UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Automatización del despliegue en la nube de sistemas de formación computacional

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Moreno González, Alejandro

Tutor/a: Moltó Martínez, Germán

CURSO ACADÉMICO: 2021/2022

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Automatización del despliegue en la nube de sistemas de formación computacional

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Alejandro Moreno González

Tutor: Germán Moltó Martínez

2021 - 2022

Resumen

Un problema común en empresas que trabajan con las *TIC* (Tecnologías de Información y Comunicación), es proporcionar los recursos necesarios para la formación en un campo, ya sea inteligencia artificial, criptografía, etc., a un nuevo empleado.

Este proyecto pretende automatizar el despliegue de sistemas en el proveedor de Cloud público *Amazon Web Services* (AWS)[1] para la formación, en este caso, en tecnología *Big Data y Data Quality*[2], si bien se podría aplicar un procedimiento similar para la formación en otras áreas.

Para ello primero se realizará un estudio de los requisitos de la aplicación que vamos a utilizar, de forma que se despliegue una máquina lo más eficientemente posible. También que el entorno se genere dinámicamente a partir del identificador de usuario que lo vaya a utilizar, ya pueda ser su nombre y apellidos, u otro parámetro, así como definir unas competencias de aprendizaje, con las utilidades necesarias, para que la persona que vaya acceder a este sistema pueda focalizarse en aprender los conocimientos y adquirir la experiencia para su puesto de trabajo, y no tener que configurar o cambiar parámetros del ordenador para desempeñar sus tareas.

Se plantea el uso de despliegue automatizado de las máquinas virtuales usando Ansible, además apoyado de scripts *SQL*[3] para la definición de las bases de datos que serán utilizadas y de configuración de la aplicación.

Palabras clave: *Big Data, formación, despliegue, sistemas.*

[1] *AB2* es una plataforma de despliegue de máquinas en la nube.
[2] *Biga Data y Data Quality* son tecnologías enfocadas en la gestión masiva de datos, y el enriquecimiento de la información aportada por el mismo dato
[3] *SQL* es un lenguaje que te permite lanzar consultas y modificaciones sobre las bases de datos, piedra angular de el producto que se quiere utilizar

Tabla de contenidos

Índice de figuras

1. Introducción

Este proyecto está respaldado por *Infoverity*[4], es una consultora que ofrece soluciones en productos relacionados con su partner *Informatica*[5].

Informatica se encarga del desarrollo de los productos, la gestión de versiones, y *debugging*, e *Infoverity* del desarrollo de soluciones, instalación, configuración y soporte en proyectos para otras empresas.

El software sobre el que colaboran ambas empresas se focaliza en la gestión de datos masivos, y enriquecimiento del dato.

En un primer momento se planteó este proyecto como una forma de automatizar la instalación de la aplicación en clientes de forma que se agilizaría esta tarea, y con los recursos disponibles parecía alcanzable, pero se descartó dado los riesgos que esto conllevaba.

Una instalación automatizada en un sistema cliente, sin la supervisión de un técnico que pueda realizar las comprobaciones necesarias no genera confianza, además de la gran variedad de sistemas operativos, plataformas, aplicaciones, pasos previos, y herramientas diversas que puede haber en cada equipo genera situaciones muy diferentes.

De forma que el proyecto se simplificó y se decidió llevarlo a cabo a nivel interno, dentro de *Infoverity,* ya que la demanda de técnicos formados en la herramienta **Informatica MDM**[6] crece cada año, para participar en proyectos dando soporte a otras empresas, resulta necesario tener una forma de crear entornos de pruebas y de formación rápidamente.

La idea es generar unos entornos virtuales en una plataforma cloud, y automatizar el despliegue de los mismos entornos, identificando cada sistema según el usuario que se vaya a formar.

[4] Infoverity: https://www.infoverity.com/
[5] Informatica: https://www.informatica.com/
[6] Informatica MDM (*Master Data Management*) es el producto que se quiere desplegar, es una aplicación de desarrollo de soluciones para la gestión de datos multidominio.

1.1 Tecnologías

Máquinas virtuales[7]:

Una máquina virtual [1] es una aplicación que emula una computadora física, con sus especificaciones de Hardware como la memoria, la CPU… y que también puede ejecutar un Sistema Operativo distinto al del host que lanza esta aplicación, simulando también la Interfaz de Usuario, etc.

AWS y AMI:

AWS [8]*(Amazon Web Services)* es una subsidiaria de Amazon, una plataforma que ofrece pagar por el uso que hagas de las máquinas desplegadas en la misma, de forma que si en cierto momento no se realiza tanto uso de los sistemas desplegados en esta plataforma, no tienes por qué pagar tanto por su mantenimiento.

Esta plataforma es muy interesante, porque permite desplegar sistemas con muchas características o aplicaciones ya instaladas, como el sistema operativo, un software de bases de datos, ciertos usuarios, y demás. Además que permite crear máquinas virtuales propias a esa plataforma, una vez se convierta al formato AMI.

AMI [9]*(Amazon Machine Image)* es una imagen de una máquina virtual en un formato compatible con AWS, contiene la información del software como las aplicaciones, y las especificaciones hardware.

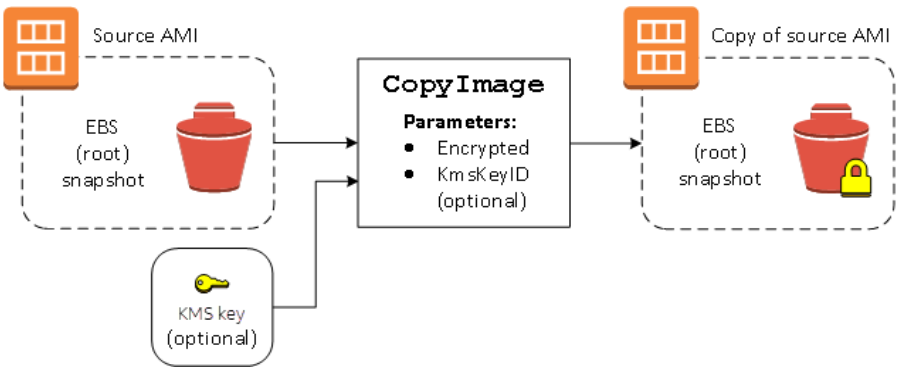

Fig.1 Proceso de clonación de una AMI en AWS [2]

[7] Las máquinas virtuales se pueden emular en local con aplicaciones como VMware Workstation o VirtualBox
[8] AWS tiene muchos servicios de despliegue de sistemas en todo el mundo y multitud de herramientas con las que mejorar el servicio
[9] AMI existen AMI con todo tipo de SO instalados, y bases de datos, listas para su ejecución

EBS[10]:

Elastic Block Storage es una tecnología que permite crear discos virtuales, volúmenes en los que volcar unos datos o un contenido, de forma que se pueden duplicar o añadir a otras instancias de *AWS*.

Un caso de uso sería tener una base de datos en un *EBS* y que fuera rotando entre diferentes instancias adquiriendo los nuevos datos que fueran produciendo las mismas.

IaC[11]:

Infraestructura como Código, es un tipo de herramienta que sirve para desarrollar scripts que describen no solo el comportamiento de la propia aplicación, sino también de la propia infraestructura de donde se ejecute el software desarrollado.

De forma que te permite, a través de código, instalar aplicaciones, configurar sistemas…

Ejemplos de estas herramientas son Ansible, Terraform, o CloudFormation:

- **Terraform**[12] Es preferible en entornos que requieran ciclos de mantenimiento, muy útil para construir en base a versiones y monitorización de entornos.
- **CloudFormation**[13] Es una herramienta exclusiva para sistemas en AWS, muy potente pero no es posible reutilizar el código para otras plataformas.
- **Ansible**[14] Es una aplicación especializada en el despliegue de aplicaciones y configuración de entornos.

[10] EBS Es una tecnología de aprovisionamiento de volúmenes de datos
[11] IaC Sirve para describir procesos en el sistema a través de código
[12] Terraform Requiere que la máquina sobre la que se quieren realizar cambios, tenga una interfaz de comunicación que entienda las órdenes que se ejecutan desde Terraform sobre ese sistema
[13] CloudFormation Tiene también disponible muchas plantillas para agilizar su desarrollo
[14] Ansible Desarrollado por Red Hat, **permite realizar todo tipo de cambios en sistemas Unix**

1.2 Resolución del problema

Tras analizar las tecnologías relacionadas con el problema descrito, se encuentran las siguientes resoluciones al problema.

Clonación de máquinas virtuales:

Una de las cualidades de las máquinas virtuales es la capacidad de clonarse, de forma que puedes tener 2 máquinas exactamente iguales, de forma que una primera solución podría ser generar una máquina virtual con **Informatica MDM** ya instalado, e ir generando copias de esta, o subirla a alguna plataforma.

De esta forma los nuevos empleados tendrán un acceso inmediato y eficiente para el mismo entorno, todas las máquinas para todos los empleados serían la misma.

Esto también tiene sus desventajas, ya que pueden haber empleados que no requieran la misma formación y este entorno sea de utilidad, o que haya que realizar cambios en el entorno y **no se puedan propagar actualizaciones de una manera sencilla**, en las máquinas ya clonadas.

Crearse una AMI en AWS :

Otra posibilidad sería crear una AMI en AWS, con una base de datos instalada, e Informatica MDM de forma que cada empleado tendrá una copia de la instalación exactamente igual, y en caso de que se actualice la versión del producto sería actualizar esas AMI que estén desplegadas.

Usar tecnología IaC:

La opción más interesante sería desarrollar un código que describa cómo se configura un entorno de distribución Unix para que se pueda ejecutar Informatica MDM, de forma que se tendría que definir cómo se instala todos los componentes necesarios en el sistema:

- Java 8
- Un SGBD compatible
- Informatica MDM

La ventaja de usar esta tecnología, es que permite generar un código, que se puede reutilizar partes del mismo, en caso de que se quiera reutilizar el trozo del script que sirve para instalar Java 8 u otro componente.

Otras ventaja es que no será necesario estar usando máquinas exclusivamente en AWS, pudiendo desplegar estas máquinas en plataforma que permitan el despliegue de Sistemas Unix

Y usando esta tecnología, es posible que este proyecto se pueda avanzar y llegar a usar para proyectos de instalación en sistemas de otras empresas, aunque esto queda muy grande todavía.

La principal desventaja es el costo de tiempo, el desarrollo y pruebas con herramientas IaC.

1.3 Objetivos

Los objetivos finales para que el proyecto esté completado son:

- Usar una aplicación de IaC, de forma que en el sistema esté instalado una versión del producto *Informatica MDM* totalmente funcional, con la capacidad de realizar cambios en la configuración, lanzar a ejecución nuevos trabajos sobre la misma, y poder explorar la herramienta con total libertad, para que el usuario que se conecte pueda aprender a usar el producto.
- El producto deberá ser posible instalarlo desde un sistema vacío, en el que solo se encuentre el SO instalado, sin ningún otro componente.
- La aplicación será escalable y susceptible a nuevos cambios en el futuro, de forma que si se quieren realizar cambios, o añadir funcionalidades, sea de la manera más simplificada posible.
- Parametrizar la instalación de *Informatica MDM*, de forma que cambiando algunas variables de Ansible se pueda ejecutar una instalación personalizada para cada usuario.
- Que el entorno sea seguro. Esto se consigue generando los mínimos usuarios para los servicios, y otorgando solo los permisos necesarios a esos usuarios.

2. Ansible

Dado que Terraform y CloudFormation están más especializados en la monitorización del sistema, y en la gestión de la memoria, la congestión de red, … y siendo que Ansible es muy potente en el despliegue de aplicaciones, configuración de entornos, y automatización de tareas, se ha considerado que es la mejor herramienta para llevar a cabo los objetivos del proyecto.

Wikipedia: "*Ansible is an open-source software provisioning, configuration management, and application-deployment tool enabling infrastructure as code. It runs on many Unix-like systems, and can configure both Unix-like systems as well as Microsoft Windows.*"

Esta herramienta permite lanzar tareas a un sistema, emulando que se ha conectado un usuario con unas credenciales a través de *SSH* [15]y ejecuta unas órdenes previamente escritas en un script que desarrollamos.

Un ejemplo de tarea sería la siguiente:

```
- name: Dar todos los permisos al fichero /work
  ansible.builtin.file[3]:
      path: /work
      owner: root
      group: root
      mode: '777'
```

Durante el proyecto se verán muchos más módulos y funcionalidades que habilita Ansible y su potencial, Ansible ejecuta tareas, emulando que se conecta a un equipo y que lanza órdenes en línea de comandos. Tareas que se describen previamente en un fichero "main.yml", que sirve para almacenar las sentencias.

Entonces para poder realizar la tarea es necesario un equipo que tenga Ansible instalado, que será el que ejecute todo el proceso, y otro sistema destino que tenga habilitadas las conexiones *SSH* desde nuestra máquina, y que haya una conexión de red entre ambas.

[15] SSH (Secure Shell) es un protocolo de comunicación segura, genera un canal encriptado entre el cliente y el servidor para el envío de mensajes y ficheros.

2.1 Roles y finalidad de Ansible

La finalidad de Ansible es automatizar las tareas de toda la instalación de una aplicación de desarrollo de software, una aplicación que contiene muchos componentes muy complejos.

Además, a través de esta herramienta es posible desplegar en muchos equipos a la vez, y usar las herramientas que ofrece para parametrizar la instalación, hacerla más genérica, realizar instalaciones personalizadas para cada usuario, y cambiar los roles o añadir a los ya existentes.

Es necesario antes de realizar cualquier automatización en esta herramienta, llevar a cabo la tarea de forma manual, y posteriormente realizar las comprobaciones necesarias para saber que el *playbook* ha alcanzado el resultado esperado.

Primero habrá que hacer un trabajo previo en los sistemas, instalando y analizando el estado final del sistema, para saber como se tiene que efectuar la automatización.

Anteriormente se ha mencionado que en Ansible existe una abstracción, similar a la estructura de programación **Orientada a Objetos** (OO). Una de las características que mejor define esta sintaxis, es la capacidad de dividir un problema en distintos actores o acciones, de forma que si el programa falla en alguna función en concreto, es más fácil saber en qué parte está focalizado el error.

Es muy importante esto porque mejora mucho el proceso de *debugging,* que se vuelve mucho más sencillo, además de realizar modificaciones en el código o sustituir bloques, módulos o funciones por otros más depurados.

Además, igual que en otros lenguajes de desarrollo como Java, se puede definir en un proyecto distintos alcances para las variables, aportando cierta versatilidad al desarrollo.

En Ansible es necesario definir un script, denominado *playbook,* que lanzará las tareas que tenga codificadas, ofreciendo muchas librerías, que aportan muchas funciones, y herramientas.

Para la creación de un rol en Ansible hay que ejecutar el comando ***ansible-galaxy init <nombre_del_rol>*** en el espacio de trabajo que queramos que se genera la estructura de fichero, podría asemejarse al crear un proyecto nuevo con Eclipse.

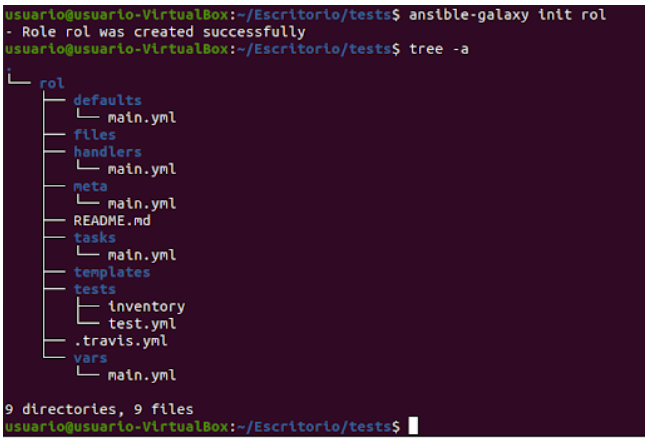

Fig.2 Estructura de un rol/proyecto por defecto

Lo que se genera (Fig.3) es una estructura de carpetas para almacenar la documentación del proyecto .

- **Vars**: Variables, es un fichero que puede guardar variables para posteriormente utilizarlas en el *playbook.*
- **Files:** Archivos, se almacenan los archivos que queramos copiar o mover al equipo donde se estén lanzando los comandos.
- **Handlers:** Manejador de excepciones.
- **Meta:** Metadatos relaciones con el proyecto (autor, fecha de creación, …).
- **Defaults**: Define variables predeterminadas, son las de menos prioridad y pueden ser sobreescritas por cualquier otra definición de variable, como las que se definan en el directorio vars.
- **Tasks**: Define todas las tareas que se llevarán a cabo en este rol, utilizando los recursos y definiciones que se encuentren en el resto de directorio y ficheros siendo **esta la carpeta más importante, ya que contiene el fichero "main" del proyecto**, y es lo que se ejecutará.

Se pueden generar otros ficheros de extensión *.yml* que realicen tareas o funciones independientes que complemente a la función main, como "clases" auxiliares. Por ejemplo podemos hacer un *playbook* independiente para reiniciar un servicio, y llamarlo las veces que queramos.

Un *playbook* muy pequeño, a modo de prueba o similar, no requeriría ninguno de estos componentes añadidos, pero para un proyecto más grande sí, y Ansible permite la creación de roles que permite una estructura de carpetas mejor organizada.

2.2 Arquitectura en Ansible

Ansible, debido a que su funcionamiento se basa en conectarse a través de SSH al host al que se le quiere aplicar los cambios, ejecutar las órdenes y finalizar su ejecución, es natural plantear un modelo cliente - servidor como topología para este proyecto.

Cliente:

Este agente se encargará de iniciar la conexión con los servidores, realizando los cambios en los equipos para instalar Informatica MDM en los mismos.

Tendrá que tener los binarios de la instalación y la herramienta Ansible instalada.

Servidor:

Este, por otra parte, solo requerirá de un canal SSH disponible para la máquina cliente.

Resulta un poco contradictorio en esta arquitectura que el sistema que tenga todo el código con la información de los cambios que se requieren para la instalación, sea el que envíe peticiones a otras máquinas para que ejecuten la carga de trabajo.

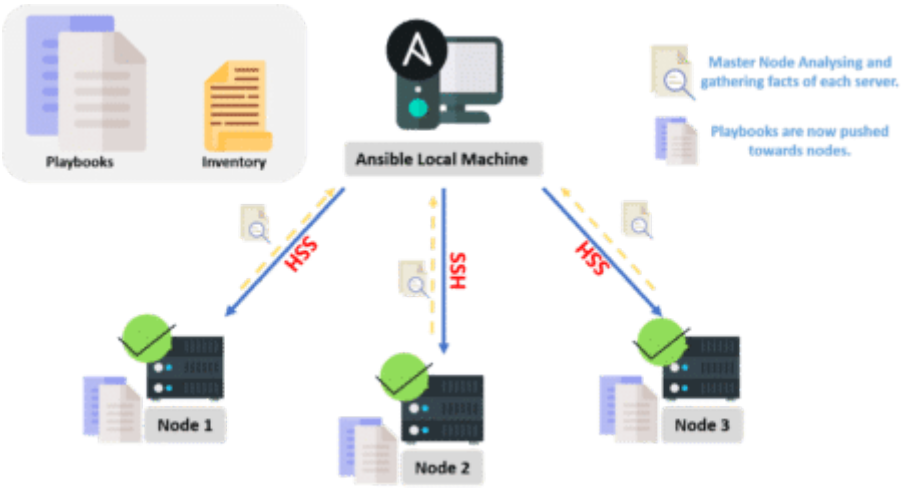

Fig.3 Estructura de ejecución en Ansible[4]

3. Entorno de trabajo

Para seguir la arquitectura que usa Ansible en su funcionamiento, se emularán 2 máquinas virtuales en local, a través de VirtualBox (Podría haberse utilizado VMWare sin ningún tipo de inconveniente, para el proyecto ambas aplicaciones desempeñarían su tarea exactamente igual), configurando una red interna para permitir la comunicación entre ambas.

Se usará la distribución de Sistema Operativo **Ubuntu 20.04**, evitando así problemas de compatibilidad, tanto entre las máquinas, como con el resto de los componentes.

Por lo que el primer paso es descargar una imagen de uso libre del Sistema Operativo, y crear 2 máquinas con las siguientes especificaciones:

- 8GB de memoria RAM, suficiente para ejecutar **Informatica MDM**
- 100Gb de memoria física *dynamic allocated*[16],espacio de sobra para los binarios
- Usar la imágen ISO descargada como *Optical Drive* de arranque del equipo, para poder instalar el SO
- Configurar interfaz de red puente para tener acceso a internet y una interfaz de red exclusiva para los entornos

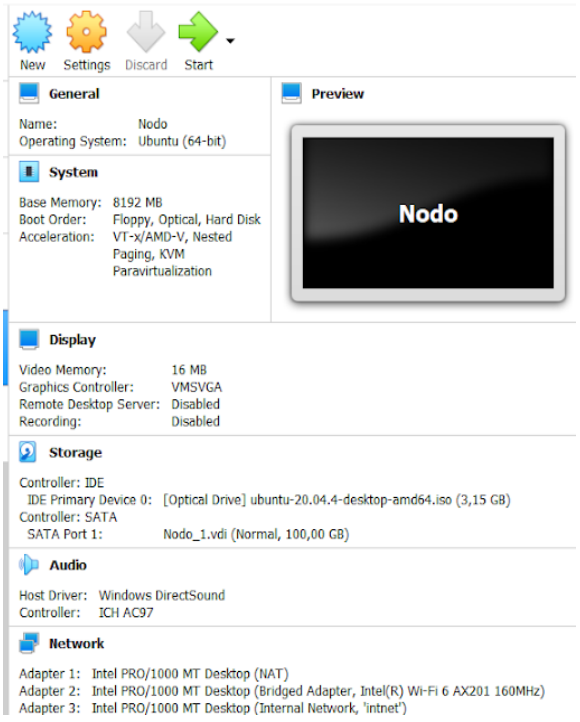

Fig.4 Propiedades de los sistemas

[16] *Dynamic allocated* permite que la máquina use la memoria que le hemos declarado, pero no ocupa ese espacio en el sistema hasta que lo requiera

Una vez creadas las máquinas con las propiedades necesarias para trabajar, se instalará el Sistema Operativo con los parámetros predeterminados, y con un usuario predeterminado y administrador del sistema.

***Master*:**

En este sistema en el que estará instalado Ansible, generando el proyecto y todo el código para la automatización.

También se modificarán las medidas de seguridad para que se habilite el servicio *SSH* e instalar el paquete *net-tools* para poder manejar las interfaces de red de la máquina.

***Slave*:**

Y este sistema solamente se harán las modificaciones en seguridad necesarias para permitir las conexiones *SSH*, y los cambios necesarios para que se conecte un usuario desde una máquina externa y haga las modificaciones necesarias al sistema para que la aplicación pueda ser instalada.

Ambos equipos son muy similares a nivel de Infraestructura son idénticos, mismo SO, mismas aplicaciones instaladas, y mismo usuario administrador, pero su propósito y evolución será lo que les diferencie.

3.1 Preparación del entorno "*Master*"

La preparación de este nodo es mucho más sencilla, ya que no se requiere de tantos componentes para poder ejecutar Ansible.

Lo primero es instalar Ansible en el sistema con la siguiente orden:

sudo apt-get install ansible

Una vez descargados los paquetes se creará un *playbook* a modo de prueba, con el que verificar que funciona perfectamente.

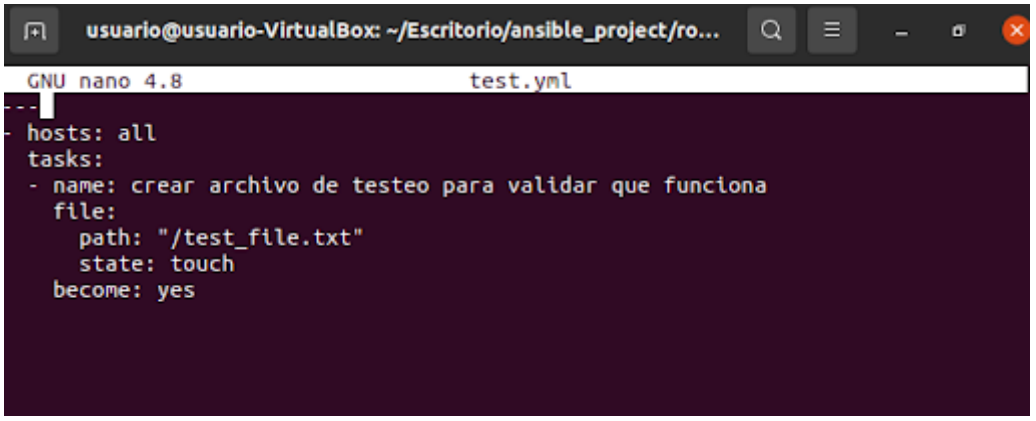

Fig. 5 Código para probar la conexión

Modificar el fichero **/etc/ansible/hosts** para que se conecte al equipo "*Slave*", introduciendo la IP que tenga actualmente, y lanzarlo para verificar que está correctamente configurado todo.

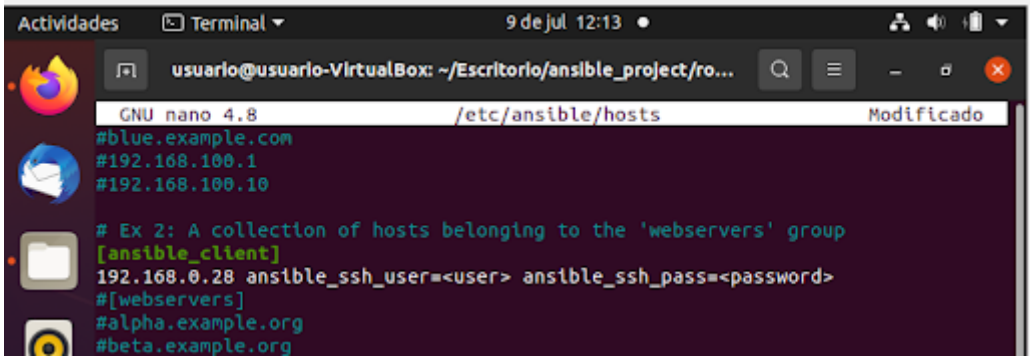

Fig. 6 Configuración de los hosts a los que se conecta

Hay que tener en cuenta que cada vez que se reinicie la máquina "*Slave*" se cambiará la IP por lo que es necesario comprobar antes de cada ejecución que la dirección IP es la correcta. Si se usara un servidor DHCP no aparecería este inconveniente**.**

Para lanzar el script generado, lo ejecutamos con la siguiente orden

ansible-playbook test.yml -K

La opción "**-K**" sirve para que solicite la contraseña a través de la consola de comandos para convertirse a root una vez se conecte al host, esto se hace para evitar problemas de permisos de escritura en el directorio raíz "/" y garantizar que no hay problemas de permisos para identificarse como superusuario más adelante.

```
usuario@usuario-VirtualBox:~/Escritorio/ansible_project/roles$ ansible-playbook
 test.yml -K
BECOME password:

PLAY [all] *********************************************************************
*

TASK [Gathering Facts] *********************************************************
*
ok: [192.168.0.28]

TASK [create empty file] *******************************************************
*
changed: [192.168.0.28]

PLAY RECAP *********************************************************************
*
192.168.0.28               : ok=2    changed=1    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

Fig. 7 Ejecución de la prueba

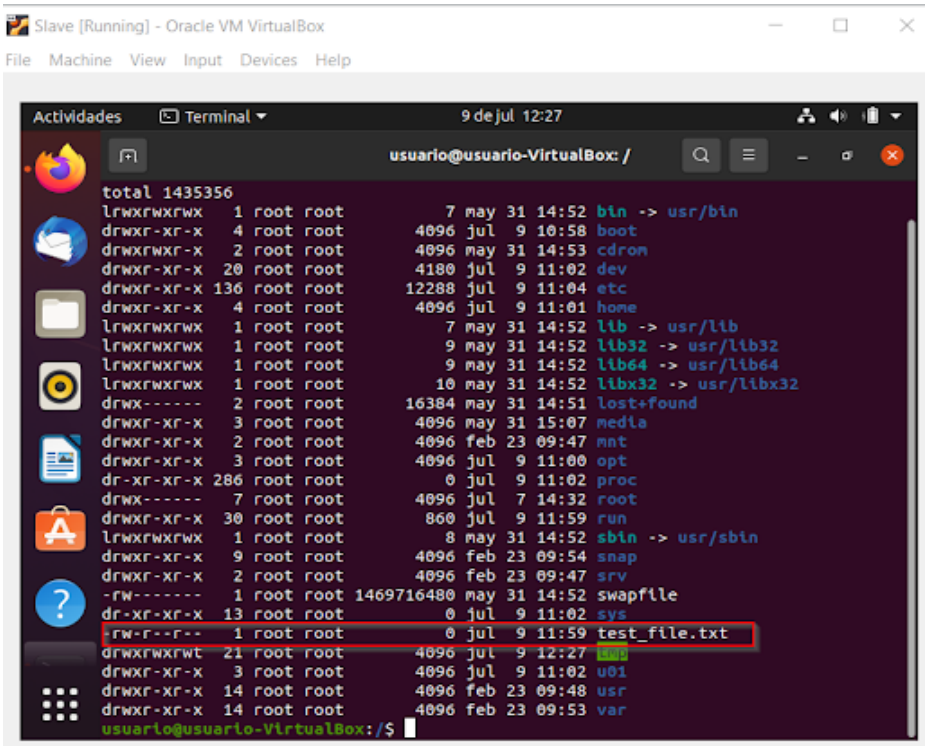

Fig. 8 Fichero *test_file.txt* creado correctamente

Y si ahora accedemos al host "*Slave*", se puede comprobar que se ha creado el fichero ***test_file.txt*** correctamente.

3.2 Preparación del entorno "*Slave*"

Para la máquina virtual "*Master*", se ha visto que usaremos las mismas especificaciones "*Hardware*", que el nodo "*Slave*". Ahora se instalarán las herramientas necesarias para empezar a trabajar.

Primero se actualizarán los paquetes existentes del SO.

sudo apt-get update

Luego se descargará el paquete net-tools para poder tener disponible la orden "**ifconfig**" y así posteriormente poder revisar las interfaces de red están y cuál es la dirección IP en la red interna.

sudo apt-get install net-tools

Después hay que instalar el paquete "**openssh-server**" que se utilizará para configurar y permitir conexiones SSH entre los equipos, ya que de forma nativa el firewall las bloquea por seguridad.

sudo apt-get install openssh-server

Tras eso, habilitar las conexiones **SSH** desde el firewall, para que a ese nivel no se inhabiliten las conexiones firewall.

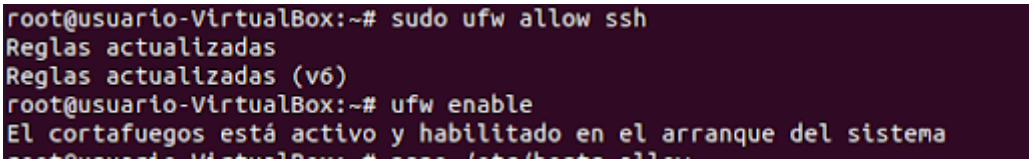

Fig.9 SSH habilitado

Por último, hay que cambiar un fichero de configuración del paquete ssh que nos hemos descargado para **habilitar las conexiones SSH a cualquiera, sin importar a qué grupo de red pertenezca**

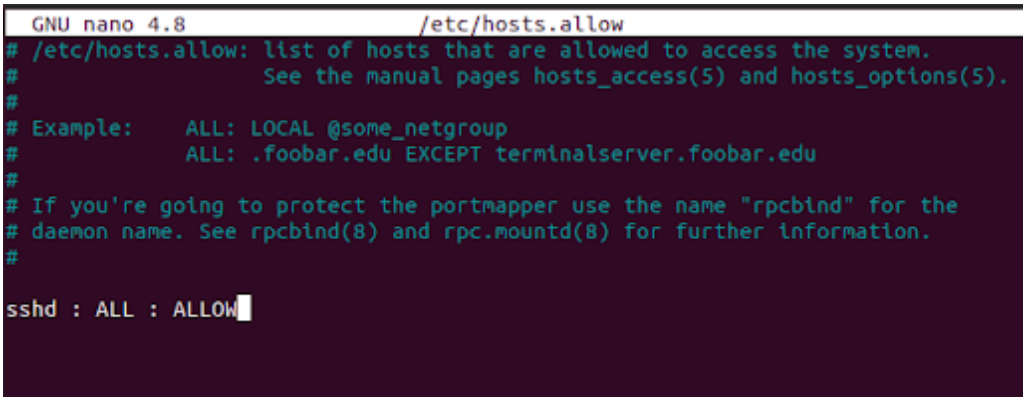

Fig.10 Configuración SSH

Ahora, verificar que nos podemos conectar a nuestro equipo que funcionará como "*Slave*" y que procese las órdenes usando el comando:

ssh usuario@localhost

En el último punto de la configuración se permite la conexión SSH desde cualquier host externo, al equipo actual. Esto es una brecha de seguridad, dado que es una red interna para los nodos, y que no hay forma de que nadie, traté de abrir una conexión a ese equipo, se ignorará.

4. Desarrollo

Una vez preparado el entorno, se dividirá el proyecto en distintos fases módulos, todas las partes se pueden ejecutar de manera individual, e independiente, en cualquier momento del proceso, y formarán en conjunto, el proceso completo de instalación y preparación para el su uso de la aplicación.

De esta forma, si en otro sistema solo sea necesario ejecutar una parte de la instalación, puede copiarse el proyecto que realice la instalación o configuración de ese componente y ejecutarlo.

- Instalación de Java
- Instalación de JBOSS
- Configuración de JBOSS
- Instalación de Oracle como SGDB
- Creación de las bases de datos
- Instalación de la aplicación
- Validación de la instalación

Esta estructura también facilita la depuración y el análisis de errores, facilitando también la actualización de distintos módulos de forma individual, siguiendo, eso sí, un órden obligatorio entre los distintos módulos para su correcto funcionamiento, ya que existe una dependencia entre ciertos módulos.

No es posible ejecutar la instalación del JBOSS, antes que la instalación de la distribución de Java que incluye esta aplicación, ya que para que JBOSS se pueda ejecutar correctamente requiere de una versión de Java compatible instalada.

Es necesario que todas las ramas previas a la instalación/uso de la aplicación estén completadas.

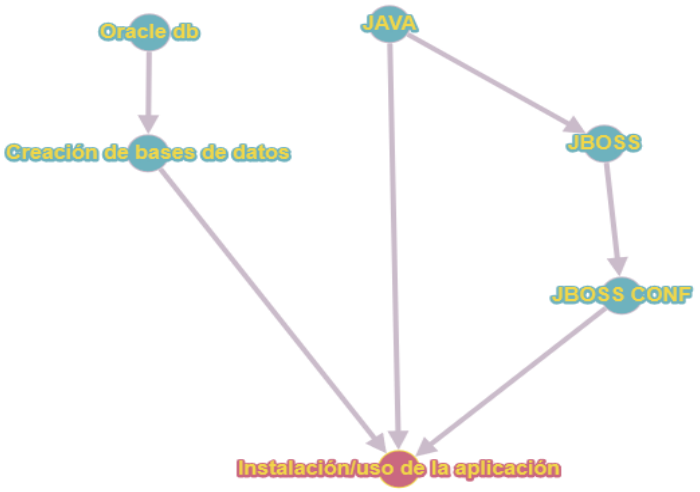

Fig. 11 Grafo de dependencias

Este grafo explica la dependencia que existe entre los módulos para que se ejecuten correctamente los componentes que se instalan, esto abre la posibilidad de paralelizar distintos procesos, ya que se pueden identificar dos procesos independientes:

- Uno que ejecuta la instalación de **Java → JBOSS → JBOSS CONF**
- Uno que ejecuta la instalación de **Oracle Db → Creación de la bases de datos**
- Y ambos procesos concluyen en la **Instalación de la aplicación**.

Como tal se pueden usar herramientas de gestión de procesos tales como semáforos, barriers, locks, etc. Hay muchas formas válidas y eficientes para paralelizar estos procesos de formas sencillas, siempre y cuando se satisfagan sus dependencias, es requerido por el desarrollador que lance el proyecto en Ansible, comprobar si hay instalado alguno de estos componentes en caso de querer ahorrar tiempo durante la ejecución.

Todas estas ideas son recogidas en este punto para futuros proyectos, a nivel de paralelización:

- Escoger un método para la paralización de estos 2 procesos identificados, con sus requisitos.
- En caso de que el estado actual del sistema ya cumpla los requisitos para instalar alguna aplicación o ya tenga instalada esa aplicación, omitir esos trabajos.
- Tener en cuenta la pesada transferencia de ficheros de instalación en esta aplicación, ya que hay que transferir los archivos desde el nodo “Master” al nodo “Slave” y no hay un rol que paralelice este proceso o que lo haga de forma independiente.

Para cada una de estas fases de desarrollo, se ha creado un rol independiente. No compartirán variables o recursos en común. Lo único que pueden compartir es una variable que sirva para identificar al usuario.

Por lo tanto la estructura de ficheros resultante, tras crear los subproyectos es la siguiente:

Fig. 12 Estructura final del proyecto

Y el fichero **full-install.yml** servirá para llamar a todos los roles en el orden requerido para que se ejecute correctamente la instalación.

4.1 Instalación de Java

Para esta parte se consultará la documentación [17]que aporta *Informatica*[18], la desarrolladora, con respecto al producto, donde nos encontramos que es necesario instalar una distribución de JRE (**Java Runtime Environment**) versión 1.8.

En este caso, ya que Java Azul Zulu [19]es un paquete muy sencillo de instalar en Ubuntu, es también una versión, que a día de hoy recibe actualizaciones y tiene soporte.

También se ha explorado la posibilidad de descargar la versión de Oracle JDK 8u202, pero dada la simplicidad de instalar el primer paquete, es mucho más conveniente instalar la versión Azul Zulu.

Primero se instalará **Java Azul Zulu 8** de manera manual en el sistema "*Slave*", y más adelante se regresará la máquina "*Slave*" al estado anterior a la instalación de Java para lanzar a ejecución este proceso a través de Ansible.

Como primera tarea se actualizarán los paquetes y servicios instalados en el sistema, esto se hace para que el repositorio tenga todos los servicios disponibles para actualizados.

Además de instalar las librerías que se irán necesitando a lo largo del proyecto

Hay que importar al repositorio del sistema manualmente una clave pública que proporciona la plataforma Azul Zulu para las descargas de sus distribuciones. Si no se añade esa "key" directamente al repositorio Ubuntu, no permite conectarse, no se pueden leer, ni descargar los paquetes de Azul Zulu.

También crear un directorio **/opt/Informatica/Java,** donde se descargá un paquete llamado *zulu-repo_1.0.0-3_all.deb,* accesible desde la página de descargar de Azul Zulu, que se instalará en el equipo, este paquete contiene **la información necesaria para actualizar el repositorio local y añadir las direcciones, y el nombre, de los paquetes de Azul Zulu para instalar,** este paso es necesario ya que **de forma nativa no se puede instalar azulzulu8-jre desde la línea de comandos.**

Tras esto solo hace falta volver a actualizar con el comando **apt-get update**, y finalmente ya será posible lanzar la órden de **apt-get install zulu8-jre**.

Sin los pasos previos, de **añadir la clave pública, instalar el fichero zulu-repo_1.0.0-3_all.deb que permite actualizar el repositorio de librerías disponibles y después actualizar el repositorio, no será posible instalarlo.**

Una vez ejecutado este módulo, si ejecutamos la órden **java -version** aparecerá lo siguiente:

```
usuario@usuario-VirtualBox:~$ java -version
openjdk version "1.8.0_332"
OpenJDK Runtime Environment (Zulu 8.62.0.19-CA-linux64) (build 1.8.0_332-b09)
OpenJDK 64-Bit Server VM (Zulu 8.62.0.19-CA-linux64) (build 25.332-b09, mixed m
ode)
usuario@usuario-VirtualBox:~$
```

Fig. 13 Validación de java correctamente instalado

[17] Enlace a la documentación de los requisitos de instalación de la aplicación
[18] Informatica es una empresa que ofreces soluciones cloud y programas de gestión de datos
[19] Azul Zulu es un software reciente que da soporte a las aplicaciones que usan versiones de Java

De hecho, si se explora el equipo se podrá observar que está instalado en ***/usr/lib/*** de forma que es accesible para todas las aplicaciones del sistema, ya que este directorio es accesible desde la variable **$PATH**.

Fig. 14 Carpeta de instalación de Java

Todo esto indica que la instalación del paquete de Java, ha sido todo un éxito, posteriormente hay que replicar los mismos pasos en un *playbook* en Ansible.

El apartado de actualizar el repositorio y de instalar los paquetes necesarios es sencillo, son 2 módulos que vienen de forma nativa en las librerías predeterminadas de Ansible, y se les puede llamar a través de la siguiente sintaxis.

```
- name: Update apt
    apt:
      update_cache: true

  - name: Install packages
    apt:
      name: gnupg libiaol
```

También hay otro módulo para añadir keys al repositorio Ubuntu

```
  - name: Añadir una llave para poder conectarse a cdn.zulu.com
    ansible.builtin.apt_key:
      keyserver: hkp://keyserver.ubuntu.com:80
      id: 0XB1998361219BD9C9
```

Ahora descargar el fichero zulu-repo_1.0.0-3_all.deb e instalarlo.

```
- name: Descargar azul zulu 8.deb
  get_url:
     url: https://cdn.azul.com/zulu/bin/{{ zulu_package }}
     dest: /opt/Informatica/Java

 - name: Añadir el paquete .deb al repositorio Ubuntu
   apt:
     deb: /opt/Informatica/Java/{{ zulu_package }}
```

Se usa la variable zulu_package, ya que pueden existir otros paquetes de la aplicación que se quieran usar más adelante, de forma que si se quiere elegir otra versión de paquetes zulu simplemente es cambiar la variable que usa este código.

La principal ventaja de este método es el dinamismo que aporta, y la ágil modificación para nuevas versiones.

Lo negativo de este método es que si se modifica sin asegurarse que la nueva versión es compatible con la aplicación a instalar, pueden surgir problemas de compatibilidad.

Y usar la misma sintaxis usado anteriormente para instalar las librerías libiaol y gnupg, para instalar zulu8-jre, una vez hecho todo esto, ejecutarlo a través de Ansible y realizar las validaciones en la máquina "***Slave***" restaurada.

Se puede reutilizar indefinidamente, la misma "***key***", ya que no caduca, ni requiere de otros pasos intermedios para volver a usarla.

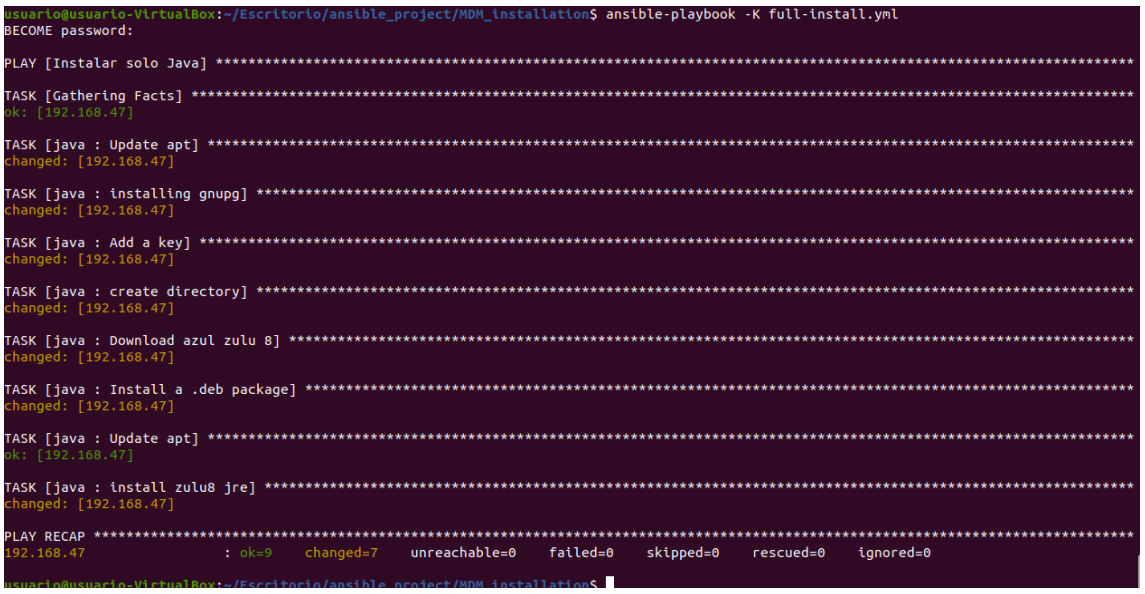

Fig. 15 Traza de la ejecución de la instalación de Java

Como se puede ver en la anterior figura, Ansible ha podido realizar todos los pasos satisfactoriamente.

Si se accede al sistema "Slave" se puede validar que, en el aspecto de la instalación de Java, **es una imagen equivalente al resultado de cuando se realizó la instalación de forma manual, del mismo componente**, el paquete está en el directorio */usr/lib* y se pueden ejecutar comandos Java.

Los servicios tienen la posibilidad de ejecutarse y leerse por cualquier usuario y grupo, son de propiedad del usuario root del sistema, tal y como en la otra máquina, y también tiene los enlaces simbólicos en la misma carpeta */usr/bin*.

Esto es interesante para portarlo a otros proyectos, ya que poder tener instalado rápidamente una JRE de Java de forma automática puede ahorrar mucho tiempo y retrasos inesperados, además de que el script es bastante accesible y susceptible para parametrizar para otras distribuciones de Java.

4.2 Instalación de JBOSS

Como entre los requisitos que necesita la aplicación para su despliegue y utilización, es la de una plataforma de despliegue de aplicaciones, y ya existe una JRE disponible para su uso, lo siguiente es proceder a la instalación de una plataforma compatible con la aplicación y con la versión de Java que hay instalada.

Hay 3 opciones: webSphere, JBOSS, y webLogic. Ya que durante la asignatura de Sistemas Distribuidos se estudia como el comportamiento de la plataforma JBOSS[20] en el despliegue y desarrollo de aplicaciones, y ya que las otras plataformas no son tan populares, **se usará esta como plataforma**, en su versión 7.3, ya que es la compatible para la aplicación que se quiere desplegar.

Primero se creará un usuario de aplicación, con el grupo correspondiente, y como directorio **$HOME** el directorio donde esté instalado *jboss-eap-7.3*. Este usuario será el único que ejecutará la plataforma, y solo tendrá visión del mismo directorio **$HOME**.

Esto asegurará que solo un usuario lanzará la plataforma, y no comprometerá la seguridad del resto del sistema, además de apoyar en la jerarquización los usuarios, siendo este un usuario que tome el rol de "daemon". De esta forma en caso de brecha de seguridad desde la plataforma o del usuario, no sería crítico para el resto del equipo.

Código:

A nivel de código lo primero será asegurarse que si existe el servicio, apagarlo, posteriormente crear el usuario **jboss-eap-user** y añadiendolo a su grupo de usuarios correspondiente, y proporcionarle su directorio **$HOME**. A continuación, se debe limpiar el directorio donde queramos desplegarlo, en caso de que exista, y descomprimir el archivo de instalación en **/opt/Informatica/jboss-eap-7.3/**

Luego se modifica la configuración para que utilice el fichero de configuración *standalone-full.xml,* durante la ejecución, en vez de *standalone.xml*, ya que este último es una versión reducida de *standalone-full.xml,* que guarda TODA la información de conexiones a bases de datos, y demás configuraciones.

También se copiará un fichero de configuración, con los parámetros relacionados con los recursos de memoria de ejecución disponibles. Una vez hecho esto, se lanza la aplicación, y se valida el correcto despliegue de la plataforma.

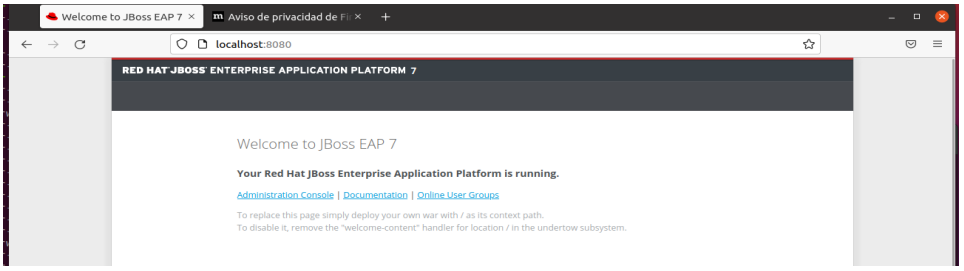

Fig. 16 Acceso a la plataforma JBOSS a través del navegador

[20] Jboss ofrece también herramientas de administración de usuarios, seguridad y despliegue de aplicaciones

Modificaciones:

Originalmente el código era como está descrito en el anterior punto, pero posteriormente, durante la instalación de la aplicación, apareció un error que decía que **es necesario que sea el mismo usuario el que lance el proceso de la plataforma JBOSS, y el que lance el instalador de la aplicación**.

Como en el momento del desarrollo de este punto no se sabía, más adelante se realizaron las modificaciones en el código necesarias para mitigar este fallo, haciendo que fuera el mismo usuario el que desplegará la plataforma, como el que lanzará la instalación, además de omitir las líneas que permitían el despliegue del servicio por el "daemon", esta funcionalidad se puede añadir más adelante.

Actualmente el servicio se lanza a través de línea de comandos con la orden ***standalone.sh -c standalone-full.xml -b 0.0.0.0 -Djboss.as.management.blocking.timeout=5000 &***

De hecho en los procesos activos en el sistema, se encuentra que el usuario que lanza el proceso es el predeterminado durante la instalación del SO:

Fig. 17 Servicio JBOSS en marcha

Para validar que está correctamente desplegado es añadiendo un usuario de administrador a través del script add-user.sh que está en el directorio */opt/jboss-eap-7.3/bin* y acceder como ese usuario.

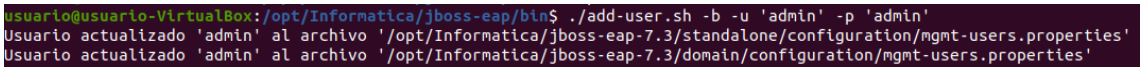

Fig. 18 Añadiendo usuario a JBOSS

De esta forma es posible monitorizar todo lo relacionado con la gestión de la plataforma, recursos desplegados, usuarios, seguridad, etc., accediendo con este usuario administrador a *http://localhost:8080*.

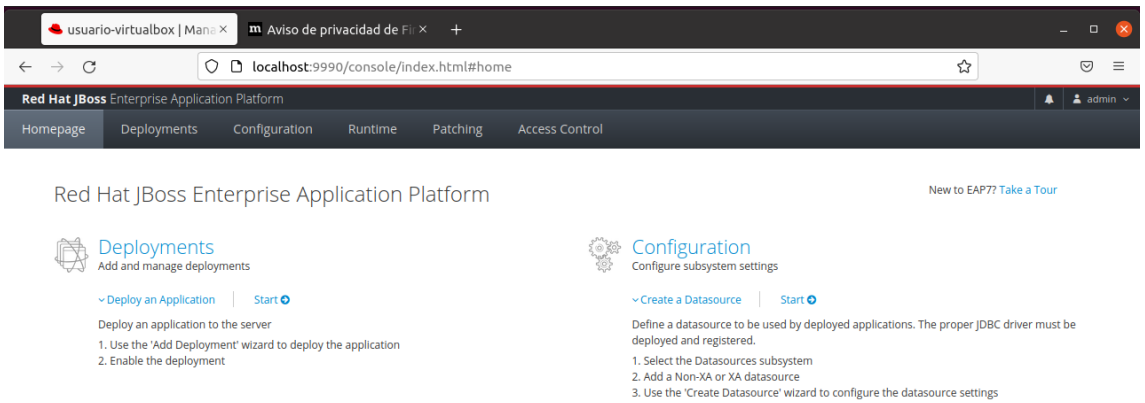

Fig. 19 Consola de administración web de JBOSS

4.3 Configuración de JBOSS

En la documentación aportada por *Informatica*, se indica que es necesario configurar la plataforma[21], cambiando algunas propiedades y atributos de la misma, para su correcto despliegue, y evitar que aparezcan errores más adelante.

Primero hay que lanzar a ejecución el fichero **jboss-cli.sh**, que es una herramienta para la gestión del servidor JBOSS. Se abrirá una consola que proporciona la capacidad de realizar ciertos cambios en las propiedades, editando ciertos ficheros, en función de los comandos que se ejecuten dentro de la misma.

Los cambios que hay que realizar son los siguientes:

- Definir el tiempo máximo entre transacciones

  `/subsystem=transactions:write-attribute (name=default-timeout,value=3600)`

- Tamaño máximo de los archivos que se suben a la plataforma JBOSS

`/subsystem=undertow/server=default-server/http-listener=default/:write-attribute(name=max-post-size,value=20000000)`

- Creo un puerto de acceso remoto, para poder acceder a la aplicación desde fuera de la máquina

`/subsystem=undertow/server=default-server/http-listener=default/:undefine-attribute(name=redirect-socket)`
`/socket-binding-group=standard-sockets/socket-binding=remoting:add(port=4447)`
`/subsystem=remoting/connector=remoting-connector:add(socket-binding=remoting)`

El problema surge porque esta consola solo permite leer a través de la entrada estándar, el teclado en este caso, de forma que dificulta la automatización de este paso a través de Ansible, de forma que existen varias soluciones posibles:

- **Simular la entrada estándar**: Usar una librería de Ansible que permita simular que el usuario escribe a través del teclado
- **Modificar manualmente las propiedades**: Buscar los ficheros de propiedades que se modifican, y sustituirlos por los ficheros ya editados.
- **Hacer un fichero de entrada estándar:** Crear un fichero .txt con las órdenes y al lanzar el servicio usarlo de entrada estándar:
- **Crear un fichero batch:** Un fichero que soporte esta herramienta, escrito en una sintaxis que entienda, y pasarlo como argumento

[21] Toda la información relacionada con los cambios que se realizan están aquí

Simular la entrada estándar: La librería que implementa esta funcionalidad se llama "**stdin**", viene por defecto cuando se instala Ansible en el equipo y la sintaxis es la siguiente:

```
- name: conf jboss-cli.sh
  command: jboss-cli.sh
  args:
    stdin: "{{ commands }}"
```

El primer problema que aparece al plantear esta solución, **es la falta de documentación** con respecto a esta librería, debido a que esta herramienta se encuentra en desuso, y **no tiene una depuración de errores eficaz,** muchas veces los intentos de resolver o avanzar en un proyecto con este método resulta en abandono, sobretodo cuando se pretende llevar a cabo una tarea más compleja**.**

El problema surge al plantearse cómo emular los saltos de línea en la entrada estándar, como ejecutar cada órden dentro de esta consola, como indicar que salga de la consola de gestión, o como verificar que se ha ejecutado sin error, ya que las facilidades de depuración con esta librería son nulas, no existe "feedback" alguno del progreso de la tarea, en caso de fallo, o de éxito la salida es la misma, y tampoco había logs de los cambios que estaba realizando.

A pesar de todos esos problemas que surgían ya antes de empezar a desarrollar una solución con este módulo, se ejecutaron algunas pruebas.

Primero emulando que en la líneas de comandos de Linux escribiera **echo "test":**

```
- name: prueba
  command: echo
  args:
    stdin: "test"
```

Tras varios intentos fallidos, modificando la entrada estándar "test", ya sea añadiendo un "\n" al final, como añadiendo una cadena vacía, etc., se abandonó esta solución para explorar otras que puedan ser más eficientes.

Modificar manualmente las propiedades: Esta segunda opción consiste en revisar las propiedades que se cambian al ejecutar esas órdenes en la consola interactiva, y realizar esos cambios manualmente, sustituyendo esas líneas por otras como ya se hizo en el apartado de instalación y despliegue de JBOSS, cuando se cambiaron los parámetros JBOSS_OPTS y JBOSS_USER.

El problema surge cuando no está claro qué ficheros se modifican, o qué líneas o atributos de los mismos hay que editar, en JBOSS hay muchísimos ficheros de propiedades, donde se definen el comportamiento de la plataforma, además de que no hay muchas documentación con respecto a los ficheros que se accede al interactuar con la consola jboss-cli.sh.

Redirigir la entrada estándar: Durante la carrera se aprende a manejar las entradas y salidas estándar de los procesos, por lo que podría ser una solución factible.

En un fichero *input.txt* se añaden las líneas con los comandos que se quieren lanzar, y se lanza el proceso usando este fichero como entrada estándar.

Al contrario que la anterior solución, esta sí que permite realizar ciertas pruebas dentro de la máquina "*Slave*", y validar que si que lee la entrada estándar si la redirigimos, antes de implementarla en Ansible.

./jboss-cli.sh < input.txt

La desventaja de esta solución es que si hay algunas instrucciones que hubiera que modificar más adelante, hay que cambiar el fichero **input.txt** manualmente, y en caso de que no se hubiera ejecutado correctamente, no sabríamos qué está mal hasta mucho más adelante de la aplicación, cuando la desplegamos en JBOSS.

Este método resultó satisfactorio, y tras varias verificaciones se concluyó que es una solución completamente válida.

Crear un fichero batch: Si se lee la documentación relacionada con esta consola interactiva, se puede encontrar que tiene varios argumentos con diversas opciones. Uno de ellos es el argumento **-f** que permite la entrada de un fichero con las órdenes que se quieren lanzar, si ya se sabe antes de abrir la consola.

La sintaxis de este fichero exige que se comience con una sentencia *connect,* luego abrir un bloque que se denomina *batch* en el que se introducen las órdenes a ejecutar, luego se cierra el bloque con el comando *run-batch* para que se lancen las líneas escritas.

connect
batch
****** Órdenes a ejecutar ******
run-batch

Lo mejor de esta opción es que en caso de que no se respete la sintaxis o las órdenes estén mal escritas te lo indica.

Por lo que se puede saber si se han ejecutado o no las órdenes, y no hace falta añadirle *quit* para salir de ese script, sale al terminar de ejecutar las instrucciones.

Esta última opción es la que menos desventajas tiene. Aunque sigue teniendo la problemática de que es necesario transferir un archivo con las entradas o comandos para cambiar la configuración de JBOSS, de forma que para, posteriormente, querer hacer cambios en esos comandos hay que cambiar el fichero fuente que se envía, pero de esta forma se puede saber si existe algún error durante la ejecución.

4.4 Instalación de Oracle SGBD

Tras la configuración y despliegue de la plataforma JBOSS, hay que automatizar la instalación de un Sistema Gestor de Bases de Datos Relacionales para su posterior uso por parte de la aplicación.

Revisando los requisitos para el SGBD que se recomienda para esta aplicación, se encuentra que hay disponibles distintas versiones de MSSQL y Oracle SQL, que fueron lanzadas entre 2011 y 2016 (Oracle 11.2, 12C, MSSQL 2016, etc.), con diferentes distribuciones (MSSQL Enterprise, Professional, etc.) aunque no es necesario tener esto en cuenta, ya que no es importante la cantidad de espacio disponible en estas bases de datos, sino el motor que usa.

Tras revisarlo, se decidió usar Oracle SQL 11.2 XE Xpress Edition, que no aporta mucha capacidad, apenas 5GB de espacio, pero es totalmente gratuito, sin que requiera licencia, y además la instalación de esta distribución es muy sencilla, y hay documentación sobre la instalación de esta aplicación usando Ansible, incluso se pueden encontrar máquinas en AWS preconfiguradas con distribuciones Oracle ya instaladas.

Primeramente se hace una instalación corriente siguiendo la documentación de la distribuidora para instalar Oracle 11.2, esta guía sugería crear un usuario y un grupo en el sistema, con su directorio home para la aplicación, luego descargar el instalador de la versión de Oracle que vayamos a usar, realizar unos cambios en las configuraciones del equipo, modificando algunos ficheros del sistema, e instalar la aplicación, arrancándolo como un servicio más del equipo.

Como parte de la configuración también se requería exportar ciertas variables de entorno, reiniciar la máquina para que se apliquen algunos cambios, y abrir un script con una consola de comandos para la gestión interna de la configuración de la aplicación, en la que se encuentra la misma problemática que se resolvió anteriormente en el apartado anterior de configuración de JBOSS.

Ansible de forma nativa tiene módulos “group” y “user” para crear crear grupos y crear o añadir un usuario a un grupo, siempre y cuando el usuario con el que ha accedido Ansible tenga los permisos necesarios.

También otro módulo que hay disponible es para crear directorio y darles permisos, o la posesión, a un usuario sobre el mismo, descargar y compartir archivos como se han hecho en otras etapas a través de los equipos conectados.

De forma que se crea un rol, posteriormente, en el directorio *files* del proyecto, se creará una copia de los ficheros de configuración que se modificaron del sistema durante las instalación manual, ya que estos parámetros son los mismos para cualquier sistema.

Los ficheros que vamos a modificar y su localización son los siguientes:

- *chkconfig*, Hay que copiarlo en */sbin.* Realiza configuraciones para crear el servicio oracle-xe (O el nombre que se prefiera), crea el arranque, la parada, el fichero de logs, etc.

- *60-oracle.conf* que se coloca en */etc/stsct.d/*. Este archivo indica que el servicio tiene permisos para escuchar sobre el rango de puertos expuesto (9000 – 65000) y otros parámetros para su correcto funcionamiento.

- *S01shm_load* que se instala en */etc/rc2.d/*. Esta configuración sirve para generar *locks* durante el inicio del servicio y que no haya problemas de acceso concurrente y derivados.

Es necesario arrancar el servicio procps[22], previo a esta última parte de la configuración.

Ansible simplifica estos pasos. Además, como esta configuración no requiere de personalización en función del sistema o usuario, no se añade más complejidad al desarrollo de esta parte del proyecto.

Sí que se podría cambiar el nombre a través del que se lanza el servicio, modificando el servicio *chkconfig* pero la infraestructura final no se vería afectada.

Ahora con el fichero de instalación descomprimido el directorio home del usuario oracle que se ha creado, hay que ejecutar la instalación de Oracle 11.2 XE con la orden

dpkg --install "{{ oracle_home }}"/Disk1/"{{ oracle_archive_deb }}"

Siendo las variables:

- *oracle_home* = **/home/oracle/**

- *oracle_archive_deb* = **oracle-xe_11.2.0-2_amd64.deb.**

Una vez instalado tenemos que usar un script de automatización, muy similar a lo que hicimos en el ***jboss-cli.sh***, pero esta vez no tenemos forma de pasar como argumentos de entrada un fichero *batch* con las instrucciones y hay que redirigir la entrada estándar a un fichero de texto creado previamente en el que se escriben las respuestas a los prompts que nos aparecerán.

/etc/init.d/oracle-xe configure < /etc/init.d/responses.txt

El fichero lo he llamado *responses.txt*, se copia en el directorio */etc/init.d/* y su contenido es:

8081
1521
oracle

oracle
y

[22] **procps** es un servicio que monitoriza los procesos activos en el sistema, de esta manera el servicio se autogestiona.

El fichero configura el puerto HTTPS (8081), el puerto de acceso (1521), usuario y contraseña del usuario administrador por defecto, y la última línea confirma los cambios.

Ahora queda modificar el fichero ~/.**bashrc** y añadir las siguientes variables de entorno:

```
export DISTR_CLASSPATH=$SIP_HOME/lib/ant.jar:$SIP_HOME/lib/ojdbc7.jar:...
export ORACLE_HOME=/u01/app/oracle/product/11.2.0/xe
export JAVA_HOME=/usr/lib/jvm/zulu8
export ORACLE_SID=XE
export ORACLE_BASE=/u01/app/oracle
export LD_LIBRARY_PATH=$ORACLE_HOME/lib:$LD_LIBRARY_PATH
export PATH=$ORACLE_HOME/bin:$PATH
```

Se añaden al final del archivo a con la función *insertafter: EOF* y posteriormente indicando el bloque que añadir.

Finalmente solo nos queda arrancar el servicio y comprobar que funciona.

service oracle-xe start

Si ahora se ejecuta **service oracle-xe status** en la máquina "*Slave*" se puede observar que el servicio está funcionando correctamente.

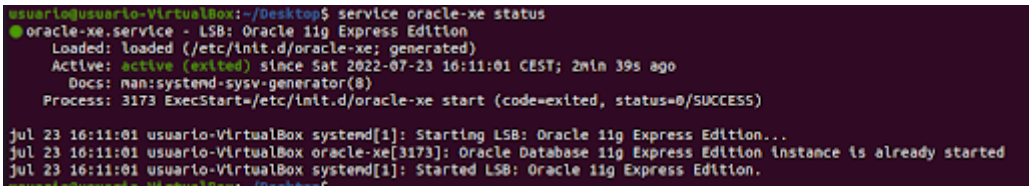

Fig. 20 Status del servicio *Oracle-XE*

De hecho si se ejecuta la siguiente orden, se abrirá una conexión, accediendo como administrador a la base de datos Oracle.

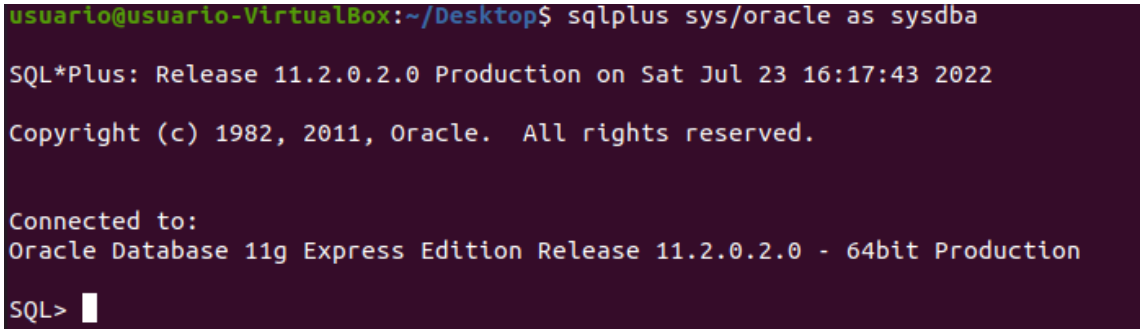

Fig. 21 Conexión al *SGBD Oracle* a través de sqlplus

Este comando es posible ejecutarlo gracias a que se ha añadido al ***PATH*** la ruta para ejecutarlo

Fig. 22 Variable *PATH* del sistema

4.5 Creación de las bases de datos

En esta etapa de la instalación se usarán los binarios y ficheros proporcionados por *Informatica* para la instalación de todos los componentes de la aplicación.

Es cierto que la transferencia de archivos se puede paralelizar en otro mejor momento de la instalación, aprovechar el tiempo de transferencia de los archivos mientras instalamos *Java*, o *Oracle SQL*, incluso podríamos configurar las máquinas de *AWS* para que se generen con esos ficheros de forma nativa.

De forma que en esta parte del desarrollo:

- Se creará el directorio donde estarán los instaladores de la aplicación y las consultas *SQL* para la generación de las bases de datos. La carpeta destino será configurable desde las variables locales del rol.

- Se le otorgarán a todos los elementos de este directorio, de forma recursiva, permisos de ejecución, ya que sino no será posible la instalación.

- También se crearán los esquemas y usuarios de las bases de datos, que se usarán durante la instalación y su posterior ejecución.

Usando la misma función, variando los ficheros fuente y destino, para transmitir los binarios para instalar *Informatica MDM* y *ActiveVos* [23](a partir de ahora se abrevia como *Avos*) en los directorios que se indiquen, reutilizando los módulos *unarchive, file* y *copy*, usados también, durante el proceso de instalación del *SGBD Oracle*.

Es necesario otorgar permisos de ejecución de forma recursiva a todos los elementos del instalador.

En la documentación de *Informatica* se detalla más todo el proceso

```
ALTER SYSTEM SET recyclebin = OFF DEFERRED;

CREATE BIGFILE TABLESPACE CMX_DATA
NOLOGGING
DATAFILE '/u01/app/oracle/product/11.2.0/xe/bin/CMX_DATA1.dbf' SIZE 1024M REUSE
EXTENT MANAGEMENT LOCAL;

CREATE BIGFILE TABLESPACE CMX_INDX
NOLOGGING
DATAFILE '/u01/app/oracle/product/11.2.0/xe/bin/CMX_INDX1.dbf' SIZE 1024M REUSE
EXTENT MANAGEMENT LOCAL;

CREATE BIGFILE TABLESPACE CMX_TEMP
NOLOGGING
DATAFILE '/u01/app/oracle/product/11.2.0/xe/bin/CMX_TEMP1.dbf' SIZE 1024M REUSE
EXTENT MANAGEMENT LOCAL;
EXIT;
```

[23] *ActiveVos* es un complemento a *Informatica MDM* que permite programar flujos de trabajos y administrarlos

Esta consulta lo que hará es crear ficheros temporales para la creación de los esquemas finales. Son el esqueleto de lo que será posteriormente, en caso de ser necesario, ese espacio que hay actualmente de 1024Mb se podrá ampliar.

El comando para lanzar consultas SQL en el sistema es:

sqlplus sys/oracle as sysdba @script.sql

Sin embargo el comando **no funciona cuando lo lanza Ansible** aparece el error:"**Command not found**" lo cuál es muy extraño porque momentos antes se había ejecutado el mismo comando y funcionaba correctamente.

El problema tiene que estar en las variables de entorno, que no estaban correctamente configuradas, o Ansible no las cargaba correctamente, dado que en la última comprobación que se hizo, pude ver que el comando lo encuentra desde la variable de entorno PATH.

En la documentación de Ansible se puede encontrar este módulo:

```
- name: Basic usage
  ansible.builtin.debug:
    msg: "'{{ lookup('ansible.builtin.env'. 'HOME') }}' is the HOME environment variable."
```

Estás líneas permiten imprimir por pantalla el valor de la variable $HOME, debido a que la librería **debug** ha recibido muchas actualizaciones, desde el momento en el que se publicó, esta función se encuentra obsoleta, y ya no funciona de la misma manera que aparece.

Pero sirvió como idea para desarrollar el siguiente script:

```
---
- name: Validar variables de entorno
  hosts: ansible_client
  become: yes
  tasks:
    - name: lanzar el comando
      shell: env | grep PATH
      register: output

    - name: ver la salida del env
      debug: var=output
```

Accede como root al sistema, lee las variables de entorno que hay y solo imprime por la salida estándar las que contengan la cadena "PATH", se guarda en una variable que posteriormente se muestra por pantalla con la función debug.

De esta forma se puede revisar si efectivamente no encontraba el comando porque no estaba cargando correctamente las variables de entorno.

La salida obtenida fue la siguiente:

Fig. 23 Salida obtenida de Ansible

Lo que confirma que la variable PATH no estaba siendo cargada desde el archivo ~/.**bashrc,** cargaba la variable descrita por defecto, sin la modificación efectuada en este último fichero.

También se lanzó la órden **sudo env | PATH** y el problema es el mismo.

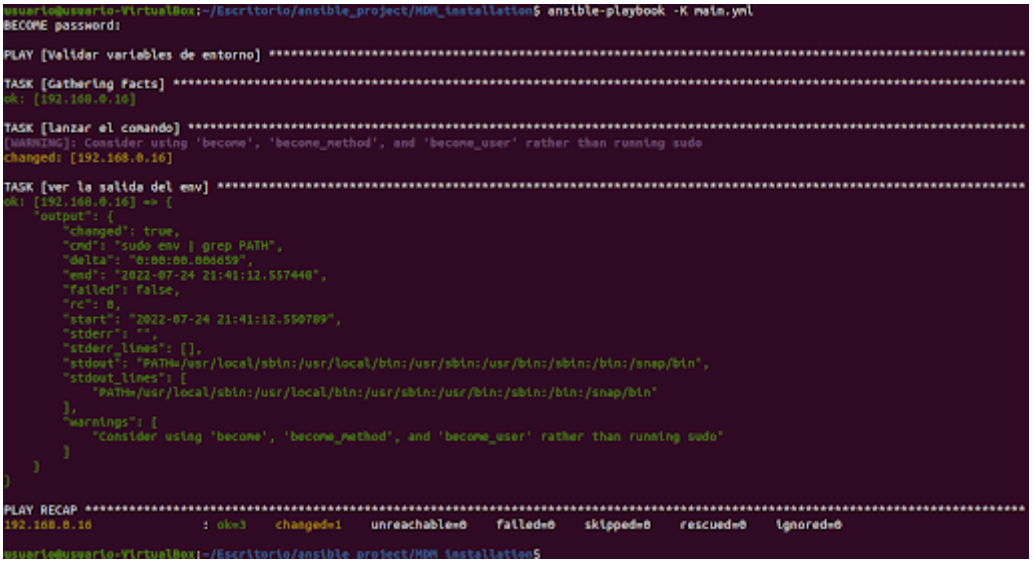

Fig. 24 Salida obtenida de Ansible con sudo

Tras comprobarlo, efectivamente las órdenes EXPORT seguían en el fichero **~/.bashrc.**

Se probaron varias soluciones a este problema:

- Forzando a que hiciera **source** al fichero (o sea, cargara la variables definidas) y posteriormente ejecuta esa órden en la misma órden, pero seguía sin cargar el PATH correcto.

source ~/.bashrc && env | grep PATH

- Usar la ruta absoluta a la aplicación **sqlplus** donde estaba instalada, pero luego no encontraba las librerías para ejecutar la aplicación

/u01/app/oracle/product/11.2.0/xe/bin/sqlplus

Finalmente, tras exportar las variables desde el fichero **/etc/profile, y** ejecutándolo como **sudo -i -u usuario env | grep PATH** se observa que esta vez sí que carga esas variables correctamente. Esto se hizo así porque en la documentación relacionada con los sistemas operativos de distribución Linux aparece que el fichero **/etc/profile** se carga siempre en el sistema, por tanto cargando las variables de entorno definidas en él.

Tras estos cambios se obtuvo la siguiente salida:

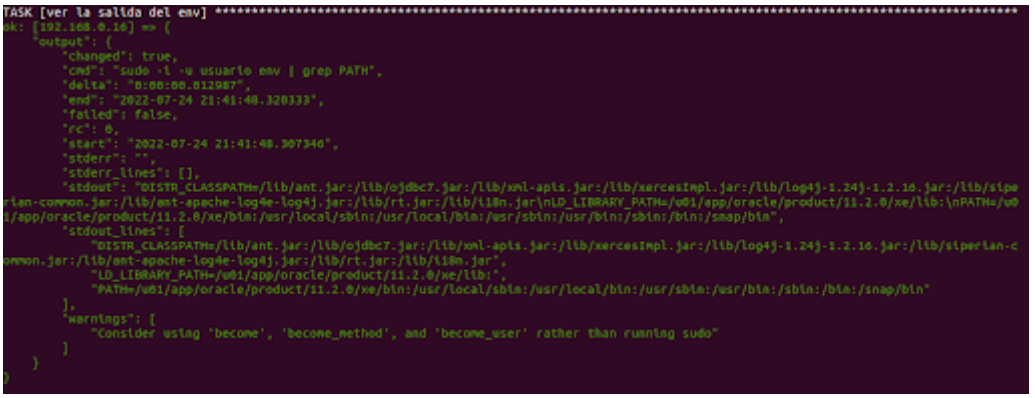

Fig. 25 Salida obtenida de Ansible con sudo -i -u usuario

De esta forma carga correctamente las variables de entorno, y además que Ansible las ejecute como el "usuario" que se conecte.

Tras esto, ahora sí que se podrá lanzar la órden en Ansible:

sudo -i -u usuario sqlplus sys/oracle as sysdba @script.sql

Y continuar con la creación de los esquemas de bases de datos.

La desarrolladora proporciona un paquete para descargar, que contiene los archivos necesarios para preparar unas bases de datos limpias para su instalación y uso, se descomprime ese archivo y accediendo al mismo tenemos el script **sip_ant.sh** que ofrece muchas funcionalidades.

- **cmx_system**: Crear los usuarios que usarán las aplicaciones, y otorgarles los permisos necesarios, y los esquemas vacíos de bases de datos.

- **import_system:** Se trae todas las tablas que usará la aplicación, y las añadirá a los esquemas creados por el *cmx_system*. Estas tablas servirán para manejar los procesos, usuarios, roles, contraseñas, nodos habilitados (en caso de estar ejecutándose en un clúster).
- **create_ors:** Similar a la opción *cmx_system*, crear los usuarios y les otorga los permisos para su funcionamiento.
- **import_ors:** Con este argumento lo que hace es importar las tablas para manejar los metadatos, creará índices y gestionará la información que queramos enriquecer (*Data Quality*), aplicando las herramientas y utilidades en gestión de cantidades masivas de información (*Big Data*).
- **update_ors:** En caso de querer actualizar un repositorio *ORS*[24] *(Operational Reference Store)* a una nueva versión habrá que usar esta opción.
- **create_bpm:** Este último crea los usuarios que usarán los procesos de *Avos* les otorgará los permisos, y también creará unas tablas que sirvan para indexar los procesos, tareas, o flujos de trabajos que se lanzarán.

Lo más importante es la *ORS*, ya que es donde se almacenará la información del cliente, sus trabajos, reglas de negocio y todo lo que se desarrolle se guarda ahí, eso quiere decir que en caso de fallo o de que alguna base de datos sea irrecuperable, podremos volver a crear la ***cmx_system***, sin más contratiempo que tener que volver a crear los usuarios que teníamos, y los flujos de trabajo, incluso en según que casos podremos salvar esos usuarios en un *Excel* y luego restaurarlos en una ***cmx_system*** creada de cero.

Para automatizar esta parte de la instalación hay que introducir todos los argumentos necesarios para los datos que requiere el script para funcionar, hay que ingresar el nombre de la base de datos, la contraseña, el usuario, el nombre del servicio, etc., en función de que vayamos a lanzar.

Como medida de parametrización de la instalación, se han definido unas variables en este proyecto para los argumentos del script, de forma que solo será necesario modificar las variables de la ejecución para cambiar este programa en otra instalación.

Para validar que todas las bases de datos se han creado de manera correcta hay que consultar el fichero ***sip_ant.log*** y aparecerá como ***"Build successful"***, además de poder acceder a esas bases de datos con el siguiente comando.

[24] *ORS (Operational Reference Store)* es un acrónimo para la base de datos que contiene la información de las transformaciones que usa *Informatica MDM* sobre los datos

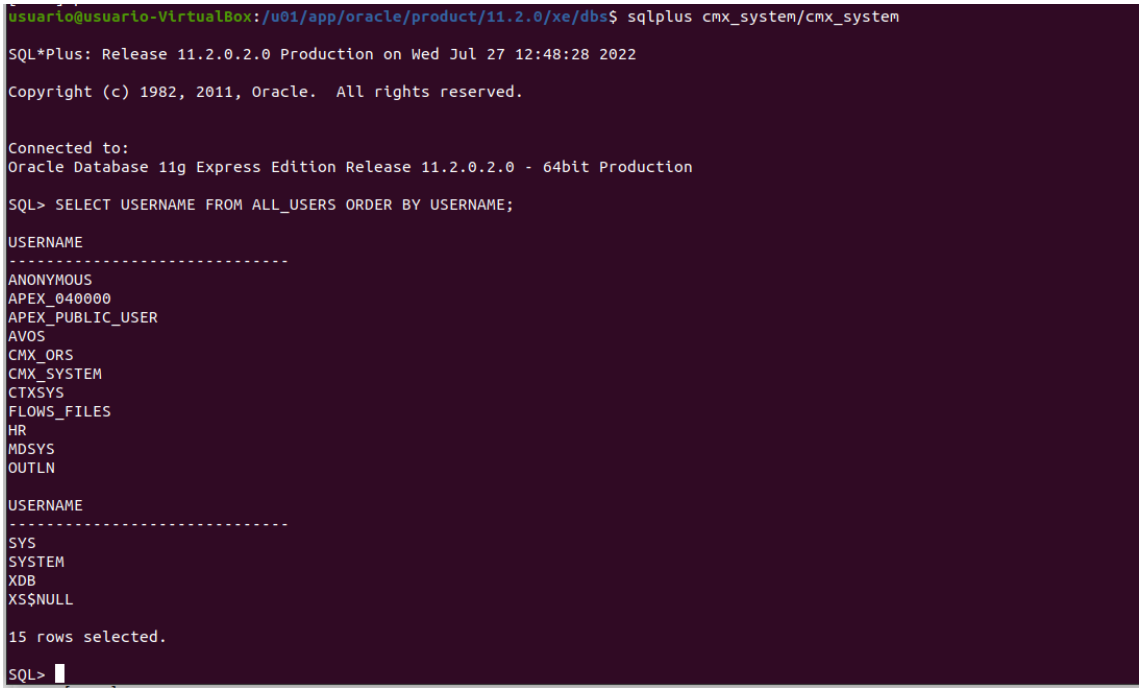

Fig. 26 Comprobación de usuarios creados con el create_system

Como se ve, se ha creado **AVOS, CMX_SYSTEM y CMX_ORS** correctamente.

4.6 Instalación de Informatica MDM

En esta última etapa del desarrollo se instalarán todas las aplicaciones necesarias de **Informatica MDM**[25] (Multidomain Data Management), hay que indicar el directorio de despliegue, se debe proporcionar la licencia de uso, los detalles de qué plataforma de lanzamiento de aplicaciones estamos usando, en este caso es JBOSS, los datos necesarios para conectarse a la base de datos Oracle y unas credenciales que use para gestionar los esquemas del SGBD.

Una vez instalado Informatica MDM HUB Server, se instalarán 2 paquetes complementarios que añaden nuevas funcionalidades, como una herramienta de gestión de usuarios y otra para gestionar los servidores de procesos, que serán el ResourceKit, y el Cleanse, y por último se instalará *Avos* que servirá para desplegar, crear, y lanzar trabajos de enriquecimientos de datos.

En la documentación de Informatica aparecen unas plantillas que se generan a través de "**InstallAnywhere**[26]", están disponibles para su uso desde su página de descargas, para modificarse las propiedades y ajustarlas a la instalación, pudiendo también desde Ansible personalizarlas, y de esa forma permite automatizar la entrada de los parámetros que necesita el instalador para que se ejecute.

Entonces para este proyecto hay que:

- Preparar los archivos para lanzar la instalación automatizada, subiendo ese fichero de propiedades.
- Lanzar la instalación.
- Desplegar los servicios en la plataforma.
- Validar que funciona correctamente.

[25] *Informatica MDM (Multidomain Data Management)* Este software se usa para el enriquecimiento y gestión de dominios de bases de datos

[26] ***Install Anywhere*** es un software usado para crear los instaladores de sus aplicaciones basado en Java

Instalación de Informatica MDM Hub Server:

En un primer momento, se lanza el instalador para Informatica MDM de forma corriente, para ver qué es lo que espera de entrada a proporcionar, anotando los datos que necesita para posteriormente, automatizar la entrada de los mismos.

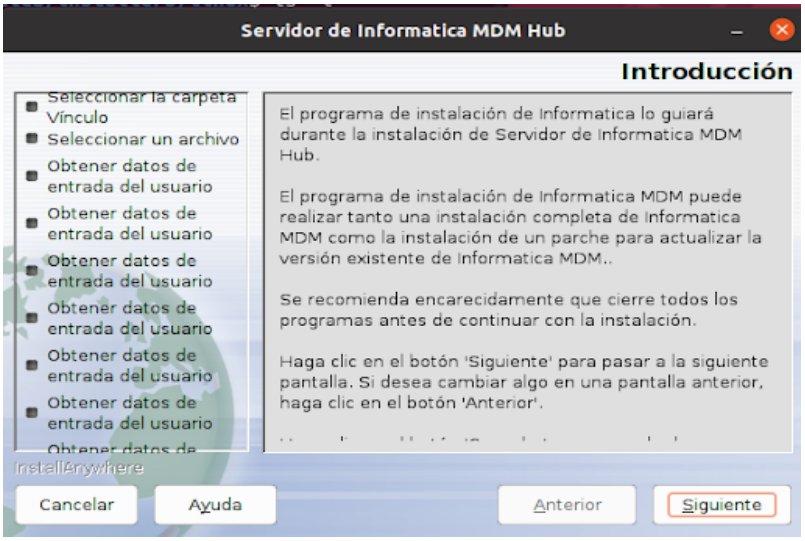

Fig. 27 Arranque del instalador con Interfaz de Usuario

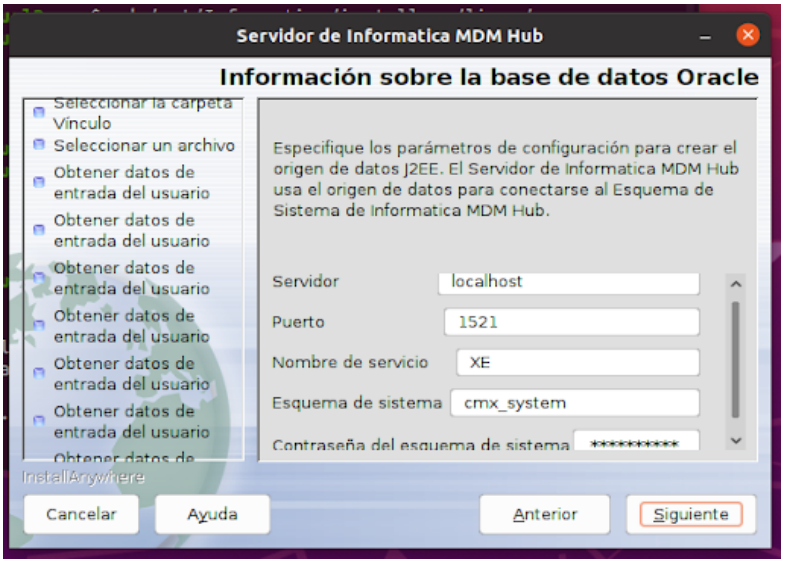

Fig. 28 Datos para la conexión a la base de datos

Pregunta por los datos para la conexión a la base de datos, hostname, puerto, tipo de base de datos, nombre del esquema, credenciales, etc., lo necesario para hacer consultas, leer datos, y realizar modificaciones.

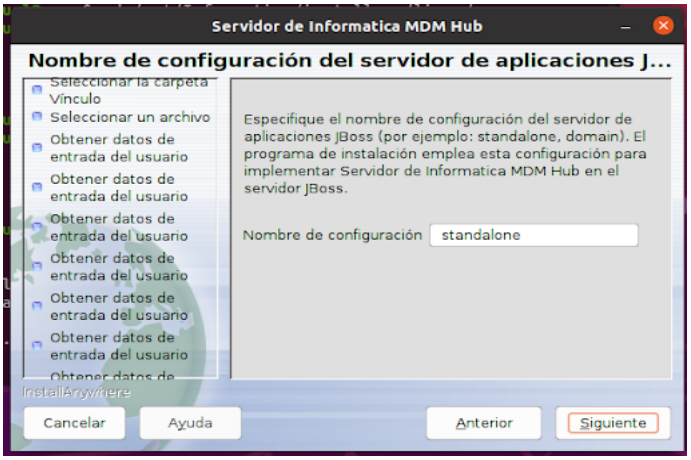

Fig. 29 Ruta de instalación de JBOSS

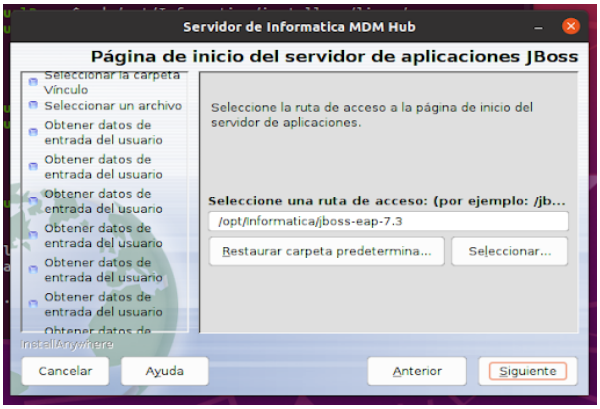

Fig. 30 Fichero de configuración que usaremos

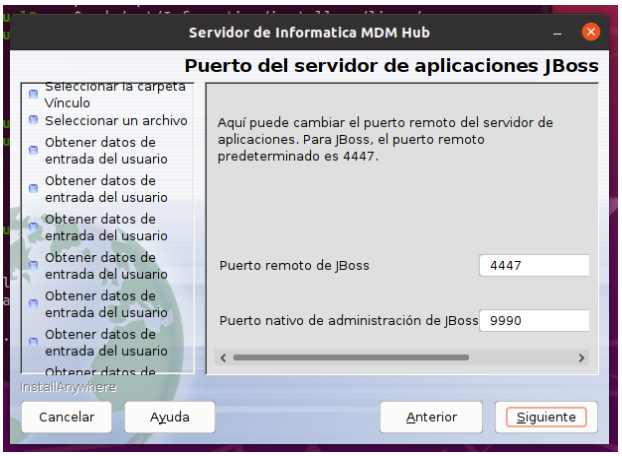

Fig. 31 Puertos a los que conectarse

Durante este proceso, resulta ser necesario que el mismo usuario que tiene arrancado la plataforma JBOSS es el que tiene que lanzar la instalación, sino no te permite avanzar.

Entonces es necesario cambiar el usuario que ejecuta la plataforma, para que sea el mismo que el que ejecuta la instalación.

Fig. 32 Resumen de las propiedades de la instalación

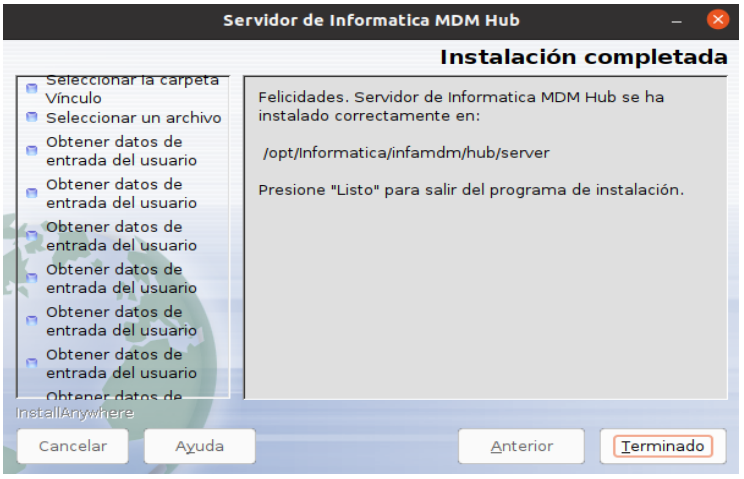

Fig. 33 Instalación completada

La instalación fue completada satisfactoriamente. De hecho, si se accede al directorio indicado previamente, veremos que la estructura de directorios está completa, y en los registros de la aplicación, se puede ver que está instalado correctamente, y consultar la versión actual.

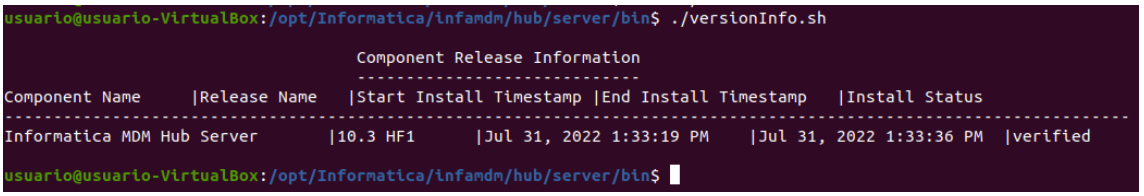

Fig. 34 Versión instalada correctamente

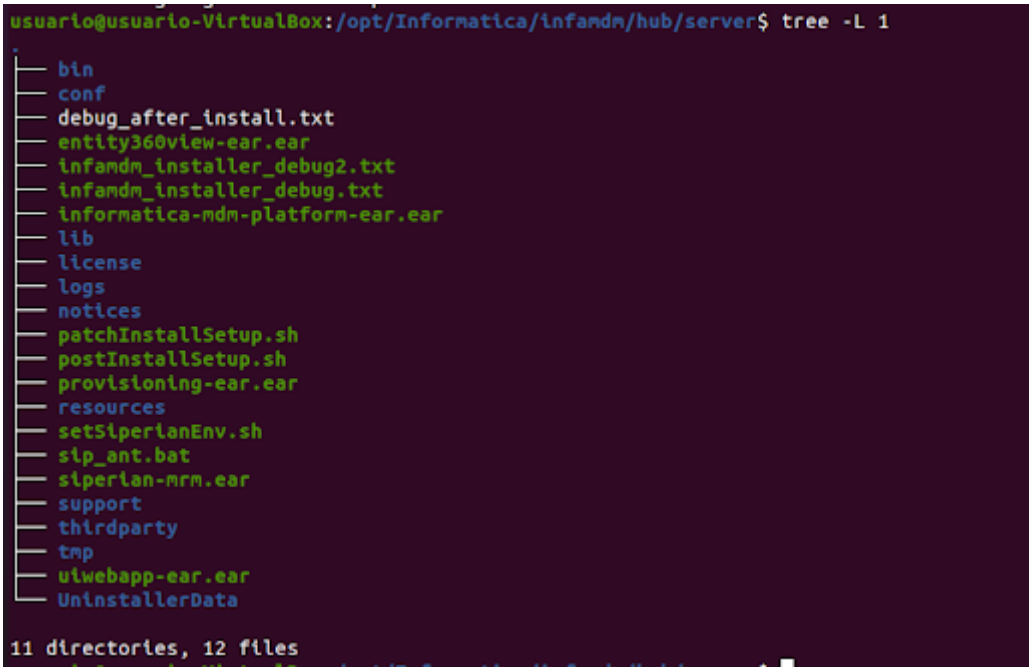

Fig. 35 Directorio de instalación desplegado correctamente

Ahora, ejecutando el **postInstallSetup.sh** se despliega la aplicación y se puede comprobar que todos sus servicios funcionan correctamente.

```
usuario@usuario-VirtualBox:/opt/Informatica/infamdm/hub/server$ ./postInstallSetup.sh -Ddatabase.password=cmx_system
------------------------------------------------------------------
Current folder : /opt/Informatica/infamdm/hub/server/bin
Current script : /opt/Informatica/infamdm/hub/server/bin/sip_ant.sh
Parameters :
JBoss Home : /opt/Informatica/jboss-eap-7.3
Siperian Home : /opt/Informatica/infamdm/hub/server
Java Home : /usr/lib/jvm/zulu8
Classpath : /opt/Informatica/infamdm/hub/server/lib/ojdbc7.jar:/opt/Informatica/infamdm/hub/server/lib/ant.jar:/opt/Informatica/infamdm/hub/server/lib/xml-apis.jar:/opt/Informatica/infamdm/hub/server/li
/xercesImpl.jar:/opt/Informatica/infamdm/hub/server/lib/log4j-1.2.16.jar:/opt/Informatica/infamdm/hub/server/lib/siperian-common.jar:/opt/Informatica/infamdm/hub/server/lib/ant-apache-log4j.jar:/opt/Inf
rmatica/jboss-eap-7.3/bin/client/jboss-cli-client.jar:/opt/Informatica/jboss-eap-7.3/bin/client/jboss-client.jar=:/usr/lib/jvm/zulu8/lib/rt.jar:/opt/Informatica/infamdm/hub/server/lib/ojdbc7.jar:/usr/li
/jvm/zulu8/lib/li8n.jar
------------------------------------------------------------------
OpenJDK 64-Bit Server VM warning: Ignoring option MaxPermSize=256m; support was removed in 8.0
Buildfile: /opt/Informatica/infamdm/hub/server/bin/build.xml
*** DEPLOYMENT SUCCESSFUL ***

*** View execution results in log file :  ./logs/postInstallSetup.log ***
```

Fig. 36 Ejecución del postInstallSetup.sh correcta

```
usuario@usuario-VirtualBox:/opt/Informatica/infamdm/hub/server$ ls -l /opt/Informatica/jboss-eap/standalone/deployments/
total 730684
-rw-rw-r-- 1 usuario usuario 110579854 jul 31 14:04 entity360view-ear.ear
-rw-rw-r-- 1 usuario usuario        21 jul 31 14:04 entity360view-ear.ear.deployed
-rw-rw-r-- 1 usuario usuario 141393528 jul 31 14:03 informatica-mdm-platform-ear.ear
-rw-rw-r-- 1 usuario usuario        32 jul 31 14:03 informatica-mdm-platform-ear.ear.deployed
-rw-rw-r-- 1 usuario usuario  70995566 jul 31 14:04 provisioning-ear.ear
-rw-rw-r-- 1 usuario usuario        20 jul 31 14:04 provisioning-ear.ear.deployed
-rw-r--r-- 1 usuario usuario      8888 feb 18  2020 README.txt
-rw-rw-r-- 1 usuario usuario 380850988 jul 31 14:03 siperian-mrm.ear
-rw-rw-r-- 1 usuario usuario        16 jul 31 14:03 siperian-mrm.ear.deployed
-rw-rw-r-- 1 usuario usuario  44350430 jul 31 14:05 uiwebapp-ear.ear
-rw-rw-r-- 1 usuario usuario        16 jul 31 14:05 uiwebapp-ear.ear.deployed
```

Fig. 37 Desplegado completamente desde el JBOSS

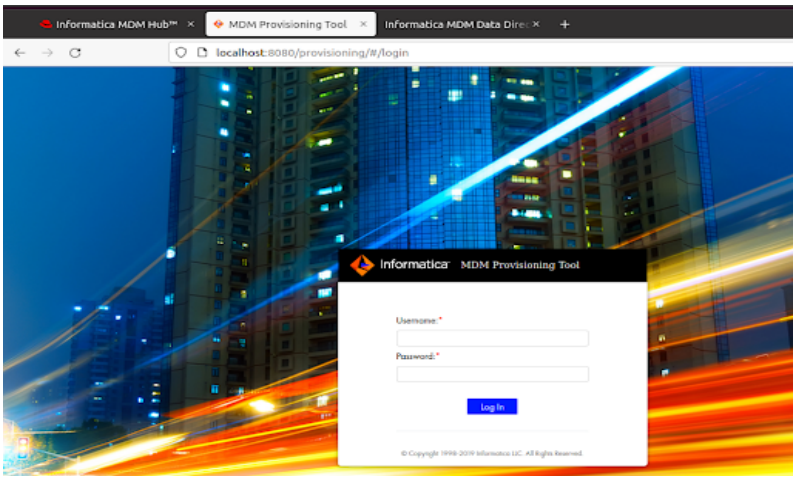

Fig. 38 Acceso a los servicio de desarrollo de Informatica MDM

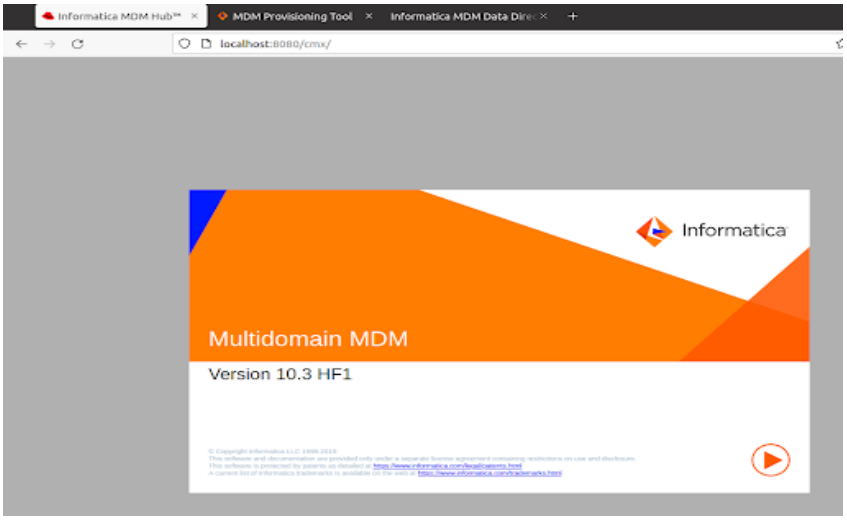

Fig. 39 Acceso a los servicio de administración de Informatica MDM

Accediendo a los servicios a través del navegador, se pueden realizar todas las comprobaciones necesarias, como poder leer la configuración, lanzar algún trabajo simple para comprobar la comunicación de la base de datos con la aplicación, creación y edición de usuarios, etc.

Una vez validado que de forma totalmente manual, la instalación ha sido correcta, hay revisar los pasos requeridos, las entradas de datos que se solicitaron durante la instalación, y que toda la instalación ha sido un éxito, se restaura la máquina al momento previo a la instalación, y se prepara el fichero **silentInstallServer_sample.properties**, con las propiedades y las entradas que espera el instalador.

```
14
15  # Directory where you want to install the Hub Server.
16  USER_INSTALL_DIR=/opt/Informatica/infamdm/hub/server
17
18  # Path to the Informatica license file.
19  SIP.LICENSE.PATH=/opt/Informatica/installers/licenses/
20
21  # License file name.
22  SIP.LICENSE.FILENAME=MDM_v103-Dev_9-8-2021.license
23
24
25  # Set the application server properties for one of the following application servers that you use in the MDM Hub environment: JBoss, WebSphere, or WebLogic.
26
27  # Properties section for the JBoss application server
28  #-------------------------------------------------------------------------------------
29
30  # Name of the application server. Specify JBoss.
31  SIP.AS.CHOICE="JBoss"
32
33  # Path to the JBoss installation directory.
34  SIP.AS.HOME=/opt/Informatica/jboss-eap-7.3
35
36  # JBoss remote port number.
37  JBOSS.AS.PORT_1=4447
38
39  # JBoss native port number.
40  JBOSS.AS.PORT_2=9990
41
```

Fig. 40 Extracto del fichero silenInstallServer_sample.properties

En el mismo instalador de Informatica MDM, aparece la opción de instalar Avos.

Primeramente se ignorará esta opción ya que suele producir bastantes errores, y al ser una parte muy pesada, ralentiza mucho la instalación del resto de componentes.

```
usuario@usuario-VirtualBox:/opt/Informatica/installers/linux/mrmserver$ ./hub_install.bin -f silentInstallServer_sample.properties
Preparing to install
Extracting the installation resources from the installer archive...
Configuring the installer for this system's environment...

Launching installer...

usuario@usuario-VirtualBox:/opt/Informatica/installers/linux/mrmserver$
```

Fig. 41 Instalándolo con la opción Silent

Se compara que ambos entornos, el que se ha hecho la instalación manual, y la que se ha hecho con el fichero SilentInstall, son equivalentes, y totalmente funcional.

Tras ejecutar el postInstall.sh también se desplegaban los mismos servicios, de forma que **podemos instalar todo Informatica MDM Hub Server con un solo comando, simplemente modificando el fichero .properties.**

Instalación del Cleanse[27]:

Cleanse funciona como servidor de procesos para Informatica MDM, sirve para gestionar y ejecutar ciertos flujos de trabajo que se lancen desde el HUB, estos trabajos están relacionados con filtros que se apliquen en las consultas a las bases de datos, o en los mapeos.

Siguiendo el mismo protocolo, en una máquina se hace una instalación corriente, revisando cuáles son los parámetros que hay que introducir.

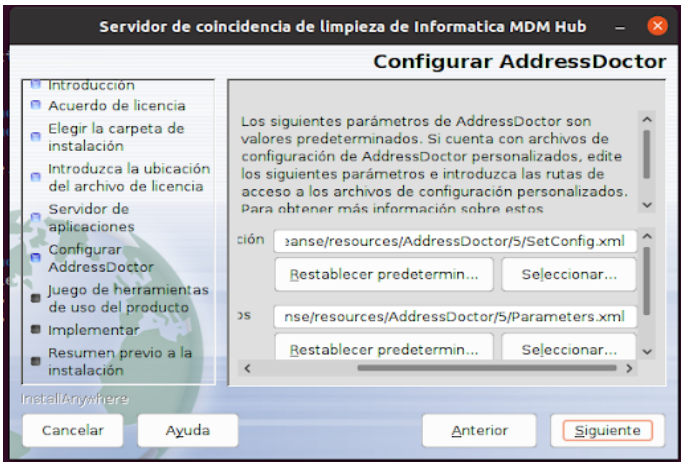

Fig. 42 Instalación del Cleanse

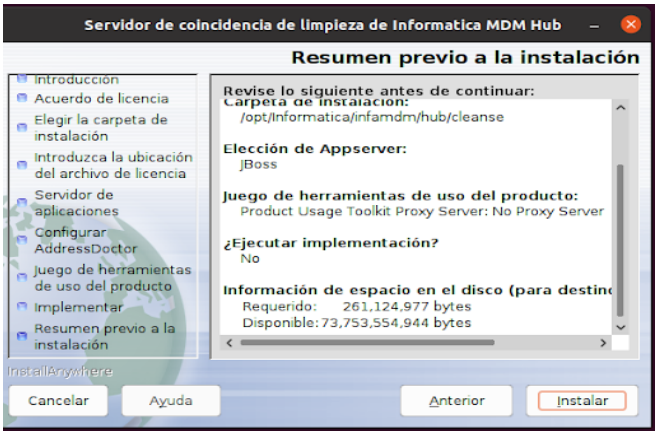

Fig. 43 Resumen de la instalación

La única diferencia en la instalación de este componente, con respecto a la instalación del Informatica MDM HUB Server, es la configuración de un añadido, que es el AddressDoctor.

Se dejará todo esto en predeterminado porque no es algo que se vaya a utilizar, y el resto de la automatización es totalmente equivalente.

Y ahora se ejecuta también el postInstall.sh del cleanse para verificar que todo funciona correctamente.

[27] Cleanse es un complemento a Informatica MDM que sirve para asignar y gestionar servidores de procesos, de forma que si se quieren gestionar flujos de trabajos creados desde Informatica MDM es necesario este complemento

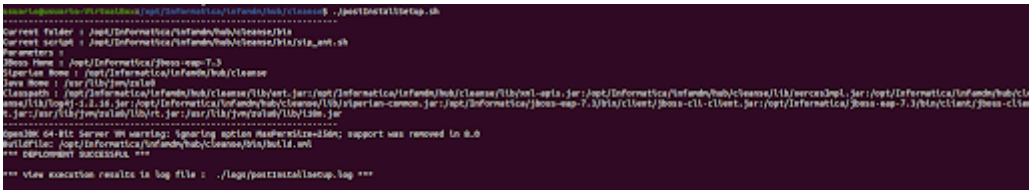

Fig. 44 Despliegue completado

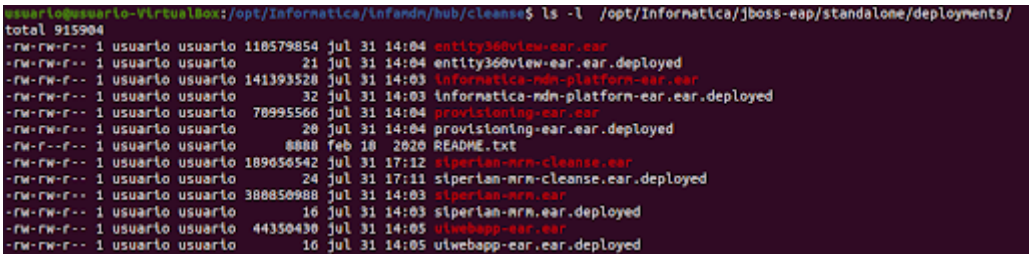

Fig. 45 Aparece también en el apartado de servicios del JBOSS

En este caso, este componente no tiene un acceso web al uso, sino que en el administrador de Informatica MDM HUB Server, se podrá**,** más adelante, **configurar el mismo equipo como un servidor de procesos,** y ya que este mismo sistema tiene instalado el componente siperian-mrm-cleanse.ear desplegado, se podrá asignar asi mismo como tal**.**

A continuación, y de manera similar, se modificará el fichero **silentInstallCleanse.properties** para poder realizar la instalación de manera silenciosa.

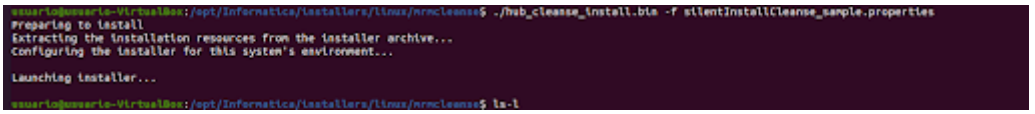

Fig. 46 SilentInstall de Cleanse

Tras restaurar el entorno al momento previo a la instalación de este componente, se comprueba que funciona correctamente la instalación con este método, reciclando algunos parámetros que también eran necesarios en el Informatica MDM Hub Server, como los datos de la plataforma que se está utilizando como JBOSS. Tras ejecutar el postInstall y comprobar que el cleanse se despliega correctamente, se puede concluir que la instalación del cleanse se ha automatizado con éxito.

Instalación del ResourceKit:

Por último, se lanza el instalador de forma corriente, revisando los datos de entrada que requiere este componente.

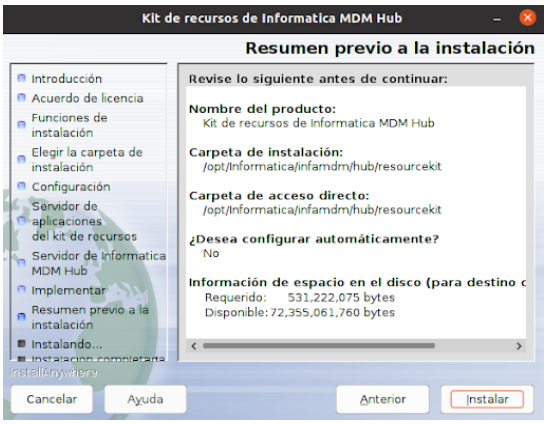

Fig. 47 Resumen de la instalación del ResourceKit

En este caso es la instalación más simple de todas, solo requiere de un directorio para instalarse.

```
usuario@usuario-VirtualBox:/opt/Informatica/infamdm/hub/resourcekit/deploy$ cat logs/postInstallSetup.log

checkIsLatestBuildDeployed:
     [echo] target file is /opt/Informatica/infamdm/hub/resourcekit/samples/PartyWebServiceExample/build/lastbuild.touch
     [echo] source file is /opt/Informatica/infamdm/hub/resourcekit/deploy/deployed_ears/PartyWebServiceExample.ear
     [echo] Latest PartyWebServiceExample is deployed

checkIsLatestBuildDeployed:
     [echo] target file is /opt/Informatica/infamdm/hub/resourcekit/samples/LookupWebServiceExample/build/lastbuild.touch
     [echo] source file is /opt/Informatica/infamdm/hub/resourcekit/deploy/deployed_ears/LookupWebServiceExample.ear
     [echo] Latest LookupWebServiceExample is deployed

BUILD SUCCESSFUL
Total time: 0 seconds
```

Fig. 48 Despliegue del ResourceKit

El despliegue ha sido satisfactorio, sólo resta añadir estos parámetros el fichero SilentInstallResourceKit.properties y los pasos para la instalación de **Informatica *MDM Hub Server*** y todos sus componentes habrá finalizado.

Ahora habrá que llevarse esos ficheros de propiedades al nodo "*Master*" para que desde **Ansible** se copien esos ficheros a cualquier máquina "*Slave*" ejecutando la misma órden para instalar usando los ficheros silentInstall, y también ejecutará los PostInstall.sh de la aplicación.

Instalación de ActiveVos:

Para la instalación de este apartado existen 2 opciones, usando el binario del instalador en crudo, es un fichero llamado **ActiveVOS_Server_Unix_9.2.4.5.sh**, que se ejecuta aparece un instalador bastante similar al que aparece cuando se instala Informatica MDM Hub Server, y también es posible instalarlo a través de una opción que nos aparece en este último instalador.

Esta última opción es más recomendada porque permite reciclar mucha información del instalador de Informatica MDM Hub Server, ahorrando así tiempo, y evitando errores en la entrada de datos.

Además, al usar el instalador específico de ActiveVos surge el problema de que este fichero no tiene la funcionalidad de automatizar la instalación del mismo con el fichero SilentInstall.properties que tienen el resto de componentes.

Este instalador tiene la posibilidad de registrar los parámetros que se indican en el instalador, si lo ejecutas como **ActiveVOS_Server_Unix_9.2.4.5.sh -q responses.txt**, pero después no te permite usarlo como parámetro de entrada para hacer una instalación automática. Su utilidad es más bien como un registro para el usuario.

Así que se decidió utilizar el instalador de Informatica MDM para instalar ambos componentes a la vez, ya que este ya estaba configurado y los parámetros introducidos, solo hacía falta habilitar en las properties que se quería instalar ActiveVos, añadir donde está localizado el instalador de Avos, los datos para conectarse a la base de datos, el nombre del esquema, usuario para conectarse, y las credenciales del administrador de Avos.

Se lanza el SilentInstall de Informatica MDM con la opción de instalar Avos.

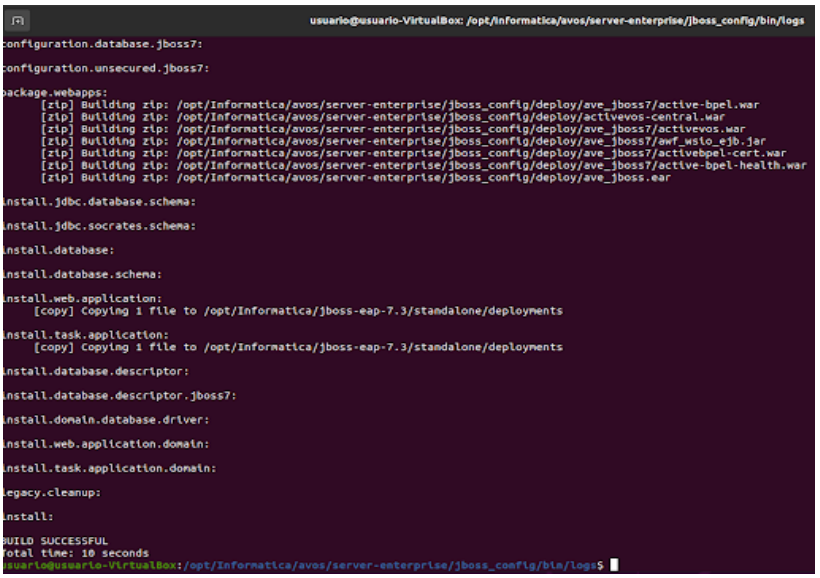

Fig. 49 Logs de la instalación de ActiveVos

Tras lanzar el instalador, toda la carpeta del Informatica MDM Hub Server, se desplegó de forma normal, y también se creó una carpeta con la instalación de Avos.

Ahora se lanza el despliegue de ActiveVos, añadiendo más argumentos, con información de la base de datos que usa, y el usuario para acceder.

postInstallSetup.sh -Ddatabase.password=cmx_system -Davos.username=avos -Davos.password=avos -Davos.jdbc.database.password=avos

En este caso el error que aparece al intentar desplegar la aplicación ave_jboss.ear es debido a que la aplicación de despliegue no encontraba un módulo instalado en JBOSS-7.3 (en este caso el módulo que buscaba era el **ojbdc7.jar**), que habilite la comunicación con la base de datos. Eso no tendría que pasar, porque uno de los pasos previos al despliegue de las aplicaciones en el postinstall es instalar esos módulos, además de crear canales de JMS.

```
hash_all_existing_passwords:
Trying to override old definition of task migrateUserPasswords
     [echo] All existing passwords are hashed !

jndi-variables-config:
     [echo] Configuring persistent jndi variables...

BUILD FAILED
/opt/Informatica/infamdm/hub/server/bin/build.xml:526: Error creating JNDI resource.
```

Fig. 50 Error al lanzar el PostInstall.sh de ActiveVos

Revisando otros ficheros *build.xml* de otros instaladores, se puede encontrar que en función de la versión que haya instalada de JBOSS, es necesario instalar unos componentes diferentes, en mi caso está instalado la plataforma JBOSS-7.3 y durante el despliegue los módulos estaba instalando módulos para la plataforma JBOSS-7.1.

```
<switch value="${cmx.server.masterdatabase.type}" caseinsensitive="true">
    <case value="ORACLE">
        <registerJbossJdbcDriver drivername="com.activevos" port="${cmx.jboss7.management.port}"
                                 driverXaDatasourceClass="oracle.jdbc.OracleDriver"/>
    </case>
```

Fig. 51 Este módulo solo sirve para JBOSS7.1

```
<case value="ORACLE">
        <getJBossVersion host="localhost" port="${cmx.jboss7.management.port}" propertyname="jboss.version"/>
        <if>
            <or>
                <equals arg1="${jboss.version}" arg2="7.2"/>
                <equals arg1="${jboss.version}" arg2="7.3"/>
            </or>
            <then>
                <registerJbossJdbcDriver drivername="com.activevos" port="${cmx.jboss7.management.port}"
                                         driverXaDatasourceClass="oracle.jdbc.xa.client.OracleXADataSource"/>
            </then>
            <else>
                <registerJbossJdbcDriver drivername="com.activevos" port="${cmx.jboss7.management.port}"
                                         driverXaDatasourceClass="oracle.jdbc.OracleDriver"/>
            </else>
        </if>
    </case>
```

Fig. 52 Instalación de los módulos para JBOSS 7.1, 7.2 y 7.3

En la Fig, 53 se puede observar que en función de si es JBOSS-7.2 o JBOSS-7.3, instala el módulo "*oracle.jdbc.xa.client.OracleXADataSource*", sin embargo para JBOSS-7.1 es necesario "*oracle.jdbc.OracelDriver*".

Así que hay que modificar el fragmento del fichero en el que se elige el módulo a instalar, y sustituir esas línea del fichero *build.xml*.

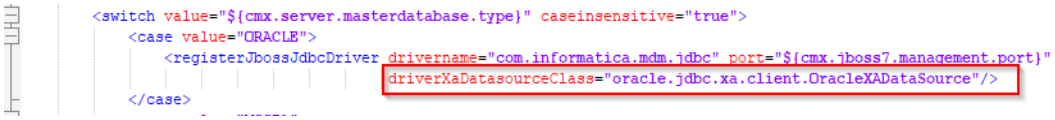

Fig. 53 Nuevo Driver para JBOSS 7.3

Tras esto si se vuelve a lanzar el postInstall con los mismos datos para desplegar ActiveVos y tras unos minutos, se desplegó todo correctamente y sin errores, esto se puede comprobar accediendo a la carpeta *deploys* de JBOSS, que aparecen las aplicaciones desplegadas

Otra manera de comprobar que la aplicación se está desplegando de manera correcta es revisar la carpeta de *logs* que hay en el directorio de instalación.

En Ansible lo que se hizo fue tomar ese fichero **build.xml** reparado y se usará para sobreescribir ese mismo fichero en el resto de instalaciones.

```
- name: copiar el fichero build.xml correcto
  copy:
    src: build.xml
    dest: "{{ hub_directory }}/bin"
    owner: "{{ user_id }}"
    group: "{{ user_group }}"
```

5. Validación de la instalación

Estas pruebas han sido ejecutadas en un entorno "*Slave*" totalmente nuevo y limpio, y ya con todos los cambios que han habido que hacer para poder probar que se despliega y ejecuta correctamente todos los componentes de la aplicación.

Para validar que funcionaba correctamente, primero se accede a la url http://localhost:8080/cmx y desde ahí descargar un archivo .jnlp (Java Network Launching Protocol).

Fig. 54 Página de descargar del HUB de Informatica MDM

Este tipo de extensión se encuentran bastante en desuso al intentar abrir ese archivo mediante el comando **javaws -jnlp siperian_console.jnlp** aparece el siguiente error:

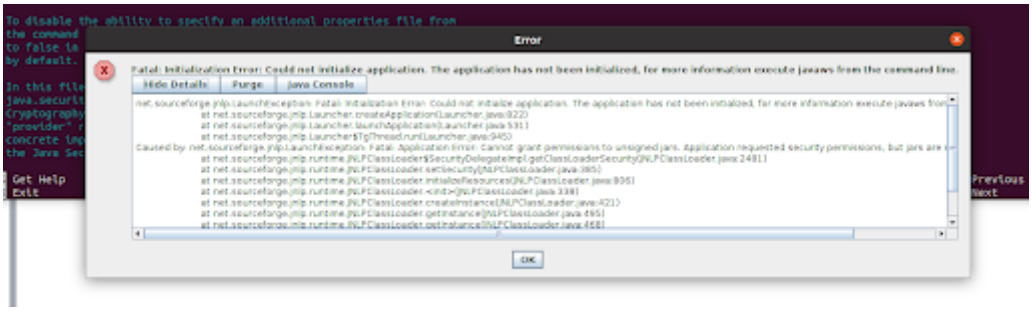

Fig. 55 Error al lanzar siperian_console.jnlp

Esto ocurre por políticas de seguridad de Java, que de forma nativa bloquea los ejecutables que usan una serie de algoritmos de firma. En concreto esta aplicación usa el **MD5** (Message Digest 5), para sortear este problema se debe cambiar una línea del fichero */usr/lib/jvm/zulu8/jre/lib/security/java.security*

Esto se hace desde un comando con Ansible.

```
    - name: modificar política de Java para poder lanzar Informatica hub
      lineinfile:
        state: present
        path: "{{ java_security }}/java.security"
        regexp: "{{ item.regexp }}"
        line: "{{ item.line }}"
      with_items:
        - {regexp: '^jdk.jar.disabledAlgorithms=MD2,', line: 'jdk.jar.disabledAlgorithms=MD2,
RSA keySize < 1024, \'}
```

Ahora sí, se puede continuar validando que todo está desplegado y funcional

Para poder comprobar todas las funcionalidades hay que registrar la base de datos *cmx_ors* en el HUB.

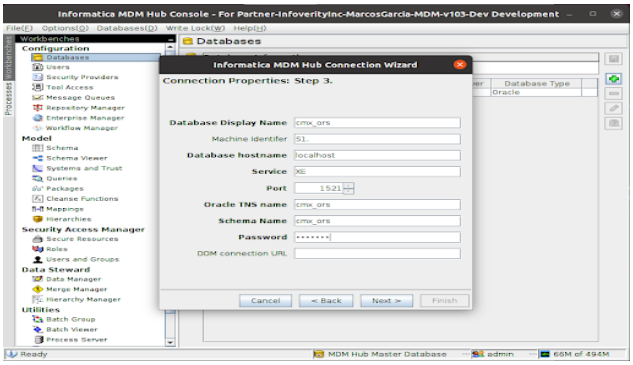

Fig. 56 Registro de la ORS

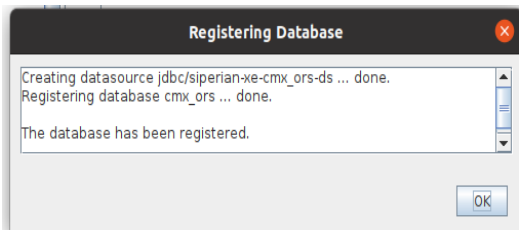

Fig. 57 Registro completado correctamente

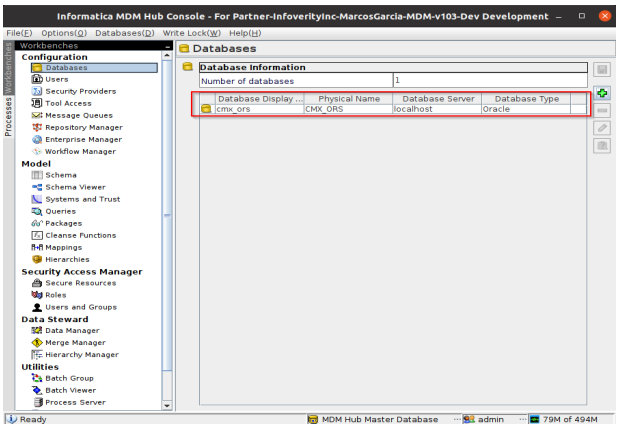

Fig. 58 Bases de datos registradas

Y también registrar el servidor de procesos:

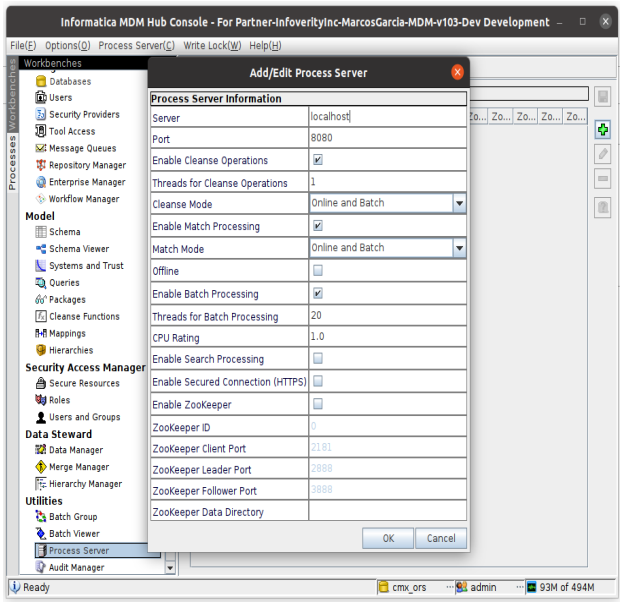

Fig. 59 Registrando el servidor de procesos

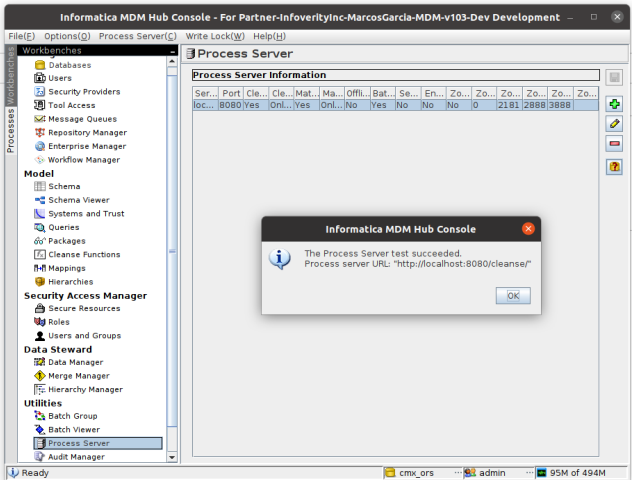

Fig. 60 Test de conexión al servidor de proceso se completa con éxito

Una vez hecho esto se puede comprobar las diferentes herramientas que tiene la aplicación y comprobar que todo funciona correctamente.

Para otra versión, tanto este paso para el registro del servidor de procesos, como el registro de la ORS, se puede hacer mediante una consulta a la base de datos *cmx_system*, y añadir una fila a las tablas que registran las bases de datos y servidores de procesos enlazados en la aplicación.

De forma que desde Ansible se pueda lanzar esa consulta, y automatizar más toda la instalación.

Provisioning

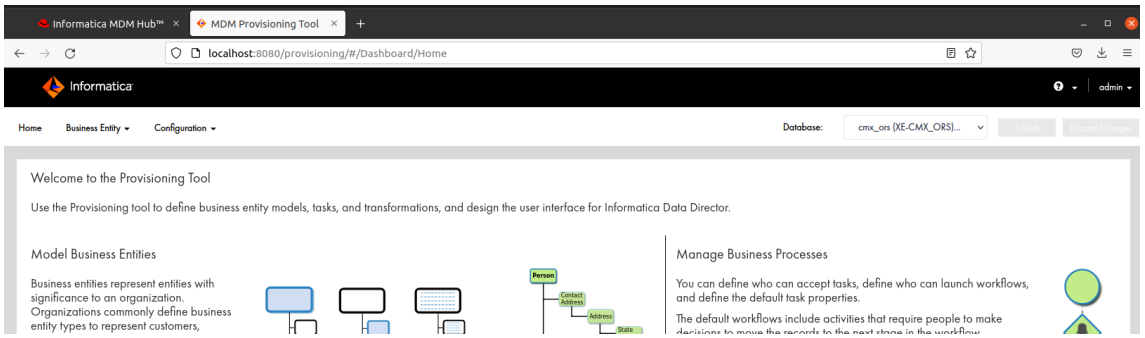

Fig. 61 Acceso al provisioning

ActiveVos Console

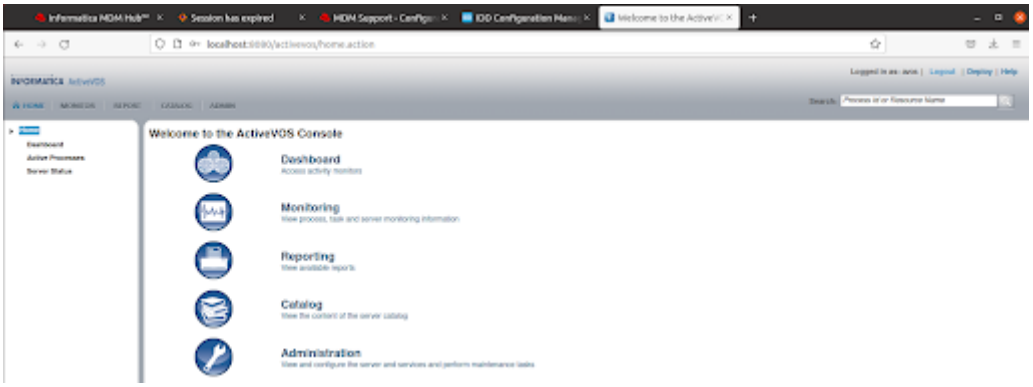

Fig. 62 Acceso al ActiveVos console

Mdmsupport tool

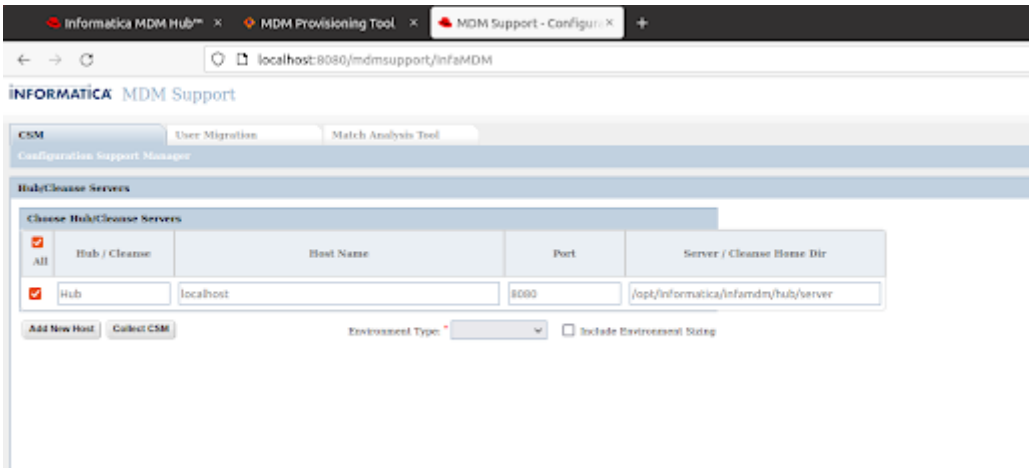

Fig. 63 Acceso a la herramienta de migración de usuarios

Esta herramienta solo está disponible al instalar y desplegar el ResourceKit.

En algunas páginas web aparecen mensajes de precaución ya que proviene de una página que no tiene certificado SSL, o que es un certificado autofirmado. Esto último se puede ignorar, no bloquea ningún aspecto de la aplicación y si se le indica que la página es confiable, la aplicación nos permite continuar sin problema.

Si que es cierto que en cierto momento del proyecto es posible añadirle certificados SSL firmados por una entidad certificadora para habilitar un canal de comunicación más seguro, pero actualmente para el propósito de la aplicación no es necesario.

La aplicación se ejecuta perfectamente, y se puede también acceder a todos los componentes, a través del HUB y del navegador.

Dado que los proyectos proyectos de ActiveVos se despliegan correctamente, y se puede acceder y utilizar todos los componentes, lanzar consultar y trabajos, se puede extrapolar que, la aplicación ha sido correctamente instalada, y la automatización de la misma ha sido completada.

De forma que se ha conseguido instalar automáticamente Informatica MDM con todos su complementos necesarios para completar su instalación rápidamente, de forma segura y fiable, usando una tecnología de *IaC (Infrastructure as Code)*, y haber dejado el sistema listo para que un empleado pueda acceder y realizar la formación o prueba que requiera.

Gracias a esto es posible escalar este proyecto en un futuro, aumentando su alcance, o aplicando la misma metodología para instalar otras aplicaciones relacionadas.

6. Conclusiones y trabajos futuros

A lo largo del proyecto se ha planteado la necesidad de automatizar el despliegue de la instalación, para agilizar todo el proceso de formación de nuevos empleados, también se ha planteado diferentes formas de atacar este problema, y tecnologías relacionadas con los procesos de lanzamiento de máquinas cloud.

También se han tenido que superar muchos obstáculos que han aparecido durante el desarrollo de la aplicación, ya pueda ser cuando el SO no encontraba el comando sqlplus, o cuando alguna aplicación no se desplegaba correctamente debido a que no instalaba los módulos que eran necesarios, y ha exigido aprender de esos problemas, y lo dependiente que pueden llegar a ser las aplicaciones con las variables del sistema..

Ha permitido explorar en profundidad el despliegue de una aplicación muy completa, con componentes más antiguos y otros más modernos, que han ido añadiendo capas de complejidad y funcionalidades nuevas sobre otras más viejas, la variedad de lenguajes, motores y aplicaciones intermedias que requiere un proyecto trabajando entre sí, y lo **necesario que es un equipo de Ingeniero de Computadores** para hacer que todo estos componentes funcionen correctamente.

En resumen, esta aplicación, al igual que otras, depende de tantas otras aplicaciones de Middleware como JMS, JBOSS, tecnología de sistemas distribuidos como CORBA, que resultar necesario a un técnico que tenga suficientes conocimientos de todas estas tecnologías, y que tenga capacidad de resolución de errores, para entender qué es lo que falla.

Para este campo, resulta más necesario un profesional muy polifacético, con un abanico de conocimientos técnicos amplio, y no tan profundo, para entender cada componentes por separado y como conjunto, que una persona muy experimentada en el comportamiento de solo unos de los componentes de forma individual.

También es importante saber cómo funciona un Sistema Operativo, porque cuando se está desarrollando, se tienen en cuenta ciertos fallos dentro de un entorno, pero no se puede tener en cuenta, que falte cierta librería, o que la librería a usar sea diferente para cada versión, por eso entiendo el valor que tiene una automatización fiable de esta aplicación.

Consigue exportar un proyecto completísimo, que abarca el desarrollo de soluciones en Big Data y Data Quality, gestión masiva de datos en distintos dominios, de manera satisfactoria, es un avance y una optimización del proceso de despliegue y formación muy grande.

Y lo más importante, es dar los primeros pasos para otros proyectos que permitan la automatización y el despliegue de sistemas de formación usando esta tecnología, con otras aplicaciones, e incluso llegar a poder ofrecer este tipo de soluciones a otra empresas de forma comercial.

Para este proyecto todavía ahí muchos cambios que se han ido recopilando para trabajos futuros:

- Paralelizar partes del proceso, como puede ser la transferencia de los archivos para la instalación, o distintas etapas de la instalación que no sean dependientes entre sí.
- Generar una UI para solicitar datos al usuario, de forma que se pueda personalizar los comandos que se lanzan a través de Ansible desde las respuestas sobre la misma.
- Explorar la posibilidad de usar scripts en Python para complementar las funcionalidades de la instalación y agilizar ciertos pasos.
- Mejorar la red interna entre las máquinas, habilitando un servidor DHCP que implemente una batería de direcciones para los clientes, y que se ejecute esta instalación sobre todas las IPs que estén activas, y una dirección estática permanente.
- También para mejorar la seguridad interna, siendo que el host desde el que se lanza Ansible será siempre el mismo, crear otra capa de seguridad basada en certificados autofirmados, que verifiquen que las interacciones se hacen desde ese sistema principal
- Crear un servicio de arranque del producto a instalar en Ubuntu, manejable desde **systemctl** [28]o algún otro comando

[28] Systemctl es un comando que se instala con el paquete systemd que gestiona los servicios o aplicaciones y los monitoriza.

7. Bibliografía

[1] *Máquina virtual* A. Silberschatz, J. Peterson, P.Galvin - Fundamentos de Sistemas Operativos - Conceptos fundamentales. pg. 78 cap 2.8 **Máquinas virtuales** [Online,Mayo de 2022] disponible en: http://biblioteca.univalle.edu.ni/files/original/2466e8c69e588a8a615cc22fc71d883e4b930b37.pdf

[2] Fuente:

https://medium.com/@praveenraghav01/aws-clone-copy-ami-across-accounts-183e2b311063

[3] Documentación relacionada con los módulos utilizados:

https://docs.ansible.com/ansible/latest/collections/ansible/builtin/

[4] Fuente:

https://intellipaat.com/mediaFiles/2018/12/8.png

[5] Documentación de instalación de Informatica MDM:

https://docs.informatica.com/master-data-management/multidomain-mdm/10-3/installation-guide-for--oracle-database-with-red-hat-jboss/

[6] L Hochstein, R Moser - 2017 - Ansible: Up and Running: Automating configuration management and deployment the easy way

https://books.google.es/books?hl=es&lr=&id=h5YtDwAAQBAJ&oi=fnd&pg=PP1&dq=ansible+deployment&ots=MKrb2nUPVN&sig=DkN-sSW-8WLp4K5z2_X1Kw8FQ3Y

[7] Y Brikman - 2019 - Terraform: up & running: writing infrastructure as code

https://books.google.es/books?hl=es&lr=&id=57ytDwAAQBAJ&oi=fnd&pg=PP1&dq=terraform+ansible&ots=BlH1uUY2R8&sig=U9Tnjp4-GwCg7nZq168E7Z_8zWE

[8] David Cierco - 2016 - Cloud computing: retos y oportunidades

https://books.google.es/books?hl=es&lr=&id=_fTJXVjOD90C&oi=fnd&pg=PA5&dq=cloud+computing&ots=6IMIQpD4mJ&sig=4YJF9kybIzT8xB3Q3AUvMGBnnw8

[9] Kief Morris - 2020 - Infrastructure as Code: Managing Servers in the Cloud

https://books.google.com/books?id=BIhRDAAAQBAJ&printsec=frontcover&dq=Infrastructure+as+Code&hl=en&sa=X&ved=2ahUKEwiGh4zN1IP6AhWygf0HHU5kC00Q6AF6BAgEEAI

8. Anexo código fuente

En este apartado se recoge la parte del código más importante para el proyecto.

full_install.yml:

```
---
- name: Instalación y despliegue de Informatica MDM 10.3
  hosts: ansible_client
  become: yes
  roles:
       - java
       - jboss
       - jboss-conf
       - install_oracledb
       - create_dbs
       - install_informatica
```

java/main.yml:

```
---
# tasks file for roles/java
       - name: Actualizo apt
       apt:
       update_cache: true
       force_apt_get: true
       upgrade: dist
       ignore_errors: yes

       - name: instalar paquetes necesarios
       apt:
       name: gnupg, curl, unzip, hostname,libaio1, unixodbc, icedtea-netx

       - name: Añado una "key" al repositorio
       ansible.builtin.apt_key:
       keyserver: hkp://keyserver.ubuntu.com:80
       id: 0XB1998361219BD9C9

       - name: creo el directorio Java
       file:
       path: "{{ JAVA_HOME }}"
       state: directory

       - name: Descargar paquete .deb de AzulZulu8
       get_url:
       url: https://cdn.azul.com/zulu/bin/{{ zulu_package }}
       dest: "{{ JAVA_HOME }}"

       - name: Instalo el paquete .deb para añadir azulzulu8 a la lista de
paquetes de descargas disponibles
       apt:
```

```
      deb: "{{ JAVA_HOME }}/{{ zulu_package }}"

      - name: Actualizo apt de nuevo
      apt:
      update_cache: true
      force_apt_get: true
      upgrade: dist
      ignore_errors: yes


      - name: Instalo azulzulu8
      apt:
      name: "{{ java_jre }}"
```

jboss/main.yml:

```
---
      - name: Limpiar directorio eap_home
      file:
      state: absent
      path: "{{ eap_home }}/"

      - name: descomprimir EAP
      unarchive:
      src: "{{ eap_archive }}"
      dest: "{{ eap_deploy_dir }}"
      owner: "{{ jboss_eap_user }}"
      group: "{{ jboss_eap_group }}"

      - name: Crear enlace directo
      file:
      src: "{{eap_home}}"
      dest: "/opt/Informatica/jboss-eap"
      state: link

      - name: Modificar el fichero de configuración JBoss EAP
      lineinfile:
      state: present
      path: "{{ eap_home }}/bin/init.d/jboss-eap.conf"
      regexp: "{{ item.regexp }}"
      line: "{{ item.line }}"
      with_items:
      - {regexp: "^# JBOSS_USER=", line: "JBOSS_USER=jboss-eap"}
      -          {regexp:          "^#          JBOSS_CONFIG=",          line:
"JBOSS_CONFIG=standalone-full.xml"}
      -   {regexp:   "^#   JBOSS_OPTS=",   line:   "JBOSS_OPTS=\"b   0.0.0.0
-Djboss.as.management.blocking.timeout=5000 &"}

      - name: Reemplazar el fichero standalone.conf
      copy:
      src: standalone.conf
      dest: /opt/Informatica/jboss-eap/bin/standalone.conf
      owner: "{{ user_id }}"
      group: "{{ user_group }}"
```

```
      - name: Launch Jboss Server
      become_user: "{{ jboss_eap_user }}"
      shell:    "nohup    {{    eap_home    }}/bin/standalone.sh    -c
standalone-full.xml  -b  0.0.0.0  -Djboss.as.management.blocking.timeout=5000
&"
```

jboss-conf/main.yml:

```
---
# tasks file for jboss-conf
      - name: Copy the script.cli batch file to the client system
      copy:
      src: script.cli
      dest: "{{ eap_home }}/bin"
      owner: "{{ jboss_eap_user }}"
      group: "{{ jboss_eap_group }}"

      - name: Wait for Jboss to be complete deployed
      pause:
      seconds: 30

      - name: Launch the configuration commands for Jboss
      become: yes
      shell:   "{{   eap_home   }}/bin/jboss-cli.sh   --file={{   eap_home
}}/bin/script.cli"
      ignore_errors: yes

      - name: reload the jboss-cli
      become: yes
      shell: "{{ eap_home }}/bin/jboss-cli.sh --connect --command=reload"

      - name: Make owner jboss_eap_user
      become: yes
      ansible.builtin.file:
      path: "{{ eap_home }}"
      state: directory
      recurse: yes
      owner: "{{ jboss_eap_user }}"
      group: "{{ jboss_eap_group }}"

      - name: Launch Jboss Server
      become_user: "{{ jboss_eap_user }}"
      shell:    "nohup    {{    eap_home    }}/bin/standalone.sh    -c
standalone-full.xml  -b  0.0.0.0  -Djboss.as.management.blocking.timeout=5000
&"

      - name: Add Management User
      shell: "{{ eap_home }}/bin/add-user.sh -a -u 'avos' -p 'avos'"
```

install_oracledb/main.yml:

```
---
# tasks file for install_oracledb

      - name: Create Oracle Group
      group:
      name: "{{ oracle_group }}"
      system: yes
      state: present
      gid: "{{ oracle_group_gid | default('401') }}"

      - name: Create Oracle User
      user:
      name: "{{ oracle_user }}"
      comment: "Oracle User"
      uid: "{{ oracle_user_uid | default('401') }}"
      group: "{{ oracle_group }}"
      home: "{{ oracle_dir }}"
      password: oracle

      # Asegurare que no hay nada
      - name: Clean Oracle Home directory
      file:
      state: absent
      path: "{{ oracle_home }}/"

      - name: Crear directorio home de oracle
      file:
      path: /home/oracle
      state: directory
      mode: '0755'

      - name: Unarchive Oracle
      unarchive:
      src: "{{ oracle_archive }}"
      dest: "{{ oracle_deploy_dir }}"
      owner: "{{ oracle_user }}"
      group: "{{ oracle_group }}"

      - name: Change file ownership, group, and of Oracle Home
      file:
      path: "{{ oracle_deploy_dir }}"
      owner: "{{ oracle_user }}"
      group: "{{ oracle_group }}"

      - name: Copy the chkconfig file to the client system
      copy:
      src: chkconfig
      dest: /sbin/
      owner: root
      group: root
      mode: '777'

      - name: Copy the 60-oracle.conf file to the client system
      copy:
      src: 60-oracle.conf
      dest: /etc/sysctl.d/
```

```
      owner: root
      group: root
      mode: '777'


      - name: Start service procps
      service:
      name: procps
      state: started

      - name: Copy the s01shm_load file to the client system
      copy:
      src: S01shm_load
      dest: /etc/rc2.d/
      owner: root
      group: root
      mode: '777'

      - name: reboot host to apply changes
      reboot:
      msg: "Reboot initiated by Ansible"
      connect_timeout: 5
      reboot_timeout: 600
      pre_reboot_delay: 0
      post_reboot_delay: 30
      test_command: whoami

      - name: install oracle xe 11.2
      command:    dpkg    --install    "{{    oracle_home    }}"/Disk1/"{{
oracle_archive_deb }}"

      - name: copiar el fichero con las entradas de configuración
      copy:
      src: responses.txt
      dest: /etc/init.d/
      owner: usuario
      group: usuario

      - name: ejecutar configuracion
      become: yes
      shell: /etc/init.d/oracle-xe configure < /etc/init.d/responses.txt
      ignore_errors: yes

      - name: actualizar bashrc
      blockinfile:
      path: /etc/profile
      insertafter: EOF
      block: |
      export
DISTR_CLASSPATH=$SIP_HOME/lib/ant.jar:$SIP_HOME/lib/ojdbc7.jar:$SIP_HOME/li
b/xml-apis.jar:$SIP_HOME/lib/xercesImpl.jar:$SIP_HOME/lib/log4j-1.24j-1.2.1
6.jar:$SIP_HOME/lib/siperian-common.jar:$SIP_HOME/lib/ant-apache-log4e-log4
j.jar:$JAVA_HOME/lib/rt.jar:$JAVA_HOME/lib/i18n.jar
      export ORACLE_HOME=/u01/app/oracle/product/11.2.0/xe
      export JAVA_HOME=/usr/lib/jvm/zulu8
      export ORACLE_SID=XE
      export ORACLE_BASE=/u01/app/oracle
      export LD_LIBRARY_PATH=$ORACLE_HOME/lib:$LD_LIBRARY_PATH
```

```
      export PATH=$ORACLE_HOME/bin:$PATH

      - name: Lanzar servicio oracle-xe
      command: service oracle-xe start
```

create_db/main.yml:

```
---
# tasks file for create_dbs

      - name: crear directorio de instalación installer
      file:
      path: "{{ binary_directory }}"
      state: directory
      owner: "{{ user_id }}"
      group: "{{ user_group }}"

      - name: crear directorio de instalación installer/avos
      file:
      path: "{{ binary_directory }}/avos"
      state: directory
      owner: "{{ user_id }}"
      group: "{{ user_group }}"

      - name: copiar binarios de instalación de MDM
      unarchive:
      src: "{{ mdm_install_bin }}"
      dest: "{{ binary_directory }}"
      owner: "{{ user_id }}"
      group: "{{ user_group }}"

      - name: copiar binarios para instalar los esquemas de bases de datos
      unarchive:
      src: "{{ mdm_database_bin  }}"
      dest: "{{ binary_directory }}"
      owner: "{{ user_id }}"
      group: "{{ user_group }}"

      - name: copiar los binarios de instalación de Avos
      unarchive:
      src: "{{ avos_bin  }}"
      dest: "{{ binary_directory }}/avos"
      owner: "{{ user_id }}"
      group: "{{ user_group }}"

      -  name:  copiamos  el  script  de  SQL  para  las  tareas  previas  a  la
creación
      copy:
      src: "{{ sqlscript }}"
      dest: "{{ binary_directory }}"
      owner: "{{ user_id }}"
      group: "{{ user_group }}"

      - name: añadir permisos de ejecución a los binarios del instalador
```

```
      command: chmod a+x -R "{{ binary_directory }}"

      - name: ejecutar sqlplus
      command: sudo -i -u usuario sqlplus sys/oracle as sysdba @{{
binary_directory }}/{{ sqlscript }}

      - name: lanzamos el script sip_ant.sh create_system
      command: sudo -i -u usuario sh -c "cd {{ binary_directory
}}/database/bin; {{ binary_directory }}/database/bin/sip_ant.sh
create_system -Dnoprompt=true -Dcmx.server.masterdatabase.type=oracle
-Dmaster.connectiontype=SERVICE -Dmaster.server=localhost
-Dmaster.port=1521 -Dmaster.sid=XE -Ddba.username=sys -Ddba.password=oracle
-Dcmx_ind_tablespace=CMX_INDX -Dcmx_temp_tablespace=CMX_TEMP
-Dora_temp_tablespace=TEMP -Dmaster.username=cmx_system
-Dmaster.password=cmx_system -Dmaster.service=XE"

      - name: lanzamos el script sip_ant.sh import_system
      command: sudo -i -u usuario sh -c "cd {{ binary_directory
}}/database/bin; {{ binary_directory }}/database/bin/sip_ant.sh
import_system -Dnoprompt=true -Dcmx.server.masterdatabase.type=oracle
-Dmaster.connectiontype=SERVICE -Dmaster.server=localhost
-Dmaster.port=1521 -Dmaster.sid=XE -Ddba.username=sys -Ddba.password=oracle
-Dcmx_ind_tablespace=CMX_INDX -Dcmx_temp_tablespace=CMX_TEMP
-Dora_temp_tablespace=TEMP -Dmaster.username=cmx_system
-Dmaster.password=cmx_system -Dmaster.service=XE"

      - name: lanzamos el script sip_ant.sh create_ors
      command: sudo -i -u usuario sh -c "cd {{ binary_directory
}}/database/bin; {{ binary_directory }}/database/bin/sip_ant.sh create_ors
-Dnoprompt=true -Dors.connectiontype=SERVICE
-Dcmx.server.masterdatabase.type=oracle -Dors.server=localhost
-Dors.port=1521 -Dors.service=XE -Dors.locale=en_US -Ddba.username=sys
-Ddba.password=oracle -Dcmx_ind_tablespace=CMX_INDX
-Dcmx_temp_tablespace=CMX_TEMP -Dora_temp_tablespace=TEMP
-Dors.username=cmx_ors -Dors.password=cmx_ors"

      - name: lanzamos el script sip_ant.sh import_ors
      command: sudo -i -u usuario sh -c "cd {{ binary_directory
}}/database/bin; {{ binary_directory }}/database/bin/sip_ant.sh import_ors
-Dnoprompt=true -Dors.connectiontype=SERVICE
-Dcmx.server.masterdatabase.type=oracle -Dors.server=localhost
-Dors.port=1521 -Dors.service=XE -Dors.locale=en_US -Ddba.username=sys
-Ddba.password=oracle -Dcmx_ind_tablespace=CMX_INDX
-Dcmx_temp_tablespace=CMX_TEMP -Dora_temp_tablespace=TEMP
-Dors.username=cmx_ors -Dors.password=cmx_ors -Dors.granularity=0"

      - name: lanzamos el script sip_ant.sh create_bpm
      command: sudo -i -u usuario sh -c "cd {{ binary_directory
}}/database/bin; {{ binary_directory }}/database/bin/sip_ant.sh create_bpm
-Dnoprompt=true -Dcmx.server.masterdatabase.type=oracle
-Dmaster.connectiontype=SERVICE -Dmaster.server=localhost
-Dmaster.port=1521 -Dmaster.service=XE -Dmaster.locale=en_US
-Dba.username=sys -Ddba.password=oracle -Dactivevos.user=avos
-Dactivevos.password=avos -Dactivevos.tablespace=CMX_DATA
-Dactivevos.templatetablespace=TEMP -Dmaster.tnsname=XE"
```

install_informatica/main.yml:

```
---
# tasks file for create_dbs

      - name: copy silentInstallServer.properties file
      copy:
      src: silentInstallServer_sample.properties
      dest: "{{ binary_directory }}/linux/mrmserver"
      owner: "{{ user_id }}"
      group: "{{ user_group }}"

      - name: copy silentInstallResourcekit.properties file
      copy:
      src: silentInstallResourceKit_sample.properties
      dest: "{{ binary_directory }}/linux/mrmresourcekit"
      owner: "{{ user_id }}"
      group: "{{ user_group }}"


      - name: copy MDM database installation files
      copy:
      src: silentInstallCleanse_sample.properties
      dest: "{{ binary_directory }}/linux/mrmcleanse"
      owner: "{{ user_id }}"
      group: "{{ user_group }}"

      - name: copy MDM Silent installation files
      copy:
      src: MDM_v103-Dev_9-8-2021.license
      dest: "{{ binary_directory }}/licenses/"
      owner: "{{ user_id }}"
      group: "{{ user_group }}"

      - name: creamos el directorio de instalación
      file:
      path: "{{ hub_directory }}"
      state: directory
      owner: "{{ user_id }}"
      group: "{{ user_group }}"

      - name: launch jboss server
      become_user: "{{ user_id }}"
      shell:     "nohup     {{     eap_home     }}/bin/standalone.sh     -c
standalone-full.xml  -b  0.0.0.0  -Djboss.as.management.blocking.timeout=5000
&"

      - name: modificar javaws para poder lanzar el hub_console
      lineinfile:
      state: present
      path: "{{ java_security }}/java.security"
      regexp: "{{ item.regexp }}"
      line: "{{ item.line }}"
      with_items:
      -       {regexp:       '^jdk.jar.disabledAlgorithms=MD2,',       line:
'jdk.jar.disabledAlgorithms=MD2, RSA keySize < 1024, \'}
```

```
      - name: Cambiar permisos de forma recursiva
      command: sudo chown -R usuario:usuario /opt/Informatica
      ignore_errors: yes

      - name: ejecutar instalación del servidor MDM
      command:    sudo    -i    -u    usuario    "{{    binary_directory
}}"/linux/mrmserver/hub_install.bin      -f      "{{      binary_directory
}}"/linux/mrmserver/silentInstallServer_sample.properties
      ignore_errors: yes

      - name: ejecutar instalación del cleanse de MDM
      command:    sudo    -i    -u    usuario    "{{    binary_directory
}}"/linux/mrmcleanse/hub_cleanse_install.bin   -f   "{{   binary_directory
}}"/linux/mrmcleanse/silentInstallCleanse_sample.properties
      ignore_errors: yes

      - name: ejecutar instalación del resourcekit de MDM
      command:    sudo    -i    -u    usuario    "{{    binary_directory
}}"/linux/mrmresourcekit/hub_resourcekit_install.bin        -f        "{{
binary_directory
}}"/linux/mrmresourcekit/silentInstallResourceKit_sample.properties
      ignore_errors: yes

      - name: copiar el fichero build.xml correcto
      copy:
      src: build.xml
      dest: "{{ hub_directory }}/bin"
      owner: "{{ user_id }}"
      group: "{{ user_group }}"

      -  name:  ejecutar  el  postInstall  de  Informatica  MDM  HUB  y  de
ActiveVos
      command:    sudo    -i    -u    usuario    "{{    hub_directory
}}"/postInstallSetup.sh -Ddatabase.password=cmx_system -Davos.username=avos
-Davos.password=avos -Davos.jdbc.database.password=avos
      ignore_errors: yes
```

9. Anexo Objetivo Desarrollo Sostenible

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No procede
ODS 1. Fin de la pobreza				X
ODS 2. Hambre cero.				X
ODS 3. Salud y bienestar			X	
ODS 4. Educación de calidad			X	
ODS 5. Igualdad de género				X
ODS 6. Agua limpia y saneamiento				X
ODS 7. Energía asequible y no contaminante				X
ODS 8. Trabajo decente y crecimiento económico		X		
ODS 9. Industria, innovación e infraestructuras	X			
ODS 10. Reducción de desigualdades				X
ODS 11. Ciudades y comunidades sostenibles				X
ODS 12. Producción y consumo responsables				X
ODS 13. Acción por el clima				X
ODS 14. Vida submarina				X
ODS 15. Vida y ecosistemas terrestres				X
ODS 16. Paz, justicia e instituciones sólidas				X
ODS 17. Alianzas para lograr objetivos				X

Se han elegido los siguientes Objetivos de Desarrollo Sostenible porque este proyecto está enfocado en automatizar los despliegues equipos de formación en la nube, avanzando en las herramientas de despliegue de sistemas para futuros proyectos educativos, también, dado que este producto suele tener mucho auge en empresas relacionadas con el sector económico y sanitario, manejando grandes flujos de datos, agiliza su instalación y configuración, permitiendo algo más de sostenibilidad, y en la industria en infraestructuras porque automatiza el despliegue de una Infraestructura (software en este caso), reduciendo tiempos donde se consume electricidad para configurar la aplicación.