix debido a que la empresa SIA utiliza este software para monitorizar sus servidores.

Se ha instalado Zabbix en la máquina local y se ha instalado el agente de Zabbix en los servidores tomcat para permitir así su monitorización. La monitorización se realiza sobre los servidores tomcat y se monitoriza el % de CPU que están consumiendo, el tráfico que pasa por cada uno de ellos, la memoria Heap que están utilizando y finalmente el estado de tomcat y del servicio ssh ya que es necesario para conectar mediante Ansible.

Para cada ítem de monitorización se ha creado un trigger, el cual alerta al administrador del sistema si algo no está funcionando correctamente y finalmente se ha reflejado la monitorización en gráficos.

4.3. Orquestación mediante Ansible

La complejidad que supone la gestión de una gran cantidad de servidores y aplicaciones lleva a la mayoría de las infraestructuras de medianas y grandes empresas a trabajar con Dockers en sus sistemas. Los sistemas de este tipo de

empresas pueden resultar complejos, por lo que gestionar todos los factores que intervienen puede ocasionar errores debido a la necesidad de combinar diferentes tareas. Es en este punto en el que se hace necesaria la orquestación de los servidores. Implementar procesos de orquestación supone varios beneficios: permite la agilización de las tareas, ya que con estos procesos las tareas más rutinarias serán automatizadas. Adicionalmente se reducen los posibles errores que se podían ocasionar, ya que, por un lado, dejan de ser manuales y pasan a ser automatizadas por un software y, por otro lado, ya no se realiza únicamente una tarea por vez, sino que se realizan varias: Se mejora así la eficiencia al posibilitar que los trabajadores inviertan un mayor tiempo en tareas que resulten más beneficiosas para la empresa. En resumen, la orquestación busca una eficiencia general de la empresa, proporcionando una mayor productividad y una mejor gestión del tiempo. Todo ello permite mejorar la organización del equipo de trabajo, generando así una armonía global. Se busca, en definitiva, optimizar el funcionamiento de los servicios para conseguir mayor eficiencia operativa.

4.3.1. Funcionamiento de Ansible

Ansible utiliza una arquitectura de servidor-cliente [13]. Los nodos se conectan entre ellos gracias al protocolo ssh y ejecutan módulos con Python, lo que hace imprescindible su instalación en cada uno de los nodos. Cada vez que se realiza una conexión remota a una máquina mediante ssh hay que introducir la contraseña de la máquina remota, por lo tanto, cada vez que se ejecute Ansible se debería introducir la contraseña de cada servidor. Esto puede ser una tarea pesada ya que cuanto mayor es el número de servidores a orquestar, mayor es el número de contraseñas que se deben introducir. La solución a este problema es añadir previamente la clave pública del servidor en los nodos clientes consiguiendo así que Ansible no requiera la introducción de la contraseña cada vez que sea ejecutado. La clave pública del servidor se debe de copiar en el fichero authorized_keys que se encuentra en el directorio /root/.ssh/ de cada servidor.

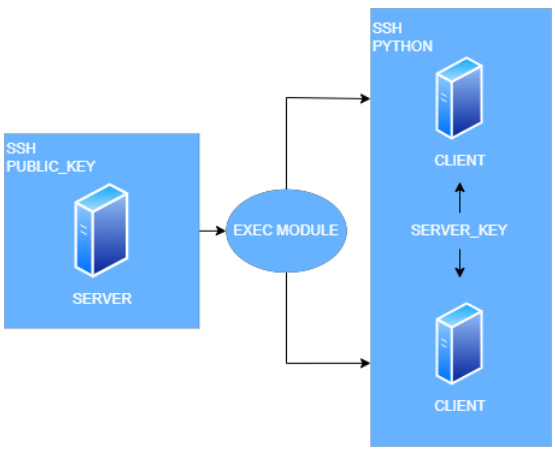

Fig. 3: Arquitectura de Ansible

4.3.2. Módulo de Ansible

Ansible ejecuta tareas. Estas tareas utilizan módulos y todos ellos utilizan Python. Existe una infinidad de módulos y todos ellos se pueden encontrar en la documentación oficial. Para el desarrollo de este proyecto se han utilizado los módulos siguientes:

- Apt: permite instalar un paquete de software.
- Service: permite cambiar el estado de un servicio, entre sus opciones se encuentran started, stopped, restarted y reloaded.
- Copy: permite copiar ficheros.
- Shell: permite ejecutar comandos de Shell.
- Template: permite diseñar una plantilla con variables.

4.3.3. Hosts en Ansible

Ansible se puede ejecutar de varias formas. La primera forma simplemente consiste en indicar el módulo que se desea ejecutar y el host sobre el que se ejecutará. Por ejemplo, la ejecución de ping se realiza de la siguiente manera: ansible 192.168.1.2 -m ping.

La segunda forma consiste en ejecutar un módulo para un conjunto de servidores y es aquí donde entra en juego la configuración del fichero hosts. Este fichero se encuentra en el directorio /etc/ansible y almacena los hosts con los que se quiere interactuar mediante Ansible. El fichero hosts se puede definir utilizando dos formatos: por un lado, tenemos el texto plano, y por el otro, el formato utilizado es el YAML. Este fichero es capaz de definir los hosts, organizarlos por grupos y generarlos por rangos. En la Figura 4 se puede observar cómo se han definido los grupos tomcats, webservers y webservers2. Mientras que en los grupos tomcats y webservers se han definido manualmente uno por uno, el grupo webservers2 ha sigo generado automáticamente incluyendo un rango del número 1 al 50.

```
192.168.1.49
192.168.1.50

[tomcats]
10.52.10.141
10.52.10.142
10.52.10.143
10.52.10.144

[webservers]
server1.sia.com
server2.sia.com

[webservers2]
server[01:50].sia.com
```

```
all:
  hosts:
    mail.example.com:
  children:
    webservers:
      hosts:
        foo.example.com:
        bar.example.com:
    dbservers:
      hosts:
        one.example.com:
        two.example.com:
        three.example.com:
```

Fig. 4: Ejemplo de fichero host en texto plano y yaml, respectivamente

Una vez definido el fichero hosts, este se puede utilizar de la siguiente manera:

- ansible all -m ping: ejecutará el módulo ping para todos los hosts indicados en el fichero.

- ansible tomcats -m ping: ejecutará el módulo ping únicamente para los hosts que formen parte del grupo tomcats.

Si surge la necesidad de disponer de más de un fichero de hosts, este se puede indicar con la opción -i de la siguiente manera: ansible -i <archivo_hosts> -m ping.

En este caso, únicamente disponemos de dos servidores tomcats, por lo tanto, el fichero hosts sólo contendrá dos registros.

4.3.4. Tareas y Playbooks

Igual que para los hosts, Ansible puede ejecutar una sola tarea o un conjunto. La ejecución de una sola tarea la hemos visto en los ejemplos anteriores, concretamente en los ejemplos de la ejecución de ping. A continuación se expone cuál es el funcionamiento que utiliza Ansible para ejecutar un conjunto de tareas. Ansible utiliza unos ficheros nombrados playbooks. En estos ficheros se pueden definir varias tareas a realizar, lo que hace posible desplegar complejas configuraciones para un conjunto de hosts. El formato utilizado por los ficheros playbooks es YAML [14]. YAML permite programar con un formato muy legible por los humanos y está inspirado en lenguajes como XML o Python. En la Figura 5 se puede observar un ejemplo de Playbook.

```
---

- hosts: tomcats
  tasks:
    - name: Instala Zabbix
      apt: name=zabbix-agent state=present
    - name: Configura Zabbix
      copy:
        src: zabbix_agentd.conf
        dest: /etc/zabbix/zabbix_agentd.conf
      notify:
        - "Reinicia Zabbix"

  handlers:
    - name: Reinicia Zabbix
      service: name=zabbix-agent state=restarted
```

Fig. 5: Playbook que instala y configura Zabbix

Los ficheros Playbook utilizan tres tipos de estructura de datos: por un lado, tenemos atributos simples, como por ejemplo el atributo “hosts”, “become” o “user” entre muchos otros. Estos sirven para definir configuraciones sobre las tareas que se van a ejecutar. Otro tipo de dato son las “tasks”, que están formadas por módulos y cada módulo utiliza un conjunto de atributos. Como su propio nombre indica, estas son las tareas que se ejecutarán. Y finalmente nos encontramos con los “handlers” que son iguales que las “tasks” pero se ejecutan tras la ejecución de una “task”. En la Figura 5 se observa cómo se realiza la instalación y configuración del servidor Zabbix a través de un fichero Playbook. Primeramente se define el atributo ”hosts”, en este caso las tareas se realizarán sobre el grupo de máquinas pertenecientes al grupo tomcats. Como recordatorio, el grupo tomcats se encuentra en el fichero /etc/ansible/hosts explicado en la sección anterior de este mismo artículo. Seguidamente se definen las tareas, en este caso existen dos, Instala Zabbix y Configura Zabbix. Estas utilizan los módulos apt y copy, respectivamente.

Finalmente, mediante el atributo notify se ejecutará el handler Reinicia Zabbix.

Cuando ejecutamos órdenes mediante Ansible podemos obtener diferentes tipos de respuestas. Suponiendo que se desea instalar un paquete de software, pueden surgir las siguientes opciones:

- El paquete de software se ha instalado correctamente, por lo que ha habido un cambio en el sistema operativo y la respuesta obtenida por Ansible es changed.
- El paquete de software no se ha instalado debido a que ya se encuentra instalado, por lo tanto, no ha habido ningún cambio en el sistema operativo y Ansible muestra el mensaje ok.
- El paquete de software no se ha instalado a causa de un error. Ansible muestra el mensaje fail.

```
PLAY RECAP *************************************
192.168.1.2                : ok=2    changed=1
192.168.1.3                : ok=2    changed=1
```

Fig. 6: Respuesta obtenida por Ansible

Una vez comprendido el funcionamiento básico de Ansible, se desarrolla una aplicación que utiliza este software y permite al usuario desplegar configuraciones básicas o complejas a través de una interfaz web.

4.4. Aplicación web

En la actualidad existe una gran variedad de tecnologías que permiten desarrollar una aplicación web. Las opciones contempladas para este proyecto eran Java o JavaScript, utilizando sus frameworks Spring Boot o Node.js respectivamente [15]. Node.js es un lenguaje de programación que utiliza un modelo de E/S sin bloqueo controlado por eventos y utiliza un solo thread, esto lo hace ligero y muy eficiente. Además, su comunidad está en constante crecimiento, al igual que el gestor de paquetes que utiliza, npm. Por otro lado, Spring Boot es un framework que es muy fácil de ejecutar, lo que evita complicaciones y permite comenzar a utilizar una aplicación productiva en poco tiempo. Este tiene toda la gran comunidad de Java tras él, soporta múltiples threads y, a diferencia de Node.js, es tipado. Valorando estas dos opciones, se ha optado por desarrollar la aplicación con JavaScript, en este caso Node.js para el backend y Vue.js para el frontend. Además de las tecnologías mencionadas anteriormente, se han utilizado las tecnologías adicionales HTML/CSS, Bootstrap, MySQL y para hacer posible la orquestación de los servidores se utiliza el software Ansible.

La aplicación web no requiere de una base de datos muy compleja, dado que únicamente almacena los usuarios que tienen acceso a la administración de los servidores Tomcat y las direcciones de dichos servidores. Por lo tanto, se podría utilizar una base de datos no relacional, pero se ha optado por utilizar MySQL debido a que el servidor Zabbix ya utiliza este tipo de base de datos y se puede aprovechar sin la necesidad de utilizar un nuevo servicio.

Una vez se han identificado las tecnologías utilizadas para el desarrollo de la aplicación, se procede a la explicación de los módulos que la forman.

La aplicación dispone de un panel de login, al que solo tiene acceso el administrador de la plataforma. Una vez iniciada la sesión, la aplicación muestra un dashboard, el cual se divide en cuatro componentes. Primeramente, el panel de los hosts. Este panel tiene como funcionalidad añadir y eliminar hosts a la plataforma para ser orquestados. Su incorporación a la plataforma es sencilla y se realiza a través de un formulario en el que hay que indicar el nombre del host y su dirección ip. Una vez añadido, este se guarda en la base de datos y a su vez se agrega al fichero hosts que posteriormente utilizará Ansible. De la misma forma, cuando el host se elimina de la plataforma, este se elimina de la base de datos y del fichero hosts.

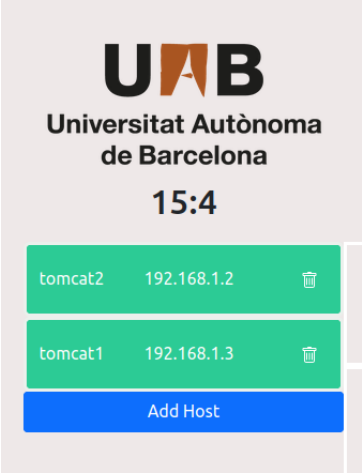

Fig. 7: Panel de hosts

Además del panel de hosts, el resto del dashboard está compuesto por tres componentes que utilizan Ansible. Estos componentes generan automáticamente un fichero programado en Ansible, el fichero es parecido al mostrado en la Figura 5, pero este irá variando según el módulo que se ejecute. Una vez generado, la aplicación web utilizará el fichero hosts generado mediante el componente panel de hosts y ejecutará las instrucciones sobre los hosts que estén dados de alta en la plataforma.

El primer componente permite al usuario realizar instalaciones de software. Únicamente requiere de introducir el nombre del paquete que se quiere instalar y lanzar la función clicando en su botón correspondiente. Este componente utiliza el módulo apt de Ansible que, como se ha mencionado previamente, sirve para instalar paquetes de software. Para utilizar el módulo apt tenemos que indicarle el nombre del paquete que queremos instalar y indicarle que el estado del paquete sea presente en el sistema. Lo que Ansible realiza a través de este módulo es buscar en los repositorios y en el sistema operativo el paquete indicado, y además comprueba el estado en el que debe estar una vez finalizada su ejecución. Por lo tanto, lo que conseguimos indicándole a Ansible que el estado del paquete debe de ser presente es que cuando termine la instalación el paquete esté instalado.

El segundo componente es muy parecido al anterior, con la particularidad de que su funcionalidad no es instalar, su funcionalidad es cambiar el estado actual de los servicios del sistema. En este caso, al igual que en el anterior, hay que introducir el nombre del servicio que se desea gestionar y mediante un desplegable se indica el estado al que se quiere cambiar. Los estados son started, stopped, restarted y reloaded. Igual que el primer componente, este genera un archivo de Ansible pero utilizando el módulo ”service”.

Finalmente el tercer componente permite al usuario realizar tareas de forma personalizada. El usuario debe de tener conocimientos en Ansible y debe de programar las tareas en un cuadro de texto. Debido a que este módulo permite realizar configuraciones más avanzadas, se ha incorporado la funcionalidad de poder subir ficheros al servidor y, de esta forma, el usuario también puede distribuir ficheros de configuración en todos los servidores.

4.5. Problemas

A lo largo del proyecto han surgido algunos problemas. La mayoría se han solventado sin grandes dificultades (problemas de programación web, instalación y configuración de paquetes, etcétera), sin embargo, destaca el problema surgido en el despliegue de los servidores tomcats. Como se indica en el apartado correspondiente a esta fase, el despliegue de los servidores se planteaba realizarlo mediante Kubernetes.

El problema surge al realizar el despliegue y observar que solo era posible la creación de un nodo debido a que el software de MiniKube sólo permite desplegar un nodo en la máquina local. Para comprender mejor el problema hay que entender la arquitectura que utiliza Kubernetes.

Kubernetes funciona con nodos, pods y deployments entre otros elementos. Un nodo es una máquina de trabajo virtual o física, y Kubernetes le asigna una IP interna y una IP externa. Además, un nodo puede contener uno o más pods para que ejecuten sus servicios. Un pod es una instancia de una aplicación y está asignado a un contenedor. Y un deployment es el despliegue de un número de réplicas para un mismo pod.

Considerando que un nodo puede contener uno o más pods, esta limitación no parece un problema, sin embargo, con la ayuda de la Figura 8 se detalla la complejidad del problema.

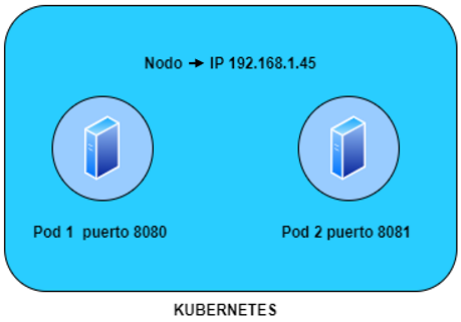

Fig. 8: Problema MiniKube

Se observa que el acceso a los pods sería mediante IP y un puerto diferente para cada pod ya que no podemos utilizar el mismo puerto para ambos. Este inconveniente rompe con el esquema de la empresa, ya que todos los tomcats funcionan con el puerto 8080 y no hay manera de acceder individualmente a cada uno de ellos. Se han probado diferentes formas de desplegar los contenedores y tratar de acceder individualmente a ellos, pero no se ha conseguido hacerlo con Kubernetes. La solución más sencilla que se ha encontrado ha sido utilizar la nube ya que en estos servidores no se

utilizará MiniKube y sí que se podría desplegar más de un nodo, pero para ello era necesario contratar algún servicio adicional que no estaba incluido en el presupuesto inicial del proyecto. Al comprobar que este problema perjudicaría los plazos estimados de la planificación, se tomó la decisión de desplegar el entorno virtual mediante Docker, tal y como se ha descrito en la fase.

5 PLANIFICACIÓN

En este apartado se detalla cuál ha sido la planificación para el desarrollo completo del proyecto. Se explican las fases en las que se ha organizado y en qué consiste cada una de ellas, así como las fechas de entrega que se han establecido.

El proyecto ha tenido una duración de cuatro meses, iniciando la fase de conceptualización en febrero y el desarrollo como tal el día 07/03/2022 y finalizando el día 07/07/2022. Este se ha organizado en cinco fases que se explican a continuación.

- La primera fase es el despliegue de servidores tomcat con Docker. Esta fase tenía como objetivo instalar Docker en el servidor principal y poder desplegar dos servidores tomcats haciendo posible que estos se comunicaran con el servidor Zabbix y la aplicación web.
- La segunda fase consistía en realizar el despliegue del servidor Zabbix. El objetivo de esta fase era instalar el servidor Zabbix en la máquina local e instalar los clientes de Zabbix en las máquinas desplegadas en la fase anterior. Con este servidor se quería poder monitorizar el estado de los tomcats y comprobar que todo estaba funcionando correctamente entre servidores y máquina local.
- La tercera fase hace referencia al despliegue de Ansible. En este punto los servidores tomcats y el servidor Zabbix que se encuentra en la máquina local ya tienen conexión, por lo tanto, el objetivo de esta fase era conseguir realizar tareas simultáneas en varios servidores gracias al software Ansible.
- La cuarta fase consistía en el desarrollo de una aplicación que fuese capaz de orquestar los servidores tomcats ejecutando el software de Ansible a través de una interfaz web.
- Finalmente, en la última fase se ha desarrollado toda la documentación final.

Para el seguimiento y control de las fases explicadas, se ha establecido un total de seis entregas:

- Desarrollo completo del entorno con Docker → 23/03/2022
- Desarrollo, configuración y testeo de todos los servidores mediante Zabbix → 30/03/2022
- Desarrollo de todas las tareas Ansible sobre los servidores Tomcat → 07/04/2022
- Aplicación web con el login y cierre de sesión completado → 03/05/2022
- Aplicación web con las funcionalidades de Ansible → 07/06/2022
- Documentación final → 07/07/2022

El calendario de entregas se ha respetado a lo largo del proyecto, y salvo desviaciones menores, puede decirse que se ha cumplido la planificación prevista.

6 METODOLOGÍA

Para el desarrollo de este proyecto se ha utilizado la metodología DevOps [16]. Esta metodología consiste en la fusión de los departamentos de desarrollo y operaciones. El objetivo de esta metodología es que ambos departamentos trabajen conjuntamente para poder realizar una integración y entrega continua. DevOps es ideal para este proyecto debido a que se está tratando sobre un proyecto donde debe de haber mucha comunicación entre los departamentos que gestionan la infraestructura del cliente y los desarrolladores del software final, ya que no solo se está desarrollando una aplicación web, sino que se está trabajando sobre varios servidores.

En este caso, el departamento de desarrollo y de operaciones ha funcionado de la siguiente manera:

- Desarrollo: A lo largo del proyecto se han realizado grandes cantidades de configuraciones, entre ellas podemos destacar: construir la imagen utilizada para el despliegue de los servidores mediante Docker, la configuración de red para que los servidores puedan tener visibilidad en la infraestructura desplegada, las configuraciones necesarias de Zabbix y la configuración base de la aplicación web. Además, se han completado tareas de puro desarrollo, como por ejemplo, ítems de monitorización en Zabbix, scripts de configuración en bash, la programación de tareas en Ansible o la propia aplicación web.
- Operaciones: Todo el trabajo realizado por el departamento de desarrollo no tiene ningún sentido si no se implementa. Este departamento ha proporcionado todo lo necesario para que el departamento de desarrollo pueda continuar con el trabajo realizado. Ha proporcionado la infraestructura sobre la que trabajar, ha implementado todo lo desarrollado y ha comprobado que todo funcionase correctamente. Se puede decir que el departamento de operaciones ha integrado constantemente las configuraciones y software desarrollado, verificando que todo funcionase según lo previsto y, en caso contrario, notificándolo al departamento de desarrollo.

Para el seguimiento de las tareas que se debían de desarrollar se ha utilizado la herramienta Jira, que permite gestionar de forma sencilla todas las tareas y se puede llevar un control visual de todo el proyecto con los paneles que esta ofrece.

Como versión de control se ha utilizado la herramienta GitHub. Se han creado tres ramas. En la versión 1 podemos encontrar todas las configuraciones desarrolladas pertenecientes a las fases 1 y 2, las cuales hacen referencia al despliegue de los servidores mediante Docker y la instalación de Zabbix. Por otro lado, encontramos la rama nombrada

versión 2 que implementa las mismas configuraciones, pero esta vez realizadas con Ansible. Finalmente se puede encontrar la rama webapp, en la que se almacena el código desarrollado de la aplicación web.

7 Resultados

Una vez desarrollado todo el proyecto, este se ha testeado y se ha comprobado si su funcionamiento es el correcto, así como cuál es el alcance que puede tener. Como se comenta al principio de este artículo, el objetivo de la aplicación era poder realizar configuraciones básicas, instalar software, monitorizar y cambiar el estado de servicios. Adicionalmente, se han implementado algunas configuración más avanzada como distribuir ficheros de configuracion entre todos los servidores. Sin embargo, la potencialidad que nos proporciona Ansible es muy grande y se pueden llegar a realizar configuraciones más complejas. Especificamente se han obtenido como resultado los siguientes puntos:

- Una arquitectura similar a la que dispone la empresa SIA, formada por dos servidores tomcats que han sido desplegados con Docker, un servidor Zabbix y una aplicación web.
- Un servidor Zabbix que monitoriza constantemente los servidores tomcat.
- Se ha aprendido el funcionamiento básico del software Ansible.
- Se ha desarrollado una aplicación web en Node.js y Vue.js
- Se consigue instalar paquetes de software de forma paralela en varios servidores.
- Se consigue cambiar el estado de servicios de forma paralela en varios servidores.
- Se pueden desplegar configuraciones avanzadas de forma paralela en todos los servidores.

8 Conclusión

Este artículo propone una solución para la empresa SIA, que es aplicable a cualquier empresa que disponga de más de un servidor. La solución propuesta aporta beneficios tales como el ahorro de tiempo y un incremento de productividad, fundamentales para cualquier empresa hoy en día. Como se ha comentado en el apartado de la planificación, el proyecto ha tenido una duración de cuatro meses en los que se ha trabajado constantemente y, gracias a la correcta planificación el proyecto se ha llevado a cabo en el tiempo asignado. Al principio del proyecto se establecieron un conjunto de objetivos, que se han completado satisfactoriamente. Puntualizar eso sí, que la idea inicial de utilizar Kubernetes fue reemplazada por el uso de Dockers por las restricciones del despliegue en un entorno limitado.

A nivel personal, creo que es un proyecto completo debido a que se trabaja con una gran variedad de tecnologías desde varios puntos de vista. Por un lado, se ha trabajado con Docker y Ansible, lo que me han permitido enfocar determinadas fases del proyecto desde el punto de vista de un administrador de sistemas. Además, ambas tecnologías tienen un largo camino por delante, por lo que conocer cómo funcionan y haberlas testeado es todo un beneficio para mí.

El proyecto requería también el desarrollo completo de una aplicación web. Esto me ha proporcionado el punto de vista de un desarrollador y qué mejor que utilizar las tecnologías Node.js y Vue.js aprendidas a lo largo de la carrera.

Dicho esto, el proyecto permite su ampliación, con algunas líneas futuras que se esbozan a continuación.

En primer lugar, la aplicación desarrollada está pensada para alguien que sabe utilizar Ansible. Por el momento, solo dispone de los tres módulos explicados anteriormente, pero se podrían desarrollar muchos más, simplificando así su funcionamiento y evitando que el usuario deba de saber utilizar Ansible para realizar configuraciones avanzadas.

Por otro lado, se podría implementar la funcionalidad de monitorización en la misma aplicación web. No era requisito para este proyecto, pues, la empresa SIA ya utiliza Zabbix para esta fución, pero puede ser interesante en otros entornos.

Debido a las limitaciones de hardware, únicamente se ha utilizado Docker para realizar el despliegue de los servidores tomcats, sin embargo, una mejora para el proyecto sería realizar el despliegue de toda la arquitectura utilizando Docker.

Para finalizar este artículo quisiera agradecer, por un lado, a mis profesores Juan Antonio Martínez Carrascal y Sergi Robles Martínez por su dedicación y guía a lo largo de este proyecto y, por otro lado, a la empresa SIA y a mi correspondiente tutor dentro de la empresa, Pablo Andrés Potes, por realizar la propuesta y hacer posible su desarrollo.

Referencias

[1] ”NVD - CVE-2021-44228”, NVD, 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-44228.

[2] Red Hat Ansible Automation Platform on Microsoft Azure”, Red Hat, 2022. [Online]. Available: https://www.ansible.com/.

[3] ”Developers Love Docker. Businesses Trust It.”, James Ratliff, 2022. [Online]. Available: https://www.docker.com/.

[4] ”TAKE YOUR BUSINESS SERVICE MONITORING TO THE NEXT LEVEL.”, Zabbix, 2022. [Online]. Available: https://www.zabbix.com/.

[5] .ªcerca de Node.js.”, Open JS Foundation, 2022. [Online]. Available: https://nodejs.org/es/about/.

[6] ”The Progressive JavaScript Framework.”, Evan You, 2022. [Online]. Available: https://vuejs.org/.

[7] .ªpache Tomcat.”, Apache Software Foundation, 2022. [Online]. Available: https://tomcat.apache.org/.

[8] ”¿Qué es Kubernetes?”, Kubernetes, 2022. [Online]. Available: https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/

[9] Çómo empezar con Kubernetes usando Minikube", Marco A. Muñiz Ochoa, 2022. [Online]. Available: https://cloudnative.mx/como-empezar-con-kubernetes-usando-minikube/

[10] "Docker Build", Docker INC, 2022. [Online]. Available: https://docs.docker.com/engine/reference /commandline/build/

[11] "Docker Compose", Docker INC, 2022. [Online]. Available: https://docs.docker.com/compose/

[12] "Networking overview", Docker INC, 2022. [Online]. Available: https://docs.docker.com/network/

[13] .ªnsible: Up and Running: Automating configuration management and deployment the easy way.", Hochstein, Lorin, and Rene Moser. .ºReilly Media, Inc.", 2017.

[14] "¿Qué es YAML?", Red Hat, 2022. [Online]. Available: https://www.redhat.com/es/topics/automation/what-is-yaml

[15] "Desarrollo backend para aplicaciones web, servicios web restful: Node. js vs spring boot.", Haro, Edward, 2019.

[16] "DevOps.",C. Ebert, G. Gallardo, J. Hernantes and N. Serrano, 2016.