DESPLIEGUE DE AUTOMATIZACIONES PARA EL PARCHADO DE SERVIDORES CON ANSIBLE (NEXT)

ALIX ANDREA ANGARITA CASTILLO

UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESCUELA DE INGENIERÍA
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2021

DESPLIEGUE DE AUTOMATIZACIONES PARA EL PARCHADO DE SERVIDORES CON ANSIBLE (NEXT)

ALIX ANDREA ANGARITA CASTILLO
ID: 000244106

DOCENTE SUPERVISOR
ING. LENIN JAVIER SERRANO GIL

SUPERVISOR EMPRESA
ING. JULIAN GUILLERMO MENDOZA LUNA

UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESCUELA DE INGENIERÍA
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2021

AGRADECIMIENTOS

Es importante mencionar que esta práctica no pudo haber sido lo que fue ni se habrían cumplido los objetivos, tanto de este trabajo como los demás propuestos por la empresa, si no hubiera contado con el apoyo y la guía de mi supervisor Julián Mendoza. Fue una muy grata experiencia el tiempo dentro de la empresa gracias al liderazgo de Julián que me permitió tener clara siempre la siguiente meta y el enfoque para resolver cada uno de los retos planteados.

Agradezco también enormemente a Marian Fuentes, mi compañera de práctica, quién con sus capacidades y actitud logró convertirse en una gran aliada para que juntas realizáramos un trabajo satisfactorio a nivel personal y profesional. Fue muy interesante la sinergia con la que cumplimos nuestros objetivos y como, aún con las dificultades de la virtualidad, logramos multiplicar los esfuerzos para proponer soluciones cada vez mejores.

Me complace haber compartido esta experiencia con Julián, Marian, el equipo de sistemas operativos y de I+D, especialmente con Leyder Cerón, Andrés Guillermo Moreno, Miguel Rojas, Jorge Cárdenas, Danny Rubiano, José Daniel Silva, Sebastián Gómez y otros que confiaron en mí y nunca me negaron su ayuda. Les agradezco a todos por su tiempo y apoyo y admiro el gran trabajo que hacen cada día en IBM.

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

RESUMEN GENERAL DE TRABAJO DE GRADO

TITULO: DESPLIEGUE DE AUTOMATIZACIONES PARA EL PARCHADO DE SERVIDORES CON ANSIBLE (NEXT)

AUTOR(ES): Alix Andrea Angarita Castillo

PROGRAMA: Facultad de Ingeniería de Sistemas e Informática

DIRECTOR(A): Lenin Javier Serrano Gil

RESUMEN

Este trabajo es el resultado de la práctica empresarial realizada en la compañía IBM de Colombia como requisito de grado para optar por el título de Ingeniera de sistemas e Informática. IBM Colombia provee servicios de infraestructura tecnológica a múltiples clientes a lo largo del país. Dichos servicios incluyen el aprovisionamiento de servidores que deben ser periódicamente actualizados y sobre los cuales se configuran y despliegan servicios que apoyan el negocio de cada cliente. La aplicación de actualizaciones sobre dichos servidores es una tarea periódica que se ha realizado por años de manera manual. Por ello, surgió la plataforma NEXT, construida sobre Ansible y elaborada por IBM, para automatizar estos procesos de manera que se reduzca el tiempo destinado a estas labores y los administradores de sistemas se enfoquen en tareas que impliquen la toma de decisiones más complejas. En el presente trabajo se trabaja con la plataforma NEXT para validar y mejorar las automatizaciones actualmente disponibles y desarrolladas por el equipo de LA Innovation, principalmente desde IBM Brasil, para adelantar su adopción sobre la infraestructura de los clientes de IBM Colombia. Gracias a esto se desplegaron nuevas versiones de las automatizaciones de parchado para las versiones de Windows Server y Linux Red Hat que se manejan sobre ambientes de desarrollo y ambientes productivos. En estas versiones se hicieron mejoras para aumentar la compatibilidad a servidores Red Hat versión 8 y se añadieron tareas que permiten la ejecución de scripts en cada máquina para detener y arrancar servicios de acuerdo con el proceso de parchado.

PALABRAS CLAVE:

IBM, Automatización, Ansible, Sistemas Operativos, Parchado, Administración de sistemas

V° B° DIRECTOR DE TRABAJO DE GRADO

GENERAL SUMMARY OF WORK OF GRADE

TITLE: DEPLOYMENT OF AUTOMATION FOR SERVERS PATCHING WITH ANSIBLE (NEXT)

AUTHOR(S): Alix Andrea Angarita Castillo

FACULTY: Facultad de Ingeniería de Sistemas e Informática

DIRECTOR: Lenin Javier Serrano Gil

ABSTRACT

This document presents the results of the internship completed in IBM Colombia as a requirement to opt for the degree in Systems and Computer Engineering. IBM Colombia provides technology infrastructure services to multiple clients throughout the country. These include servers provision to configure and deploy services that support clients' businesses and maintenance to keep them updated. Patching these servers is a periodic task that has been done manually for years. For this reason, the NEXT platform, built on Ansible and developed by IBM, was created to automate these processes so the time spent on these tasks is reduced and system administrators can focus on tasks that involve more complex decision-making. In this internship, we worked with the NEXT platform to validate and improve the automations currently available and developed by the LA Innovation team, mainly from IBM Brazil, to forward their adoption over the infrastructure of IBM Colombia clients. As a result, a new release of the patching automation is available for Windows Server and Linux Red Hat systems in development and production environments. This release included improvements to increase the compatibility with Red Hat version 8 servers and new tasks to run scripts on servers to stop and start services according to the patching stage.

KEYWORDS:

IBM, Automation, Ansible, Operating Systems, Patching, Systems Administration

V° B° DIRECTOR OF GRADUATE WORK

INTRODUCCIÓN

En este documento se describe el trabajo realizado durante la práctica empresarial desarrollada en la empresa IBM de Colombia como requisito de grado para optar por el título de Ingeniera de sistemas e Informática. Esta práctica se llevó a cabo dentro de la unidad de negocio de Global Technology Services (GTS), específicamente, en el área de sistemas operativos.

Esta práctica se enfocó en la consecución de algunos de los objetivos planteados para el año 2021 en el área de sistemas operativos, orientados al tema de automatizaciones de tareas rutinarias de administración de sistemas. Esto con el apoyo del área de I+D que se encarga de desarrollar y mantener las automatizaciones para todas las diferentes áreas de GTS.

En la primera parte de este documento se resume la historia de la empresa en Colombia y en el mundo y se describe cómo está organizada, haciendo énfasis en la unidad organizacional que atañe a la práctica.

Se hace entonces una descripción del problema, sus antecedentes y la justificación por la cual se decide realizar el presente trabajo. Con base en esto se plantean unos objetivos a cumplir durante los 6 meses de vinculación a la empresa.

Se toma como referencia unos conceptos clave que se explican brevemente en el marco teórico y se proponen unas actividades y un cronograma de acuerdo con las entregas planeadas por la empresa y la metodología utilizada por el equipo de trabajo.

Para finalizar, se describen los resultados alineados con los objetivos planteados inicialmente y algunas conclusiones y recomendaciones que se recopilaron a partir del desarrollo de este trabajo.

1 GENERALIDADES DE LA EMPRESA

International Business Machines Corporation (IBM) es una multinacional con origen en Estados Unidos y sede central en Armonk, Nueva York [1]. IBM tiene operaciones en 170 países entre los cuales se encuentra Colombia desde hace 84 años, actualmente con sedes en Bogotá y Medellín [2].

La presente práctica se desarrollará en su sede principal en Colombia, ubicada en la Carrera 53 #100-25 en la ciudad de Bogotá [3], pero por motivos de la contingencia sanitaria se inició de forma remota y continuará en esta modalidad hasta nuevas directivas de la empresa y del gobierno.

IBM se enfoca en la producción de hardware como los mainframes y microprocesadores de alto rendimiento y software para áreas como la inteligencia artificial, computación en la nube y computación cuántica. A partir de esto, ofrece servicios de infraestructura de TI, ciberseguridad, entre otros [4]. Además, hace grandes esfuerzos en el área de investigación y desarrollo lo que la ha mantenido como la empresa de Estados Unidos con más patentes por 28 años consecutivos [5].

Gracias a su trayectoria de 109 años y su inversión en investigación, IBM ha sido protagonista de muchos de los grandes avances en la computación [5]. Aunque algunas cosas hayan cambiado y la compañía se haya ido adaptando a las necesidades tecnológicas de la industria hoy sigue siendo uno de los líderes en esta área.

La Big Blue, como también es conocida, cuenta con aproximadamente 350.000 empleados en todo el mundo, de los cuales casi 2000 trabajan en IBM Colombia [6].

1.1. HISTORIA DE LA EMPRESA

La empresa fue fundada en 1911 por Charles F. Flint a partir de la fusión de tres compañías: Hollerith's Tabulating Machine Company, Computing Scale Company of America y la International Time Recording Company, todas precursoras de importantes tecnologías de la época. Esta fusión dio origen a lo que se conocería como la Computing-

Tabulating-Recording Company (CTR) que fabricaba maquinaria para diferentes industrias, incluyendo tabuladores y tarjetas perforadas [7].

Esta nueva compañía fue diversificando sus negocios y gracias a su crecimiento decidió incorporar a Thomas J. Watson como administrador en 1914. Sería entonces Watson quien a través de su experiencia en los negocios sentaría las bases de la identidad corporativa de la compañía, de lo cual se destaca el infundado orgullo y sentido de pertenencia en los trabajadores. Creó también el eslogan THINK que aún representa a la compañía y cambió su rumbo de manera que se enfocara en soluciones empresariales de gran escala en vez de ofrecer productos a pequeños clientes. Durante la administración de Watson la compañía amplió sus operaciones llegando a Europa, Asia, Australia y Suramérica [7].

Años más tarde, tras su gran expansión, el nombre C-T-R empieza a ser muy limitado por lo que se decide cambiarlo formalmente a International Business Machines Corporation (IBM) en 1924 [7].

En la década de los 40s y a causa de la segunda guerra mundial IBM enfoca su investigación en la computación y junto con la Universidad de Harvard construye el Mark I, primer computador electromecánico. Precedente que marca una nueva era en la que IBM construye computadores electrónicos con tubos de vacío y posteriormente con transistores con cada vez mayor capacidad de procesamiento, lo que los hace útiles en todo tipo de aplicaciones. En 1957 también desarrollan el lenguaje de programación FORTRAN [7].

Fue en los 80s cuando IBM ganó mayor renombre al comercializar los primeros computadores personales, que llegaron a todas partes del mundo. Estos por primera vez usaban componentes de otros fabricantes, como fueron los procesadores Intel y el sistema operativo DOS de Microsoft [7].

Después de esta importante etapa la empresa sufrió algunos cambios que le dieron el enfoque orientado a la industria que mantiene hoy. Es a partir de esto que IBM define su propósito, valores y prácticas [8]:

1. Un propósito: *Ser esencial*. Que la compañía sea esencial para sus clientes y que cada IBMer sea esencial para la compañía.
2. Tres valores:
    - Dedicación al éxito de cada cliente.
    - Innovación que importa, para nuestra compañía y para el mundo.
    - Confianza y responsabilidad personal en todas las relaciones.
3. Nueve prácticas:
    - Poner al cliente primero.
    - Escuchar las necesidades e imaginar el futuro.
    - Compartir la experiencia.
    - Reinventar nuestra compañía y a nosotros mismos sin descanso.
    - Atreverse a crear ideas originales.
    - Atesorar los *wild ducks* (Personas con ideas no convencionales, que piensan “fuera de la caja”).
    - Pensar, prepararse, practicar.
    - Unirse para que las cosas se hagan.
    - Mostrar interés personal.

1.1. ORGANIZACIÓN DE LA EMPRESA

IBM Colombia de acuerdo con la estructura organización dispuesta por IBM a nivel global está dividida en diferentes unidades de negocio. Estas unidades cuentan con un líder y además con su propio personal financiero, de recursos humanos, entre otros, de manera que funcione de manera autogestionada. Cada unidad se divide a su vez en líneas de negocio para las cuales hay diferentes gerentes que se enfocan en un tipo de servicio cada uno.

A continuación, se muestra un resumen de esta estructura para la unidad de Global Technology Services (GTS) dentro de la cual se desarrollará esta práctica. Se incluye el

desglose de la línea de negocio de Servicios de Infraestructura (IS) para mostrar la ubicación del cargo.

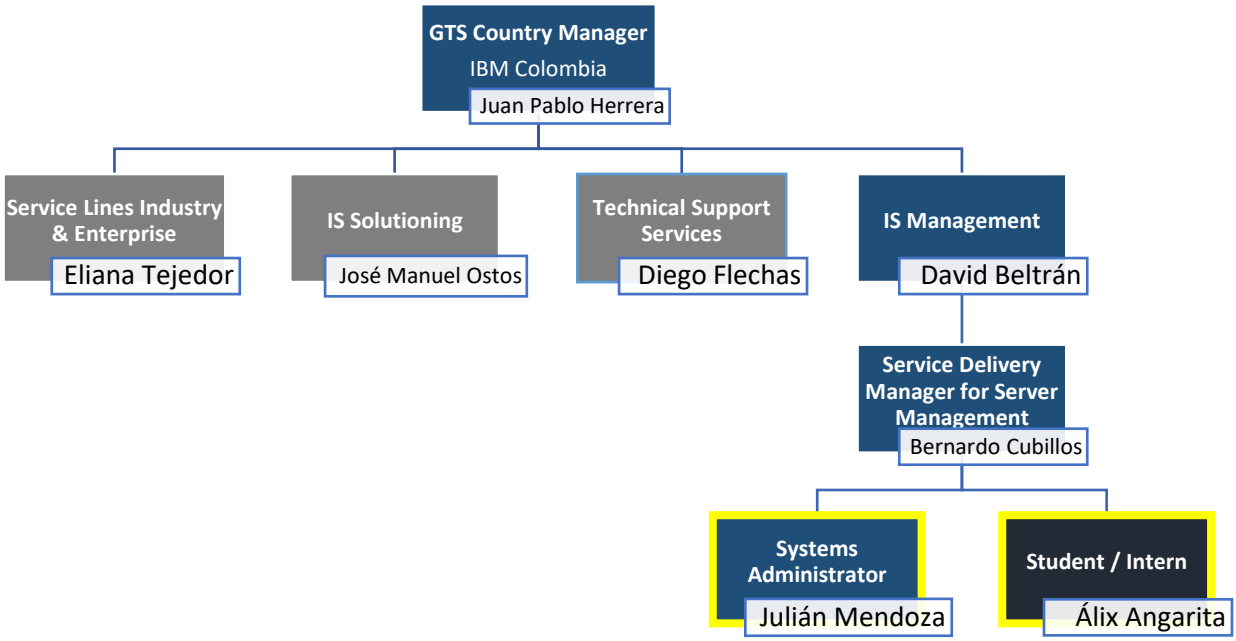


Figura 1. Organigrama con ubicación del cargo en el área de GTS. Fuente: Workday@IBM (Intranet)

Como se puede ver en el diagrama, el gerente asignado es Bernardo Cubillos, quien se encarga de liderar un equipo de administración de servidores dentro del cual hago parte junto con el ingeniero Julián Mendoza, mi supervisor en la empresa.

1.2. DATOS ESPECÍFICOS DEL CARGO EN EL CUAL SE UBICA LA PRÁCTICA EMPRESARIAL

Bajo la supervisión del ingeniero Julián Mendoza se dará apoyo a tareas diarias de administración de sistemas tanto en servidores Windows como Linux, trabajando en paralelo en el desarrollo de Playbooks de automatización de tareas de parchado. Siendo esto último una prioridad del área de Sistemas Operativos, dentro de la línea de negocio de Servicios de Infraestructura.

Estas automatizaciones son específicas de cada sistema operativo, por lo tanto, se trabajará con diferentes equipos de especialistas por sistema, además de personal de las áreas de I+D y Networking.

Este desarrollo se desplegará en una primera etapa sobre servidores de prueba creados y alistados para su administración desde Ansible y una vez se satisfagan los requerimientos para cada sistema se desplegarán en servidores de producción de algunos clientes seleccionados para validar su funcionamiento. Se pretende que el resultado sea aplicable a todos los clientes de IBM Colombia.

2 DEFINICIÓN DEL PROBLEMA

IBM Colombia provee servicios de infraestructura tecnológica a múltiples clientes a lo largo del país. Estos servicios incluyen el aprovisionamiento de servidores que deben ser periódicamente actualizados y sobre los cuales se configuran y se despliegan servicios que apoyan los procesos de negocio de cada cliente.

La aplicación de actualizaciones de seguridad sobre estos servidores es una necesidad constante que hoy se realiza de forma manual. Esto representa un gran número de horas de trabajo requeridas por cliente y puede llevar a errores humanos que desencadenen problemas de inestabilidad sobre los sistemas. Por esto, se ha propuesto utilizar la plataforma NEXT, construida sobre Ansible y desarrollada por IBM, para programar tareas de instalación de actualizaciones que se ejecuten de forma automática y simultánea sobre múltiples servidores.

Durante el desarrollo de esta práctica se pretende desplegar versiones funcionales de las automatizaciones de parchado para cada una de las versiones de sistemas operativos Windows, Linux y AIX que se manejan sobre ambientes de desarrollo y ambientes productivos. Esto mediante un proceso iterativo de ejecución de pruebas, documentación y generación de mejoras, hasta alcanzar los resultados esperados por la empresa.

3 ANTECEDENTES

En la actualidad la gestión de la automatización de la infraestructura administrada por la empresa en Latinoamérica es realizada por el equipo denominado LA INNOVATION, integrado principalmente por empleados de IBM Brasil. Este equipo ha desarrollado, entre otras cosas, la plataforma NEXT. Esta plataforma está construida sobre Red Hat Ansible y permite la ejecución de Playbooks y scripts sobre los servidores de los clientes.

Mediante esta plataforma se automatiza la instalación de parches, la validación de parámetros de seguridad, la gestión de solicitudes de servicio, entre muchas otras tareas de administración y configuración que se deben hacer de forma periódica para todas las máquinas. Estos procesos pueden ser ejecutados de forma manual a través de la plataforma o disparados automáticamente a partir de alertas, condiciones o por fecha.

Estas automatizaciones están disponibles como Playbooks de Ansible que describen una serie de bloques de tareas que se ejecutan de acuerdo con las condiciones y la lógica que se describa en forma de un archivo YAML. Estas tareas se ejecutan en una lista de máquinas que recibe como entrada este archivo y pueden llevar a cabo diferentes acciones dentro de cada una, tales como ejecutar un comando de acuerdo con el sistema operativo o ejecutar un script y esperar por un resultado. Además, es posible usar la salida de estas tareas para enviar a servicios de mensajería instantánea como Slack o generar reportes vía correo electrónico.

Los Playbooks y roles, en los cuales se programan de manera secuencial las tareas que se quieren automatizar, han sido desarrollados principalmente por el equipo en Brasil y están disponibles para que desarrolladores de la empresa contribuyan a su mejora y adaptación según las necesidades específicas por país.

En el caso de Colombia, desde el 2020 se inició la adopción de estos playbooks para la automatización de las tareas ya mencionadas, pero este ha sido un proceso lento ya que deben adaptarse los procesos que ejecutan los administradores de sistemas y capacitar a este personal para que haga uso de la herramienta siguiendo las ventanas de mantenimiento acordadas con cada cliente.

4 JUSTIFICACIÓN

Aunque el equipo de LA Innovation ha proveído mediante NEXT Playbooks para la automatización del parchado de máquinas Windows Server, Red Hat y AIX, estos deben ser probados y desplegados sobre la infraestructura con la que cuentan los clientes de IBM Colombia. Esto, porque debe asegurarse el comportamiento esperado para las versiones de los sistemas operativos administrados.

Debido a esto, para el 2021 el área de sistemas operativos se propuso completar el proceso de adopción de estas automatizaciones aplicando los cambios que fueran necesarios. Este proceso es liderado por el área de I+D para todas las automatizaciones, apoyándose desde el área de Sistemas Operativos para el caso de las automatizaciones de parchado o que están relacionadas con tareas de administración o configuración de sistemas. Con esto se espera reducir el tiempo asignado a este tipo de tareas realizadas en el día a día por los administradores de sistemas.

5 OBJETIVOS

5.1. OBJETIVO GENERAL

Desplegar automatizaciones de parchado sobre servidores Windows, Linux y AIX mediante NEXT para presentar una alternativa al proceso que hoy se hace de manera manual.

5.2. OBJETIVOS ESPECIFICOS

- Evaluar y satisfacer el cumplimiento de los prerrequisitos para el registro de máquinas en la Plataforma NEXT de cada uno de los servidores según su Sistema Operativo.
- Ejecutar y documentar pruebas de las automatizaciones sobre los servidores seleccionados para identificar fallas y aplicar mejoras.
- Desplegar automatizaciones tanto en ambientes de desarrollo como en ambientes productivos.

6 MARCO TEÓRICO

La automatización de infraestructura se realiza mediante tecnologías que permiten ejecutar tareas de administración de hardware, software, sistemas operativos, elementos de red y almacenamiento de datos con una mínima intervención humana. Esto es posible ya que existen procesos repetitivos que pueden ser reemplazados mediante software y que al acumularse producen demoras en las actualizaciones, las ejecuciones de parches y la distribución de recursos [9].

La principal ventaja de usar software para ejecutar instrucciones o procesos repetitivos es que ahorra tiempo y le permite al personal de TI enfocar sus esfuerzos en tareas más satisfactorias o que implican la toma de decisiones más complejas. Además de esto facilita la entrega de servicios con mayor rapidez, cumpliendo con estándares de calidad y seguridad [10].

Algunas de las aplicaciones más comunes de la automatización, aunque sean muchas más y cada vez se sumen más posibilidades, son:

- Automatización de servicios en la nube
- Aprovisionamiento de recursos
- Configuración
- Administración de redes
- Automatizaciones de monitoreo y respuesta a incidentes de seguridad

Debido al constante crecimiento de los servicios de nube y del aumento de la escala en la infraestructura de red cada vez es más recurrente el uso de estas tecnologías y el aumento de su alcance en todo tipo de industria que dependa de servicios de tecnología.

Ansible Red Hat

Una empresa que provee este tipo de servicios y herramientas para la automatización de infraestructura es Red Hat. Esta ofrece productos open-source empresariales compatibles con entornos físicos, virtuales y de nube, entre los que se encuentra Ansible [9].

Ansible es la plataforma de automatización de Red Hat para aprovisionamiento, configuración, administración y despliegue. Esto es realizado mediante archivos llamados playbooks que contienen las instrucciones paso a paso del proceso para la automatización [11]. Estos archivos se escriben en el lenguaje YAML el cual resulta fácil de leer por usar una sintaxis sencilla.

Esta no usa agentes ni ninguna otra infraestructura adicional lo que la hace fácil de desplegar. La forma en que funciona es conectándose a los servidores o nodos y enviando pequeños programas denominados módulos Ansible. Los módulos son como plugins que modelan recursos para llevar el sistema de un estado a otro y existe una variedad de estos que pueden encontrarse como una toolbox de Ansible con más de 750 disponibles [12], o pueden ser construidos desde cero usando cualquier lenguaje de programación, aunque sean más comúnmente escritos en Python.

Los módulos de Ansible se pueden usar desde la línea de comandos o como tareas de Ansible. Estos tienen algunos parámetros configurables y pueden requerir algún parámetro de entrada y al ejecutarse devuelve datos de salida en formato JSON que incluyen detalles de la ejecución. Los argumentos de entrada pueden ser escritos en forma clave=valor o como *complex args* siguiendo la sintaxis YAML [12].

```yaml
- name: restart webserver
  service:
    name: httpd
    state: restarted
```

Figura 2. Ejemplo de una tarea que llama al módulo "service" pasándole los argumentos "name" y "state" [13]

No se requieren servidores, demonios o bases de datos para almacenar los módulos, por lo que para desarrollar playbooks solo es necesario un editor de texto y un terminal desde el cuál ejecutarlos.

Los playbooks se ejecutan desde un nodo de control (debe ser un servidor Unix) en los nodos mediante SSH en el caso de servidores Linux y WinRM en el caso de Windows

Server y para la autenticación se usan llaves SSH con ssh-agent, Kerberos o sistemas de administración de identidad.

Los nodos administrados por el nodo de control se representan en un archivo de texto plano con el inventario donde cada línea es la dirección IP o el nombre de dominio del servidor (si el nodo de control conoce la dirección IP) que pueden estar agrupados por etiquetas puestas entre paréntesis tal como se observa en la Figura 3. Estos grupos son útiles para ejecutar playbooks en solo un grupo específico o para asignar variables que apliquen a todo el grupo y almacenen información necesaria para la ejecución de tareas según el servidor, tales como, sistema operativo, rutas, puertos, mecanismo de autenticación, etc.

```
[webservers]
www1.example.com
www2.example.com

[dbservers]
db0.example.com
db1.example.com
```

```
[atlanta]
host1
host2

[atlanta:vars]
ntp_server=ntp.atlanta.example.com
proxy=proxy.atlanta.example.com
```

Figura 3. Ejemplos de un archivo de inventario (por defecto, /etc/ansible/hosts) [12]

Como se mencionó anteriormente los playbooks de Ansible pueden ejecutar diferentes tareas en un conjunto de máquinas según como se defina en su estructura [14]. Esta estructura está conformada una línea de inicio, seguida por un encabezado con el grupo o nombre de los servidores sobre los cuales se ejecutará y finalmente por un bloque de tareas que corren de forma secuencial. Estas tareas también pueden ser agrupadas como bloques para que se ejecuten de acuerdo con una condición.

```yaml
---
- name: Update web servers
  hosts: webservers
  remote_user: root

  tasks:
  - name: Ensure apache is at the latest version
    ansible.builtin.yum:
      name: httpd
      state: latest
  - name: Write the apache config file
    ansible.builtin.template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf

- name: Update db servers
  hosts: databases
  remote_user: root

  tasks:
  - name: Ensure postgresql is at the latest version
    ansible.builtin.yum:
      name: postgresql
      state: latest
  - name: Ensure that postgresql is started
    ansible.builtin.service:
      name: postgresql
      state: started
```

Figura 4. Ejemplo de un Playbook [14]

Al final de la ejecución del Playbook Ansible retorna un reporte con las tareas ejecutadas de forma exitosa, las fallidas o las que fueron ignoradas en cada máquina [14].

Gestión de parches

La gestión de parches hace referencia al proceso de distribución y aplicación de actualizaciones de software a sistemas operativos aplicaciones o dispositivos para mantenerlos actualizados. Estos parches son creados para corregir errores que se han identificado con el uso del software, ya sea relacionado con funcionalidades o con vulnerabilidades [15].

Ya que este es un proceso que se debe hacer con cierta frecuencia surge la necesidad de automatizarlo. En el caso de los sistemas operativos el parchado se realiza con la

ayuda de aplicaciones que evalúan las versiones actuales del software y lo contrastan con una base de datos de actualizaciones disponibles y aprobadas. Una vez que se tiene una fuente de actualizaciones disponible se puede ejecutar la actualización del sistema de manera manual o con herramientas que realizan este proceso de forma remota y automática en múltiples servidores al mismo tiempo. Una de las herramientas más utilizadas para automatizar este proceso y la cual utiliza IBM en la actualidad es Ansible.

Este proceso difiere de acuerdo con el sistema operativo ya que cada uno utiliza diferentes servidores y estrategias para obtener los parches correspondientes.

En el caso de Windows Server la administración de las actualizaciones es realizada por un servidor de Windows Server Update Services (WSUS) que se conecta co0n las actualizaciones oficiales de Microsoft Update. En este caso los servidores administrados no requieren conexión a internet para acceder directamente a las actualizaciones de Microsoft, sino que a través del servidor de WSUS reciben las actualizaciones que el administrador determine de acuerdo con la configuración y la seguridad de cada máquina [16].

Para los servidores con Red Hat Enterprise Linux se utiliza un servidor denominado Satellite. En este se configuran los repositorios de paquetes de actualizaciones de acuerdo con las versiones del sistema operativo [17].

Finalmente, para los servidores que corren AIX, la distribución propietaria de IBM, la gestión de las actualizaciones se realiza mediante una AIX® Network Installation Management (NIM). Esta red está conformada por un servidor llamado maestro que se configura para proveer paquetes de parches a los demás servidores clientes. La aplicación de actualizaciones puede ser realizada desde el servidor maestro o desde cada uno de los clientes [18].

Con una fuente constantemente actualizada de parches para los servidores se procede a ejecutar las actualizaciones desde un nodo de control de Ansible que ejecuta la tarea correspondiente según el sistema operativo detectado en la máquina.

7 ACTIVIDADES POR DESARROLLAR

Para el cumplimiento de los objetivos planteados se plantea el desarrollo de las siguientes actividades:

1. Validación de prerrequisitos en las máquinas.
2. Gestión con el responsable de la herramienta NEXT.
3. Iteración en el desarrollo de pruebas y aplicación de mejoras.
   3.1. Aprovisionamiento de ambiente de pruebas.
   3.2. Ejecución de pruebas con versión preliminar de Playbooks de automatización.
   3.3. Documentación de resultados de las pruebas de automatización.
   3.4. Generación de mejoras a partir de los resultados obtenidos en las pruebas.
   3.5. Seguimiento a las mejoras propuestas.
   3.6. Pruebas de validación de las mejoras e identificación de oportunidad de nuevas mejoras.

Así, las actividades 1 y 2 que son de alistamiento de máquinas corresponden al primer objetivo de cumplimiento de prerrequisitos y las actividades del numeral 3 que se realizan de manera iterativa contribuyen al cumplimiento de los objetivos de ejecución de pruebas y validación de las automatizaciones sobre entornos de prueba y productivos.

En la tabla a continuación se relacionan estas actividades descritas.

Tabla 1. Actividades propuestas para el desarrollo de la práctica

Objetivo	Actividades
1. Evaluar y satisfacer el cumplimiento de los prerrequisitos para el registro de máquinas en la Plataforma NEXT de cada uno de los servidores según su Sistema Operativo.	1. Validación de prerrequisitos en las máquinas. 2. Gestión con el responsable de la herramienta NEXT.

2. Ejecutar y documentar pruebas de las automatizaciones sobre los servidores seleccionados para identificar fallas y aplicar mejoras.	3.1. Aprovisionamiento de ambiente de pruebas. 3.2. Ejecución de pruebas con versión preliminar de Playbooks de automatización. 3.3. Documentación de resultados de las pruebas de automatización.
3. Desplegar automatizaciones tanto en ambientes de desarrollo como en ambientes productivos.	3.4. Generación de mejoras a partir de los resultados obtenidos en las pruebas. 3.5. Seguimiento a las mejoras propuestas. 3.6. Pruebas de validación de las mejoras e identificación de oportunidad de nuevas mejoras.

8 METODOLOGÍA

Para realizar el parchado de los servidores de forma automatizada se requiere que cada uno de estos cumplan con ciertos requisitos de software, servicios en ejecución, permisos y conectividad que permitan a la plataforma NEXT comunicarse y ejecutar cada una de las instrucciones definidas para este proceso.

Por esto, en una primera etapa durante el desarrollo de esta práctica se buscará una familiarización con cada uno de los sistemas operativos instalados en los servidores de los clientes para, de acuerdo con el procedimiento de alistamiento de máquinas para su registro en NEXT, se satisfagan cada uno de los prerrequisitos exigidos por la plataforma.

Posteriormente, se completarán unos formatos de cargue para hacer la solicitud al equipo de I+D encargado de hacer el registro y seguimiento de las máquinas en NEXT. Una vez se obtenga la aprobación y la confirmación de que los servidores están disponibles se procederá con el proceso iterativo de pruebas.

Cada una de estas actividades se realizarán bajo la supervisión del tutor de la empresa y otros especialistas en infraestructura con un seguimiento semanal al estado del proyecto mediante reuniones donde se asigna tareas y se hace el empalme con los demás equipos según se requiera su apoyo.

La metodología que utiliza la empresa para la planificación de estas actividades es la metodología Agile. Por esto, además de las reuniones semanales para el seguimiento del proyecto, cada equipo realiza reuniones diarias para evaluar si cada persona está haciendo avances en sus asignaciones y en caso negativo identificar las dificultades y problemas que puede estar presentando y que pueden ser resueltas con el apoyo de otros miembros del equipo.

Para aplicar satisfactoriamente esta metodología se usa el marco de trabajo Kanban a través de la herramienta Bluesight. Con esta, se crean los equipos (denominados squads) y se construyen tableros que permiten hacer el seguimiento de las tareas asignadas representadas con tarjetas.

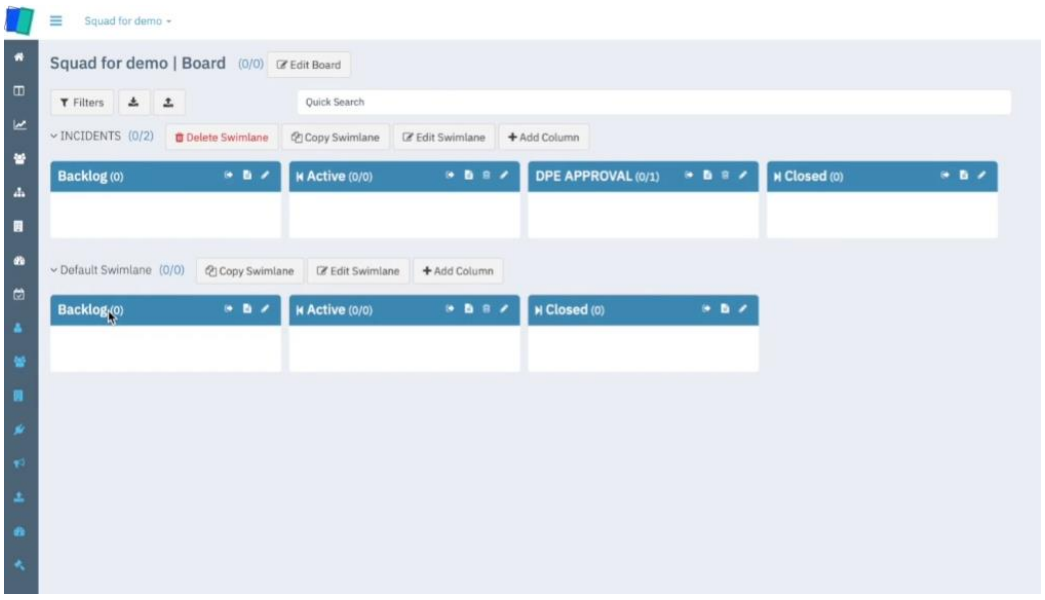


Figura 5. Ejemplo de manejo de tableros en Bluesight

9 CRONOGRAMA DE ACTIVIDADES

La práctica comprende un periodo de 6 meses desde el 4 de enero hasta el 2 de julio del 2021. En este periodo se estima que se desarrollaran las actividades antes descritas según se muestra en la tabla 2.

Tabla 2. Cronograma de actividades

Semana	Fecha inicio	Fecha fin	Actividades
1	Enero 4	Enero 8	Introducción a la empresa y área de GTS
2	Enero 11	Enero 15	Capacitación herramientas equipo de Sistemas operativos
3	Enero 18	Enero 22	Alistamiento y registro de máquinas Windows para ambiente de pruebas
4	Enero 25	Enero 29	Clonación y registro de máquinas Windows para ambiente de pruebas
5	Febrero 1	Febrero 5	Ejecución y análisis de pruebas sobre ambiente Windows
6	Febrero 8	Febrero 12	Corrección y registro de máquinas Windows línea base cliente SDC Bogotá
7	Febrero 15	Febrero 19	Alistamiento y registro de máquinas RHEL para ambiente de pruebas
8	Febrero 22	Febrero 26	Clonación y registro de máquinas RHEL para ambiente de pruebas
9	Marzo 1	Marzo 5	Revisión playbooks de sincronización WSUS
10	Marzo 8	Marzo 12	Ejecución y análisis de pruebas sobre ambiente RHEL
11	Marzo 15	Marzo 19	Corrección y registro de máquinas Linux línea base cliente SDC Bogotá
12	Marzo 22	Marzo 26	Alistamiento y registro de máquinas AIX
13	Marzo 29	Abril 2	Clonación y registro de máquinas AIX para ambiente de pruebas
14	Abril 5	Abril 9	Ejecución y análisis de pruebas sobre ambiente AIX
15	Abril 12	Abril 16	Revisión playbooks de parchado IBM Brasil
16	Abril 19	Abril 23	Generación mejoras playbooks de parchado para sistemas Windows
17	Abril 26	Abril 30	Seguimiento y aplicación de las mejoras propuestas

18	Mayo 3	Mayo 7	Validación de las mejoras sobre ambiente de pruebas máquinas Windows
19	Mayo 10	Mayo 14	Generación mejoras playbooks de parchado para sistemas RHEL
20	Mayo 17	Mayo 21	Seguimiento y aplicación de las mejoras propuestas
21	Mayo 24	Mayo 28	Validación de las mejoras sobre ambiente de pruebas máquinas RHEL
22	Mayo 31	Junio 4	Generación mejoras playbooks de parchado para sistemas AIX
23	Junio 7	Junio 11	Seguimiento y aplicación de las mejoras propuestas
24	Junio 14	Junio 18	Validación de las mejoras sobre ambiente de pruebas máquinas AIX
25	Junio 21	Junio 25	Documentación y puesta en producción de la versión final de los playbooks
26	Junio 28	Julio 2	Socialización equipo de SO sobre aplicación y uso de nuevas automatizaciones

10. RESULTADOS Y DISCUSIÓN

10.1. Evaluación y cumplimiento de los prerrequisitos para el registro de máquinas en la Plataforma NEXT de cada uno de los servidores según su Sistema Operativo

Para cumplir los objetivos planteados, en un primer momento fue necesario configurar un ambiente de pruebas que se asemejara en lo posible a un entorno real de producción. Esto para desarrollar Playbooks que se ejecuten de la misma manera en todos los servidores sin importar la versión de su sistema operativo.

La creación del entorno de pruebas se realizó a partir de un grupo de máquinas virtuales asignadas a las cuales se les instaló diferentes versiones de los sistemas operativos, con estas máquinas se hizo el proceso de alistamiento para su registro en NEXT validando que cumplieran con todos los requisitos solicitados y que son necesarios para ser administrados con Ansible de acuerdo con la guía “Alistamiento y Despliegue NEXT Colombia” de la empresa.

Principalmente estos requisitos son de conectividad entre el servidor a ser administrado y el servidor desde el cuál se dispararán las tareas de administración. Este servidor es conocido como jumphost en el contexto de NEXT y para todos los servidores se utilizan servidores Red Hat como jumphosts, uno para administrar todas las máquinas con Windows Server y otro para todas las máquinas con una distribución de Linux instalada.

Para el caso de los servidores con Windows Server se validó que cumplieran con los siguientes requisitos y se corrigieron en el caso de que no se encontraran:

- Un servidor de WSUS configurado para recibir actualizaciones. Esto se validó desde el servidor de WSUS donde la máquina en cuestión debería aparecer reportando por última vez durante las últimas 24 horas.
- El usuario registrado por defecto en el jumphost de NEXT para el cliente que corresponde con la contraseña correspondiente y que este usuario esté dentro del grupo de administradores de Windows.
- Los puertos 5985 y 5985 abiertos y escuchando.
- El servicio de WinRM configurado en los puertos 5985 y 5986. El puerto 5985 para conexiones HTTP y el puerto 5986 para conexiones HTTPS.
- Que haya conectividad al servidor jumphost correspondiente por los puertos mencionados.
- Powershell 4.0 o superior cuando fuera posible. Para versiones de Windows Server más antiguas la versión más reciente de Powershell compatible. En algunos casos se debe instalar un hotfix para corregir problemas de compatibilidad del Powershell en versiones inferiores a 4.0.

Por otro lado, para servidores Unix (AIX y Red Hat) se cumplió con lo siguiente:

- Un servidor configurado para recibir actualizaciones según la distribución de Linux.

- El usuario registrado por defecto en el jumphost de NEXT para el cliente que corresponde con la contraseña correspondiente y que este usuario esté dentro de un grupo de usuarios llamado autómata.
- El puerto 22 abierto y el servicio sshd activo.
- Que haya conectividad al servidor jumphost correspondiente por el puerto 22.
- Que se incluyan las plantillas sudo en el archivo de configuración *sudoers*. Estas plantillas fueron proporcionadas por la empresa y tienen toda la configuración de los permisos para el usuario que usará el autómata.
- Python 3 instalado. En algunas versiones más antiguas del sistema operativo también se puede utilizar Python 2.7.

Una vez que todos los servidores proporcionados satisfacen estos requisitos se procedió a clonar las máquinas virtuales para tener un número mayor de servidores que constituyeran un ambiente de pruebas más completo y con un número suficiente para identificar y corregir los errores que puedan aparecer durante la etapa de desarrollo de los Playbooks.

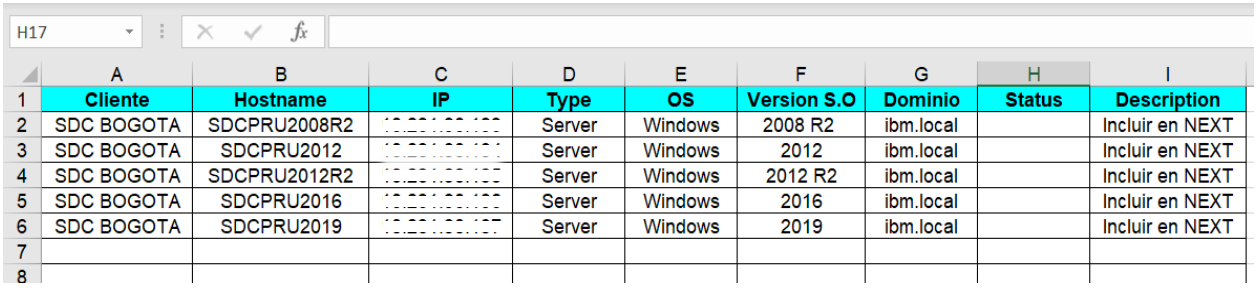
Cliente	Hostname	SO	Dominio	User	Group	Software				Conectividad puertos			Observaciones
						Powershell	Net Fram 4.5	Net Fram Version	WinRM	.3 5985	.3 5986	.4 22	
SDC BOGOTA	SDCPRU2008	Windows	SDC	YES	YES	3,0	TRUE	4.5.1	YES	YES	YES	YES	Admite hasta PowerShell 3.0
SDC BOGOTA	SDCPRU2008R2	Windows	SDC	YES	YES	5,1	TRUE	4.5.1	YES	YES	YES	YES	
SDC BOGOTA	SDCPRU2012	Windows	SDC	YES	YES	5,1	TRUE	4.5.1	YES	YES	YES	YES	
SDC BOGOTA	SDCPRU2012R2	Windows	SDC	YES	YES	4,0	TRUE	4.5.1	YES	YES	YES	YES	
SDC BOGOTA	SDCPRU2016	Windows	SDC	YES	YES	5,1	TRUE	4.6.2	YES	YES	YES	YES	
SDC BOGOTA	SDCPRU2019	Windows	SDC	YES	YES	5,1	TRUE	4.7.2	YES	YES	YES	YES	

Ya cumplía
Instalado
No cumple

Figura 6. Seguimiento a la validación de los prerrequisitos para cada uno de los servidores del ambiente de pruebas

Con el ambiente completo se diligenció un formato de registro de servidores en NEXT donde se relacionó el hostname de cada máquina junto con su dirección IP asignada, versión del sistema operativo y cliente al cual pertenece. Con esta información el personal de I+D incluye la máquina dentro del inventario del jumphost para que éste pueda

administrarla y realiza pruebas de conectividad desde el jumphost hacia el servidor para confirmar que se pueden ejecutar las automatizaciones sobre la máquina. En el caso de los servidores Linux se copia una llave SSH en el servidor que se está registrando antes de establecer la conexión.



	A	B	C	D	E	F	G	H	I
1	**Cliente**	**Hostname**	**IP**	**Type**	**OS**	**Version S.O**	**Dominio**	**Status**	**Description**
2	SDC BOGOTA	SDCPRU2008R2	[illegible]	Server	Windows	2008 R2	ibm.local		Incluir en NEXT
3	SDC BOGOTA	SDCPRU2012	[illegible]	Server	Windows	2012	ibm.local		Incluir en NEXT
4	SDC BOGOTA	SDCPRU2012R2	[illegible]	Server	Windows	2012 R2	ibm.local		Incluir en NEXT
5	SDC BOGOTA	SDCPRU2016	[illegible]	Server	Windows	2016	ibm.local		Incluir en NEXT
6	SDC BOGOTA	SDCPRU2019	[illegible]	Server	Windows	2019	ibm.local		Incluir en NEXT
7									
8									

Figura 7. Formato de cargue de máquinas en NEXT para hacer uso de las automatizaciones

Se construyeron de esta manera dos ambientes de pruebas, uno de servidores Windows Server con un total de 21 máquinas y otro de 21 servidores Linux Red Hat. En el caso de AIX fueron proporcionados 2 servidores de pruebas, pero por cuestión de tiempo no se realizó este proceso para esos dos servidores y por consiguiente no se desarrollaron automatizaciones para este sistema operativo.

En las Tabla 3 y 4 se puede observar cómo están constituidos ambos ambientes de pruebas, los cuales además de permitir el desarrollo de las automatizaciones descritas en este trabajo han sido una gran herramienta para desarrolladores dentro de la empresa para desplegar y probar sus automatizaciones.

Tabla 3. Ambiente de pruebas Windows Server

Número de servidores	Sistema operativo
1	Windows Server 2008 SP1
4	Windows Server 2008 R2
4	Windows Server 2012
4	Windows Server 2012 R2
4	Windows Server 2016
4	Windows Server 2019

Tabla 4. Ambiente de pruebas Red Hat Enterprise Linux

Número de servidores	Sistema operativo
7	Red Hat 6
7	Red Hat 7
7	Red Hat 8

10.2. Ejecución de pruebas de las automatizaciones de parchado sobre los servidores seleccionados para identificar fallas y aplicar mejoras.

Después del registro por parte del equipo de I+D el estado de conexión de estos se puede monitorear diariamente desde un dashboard de Kibana disponible con estadísticas y gráficas de conectividad y cumplimiento de prerrequisitos para los servidores registrados de todos los clientes de Latinoamérica. Esta herramienta es administrada por el equipo de LA Innovation con el apoyo del equipo de I+D de cada país.

Figura 8. Dashboard de Kibana donde se visualiza información de conectividad para los servidores para cada cliente. Fuente: IBM

Con la ayuda de esta herramienta se hizo un seguimiento diario del estado de conexión del ambiente de pruebas para mantenerlo siempre funcionando correctamente para la ejecución de las automatizaciones. Una vez se observó que todos los servidores aparecían con una conexión satisfactoria en esta plataforma se procedió con la revisión de las automatizaciones existentes y la ejecución de pruebas.

10.2.1. Pruebas y mejoras a las automatizaciones para servidores Windows Server

Para comenzar, se ejecutó una prueba para el recién creado ambiente de pruebas de Windows Server con la automatización puesta a disposición por el equipo de LA Innovation como se puede ver en la Figura 9. Esto se realizó mediante la plataforma RBEA, la cual fue construida sobre Ansible para permitir a los administradores de sistemas escoger desde una lista de automatizaciones aquella que desean ejecutar junto con la lista de servidores sobre los cuales se desea ejecutar esta automatización.

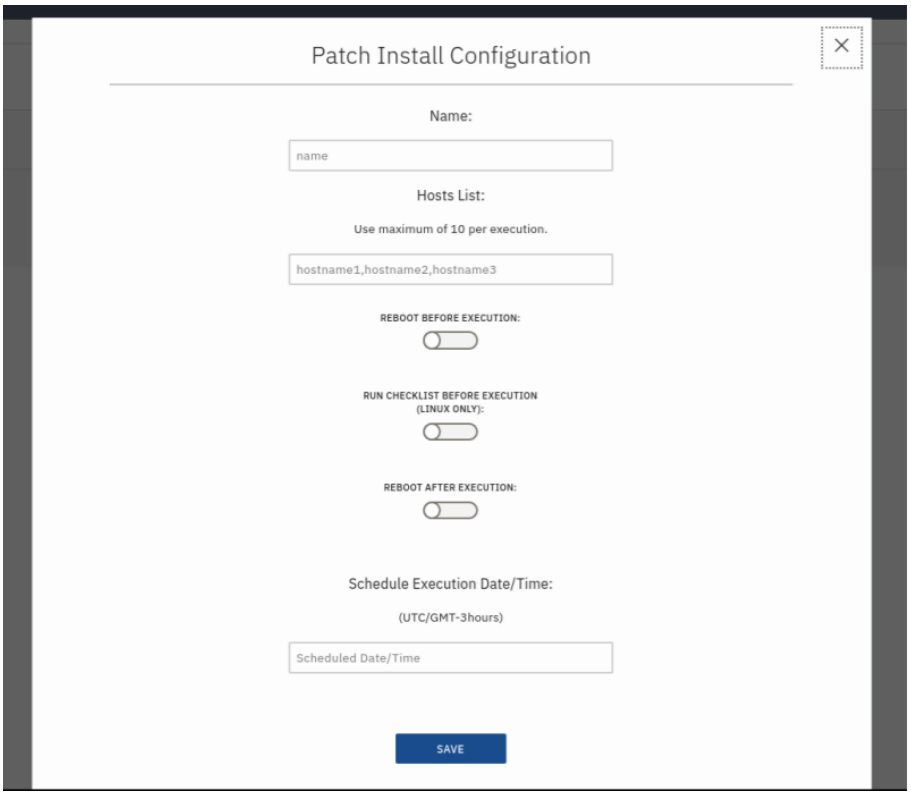


Figura 9. Ejecución de automatizaciones desde RBEA. Fuente: IBM

Los resultados de la ejecución de esta automatización sobre el ambiente de pruebas se pudieron observar desde un canal de Slack que se configuró para recibir los mensajes desde Ansible (Véase Figura 9). En estos se encontró que, aunque para la mayoría de los servidores fue posible realizar el parchado y la instalación de actualizaciones, hubo algunos que no se conectaron a su servidor WSUS para recibir actualizaciones por lo que presentaron algunos errores en la instalación de parches.

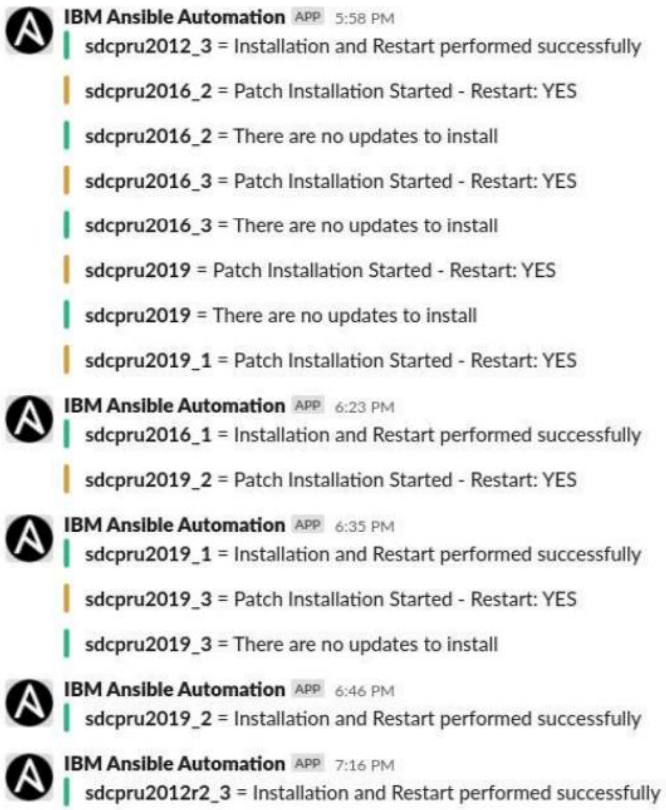


Figura 10. Mensajes en Slack con los reportes del proceso de parchado desde la automatización en Ansible

Por esto, se investigó cada uno de los errores para encontrar sus causas y hacer los ajustes en la automatización si era necesario y se encontró que estos correspondían a un problema de comunicación con su WSUS server, esto porque antes de realizar el parchado la automatización vuelve a asignar la dirección IP del servidor y en algunos casos está asignando la de un servidor que no corresponde.

Luego de encontrar la causa del error e identificar que no podía ser corregida en el código se indicó al equipo de LA Innovation quienes confirmaron que estas direcciones IP se toman a partir de una base de datos que se encontraba desactualizada e hicieron la corrección de acuerdo con la información suministrada.

Con esto se decidió por parte del equipo de sistemas operativos que la automatización de parchado para los servidores Windows Server se ejecutaba satisfactoriamente para las diferentes versiones disponibles para los clientes por lo que no se le hicieron modificaciones.

10.2.2 Pruebas y mejoras a las automatizaciones para servidores Red Hat Enterprise Linux

Para el caso de los servidores RHEL también fue posible, de la misma manera que con los Windows Server, monitorear su estado de conexión desde Kibana para mantener los servidores siempre en línea con su jumphost en Ansible y así ejecutar desde allí las automatizaciones de parchado.

Como se mencionó previamente, para que los servidores Red Hat reciban actualizaciones estos deben configurarse para que al ejecutar comandos para actualizar o instalar paquetes el gestor de paquetes (yum o dnf según la versión del OS) sepa de donde descargar los paquetes solicitados y listar todos los paquetes que apliquen para la versión del sistema que se quiere actualizar. Para el caso de este ambiente de pruebas que no está conectado a internet se hizo mediante un servidor Satellite que funciona de manera offline.

El proceso para configurar un servidor Satellite como un repositorio de paquetes implica tres pasos:

1. Configurar la máquina que se usará como servidor Satellite:

   En esta máquina se creó un directorio en el cual se montó la ISO del sistema operativo que contenía todos los paquetes proveídos por Red Hat para instalar nuevo software

o actualizar los que vienen por defecto en el sistema. Esto se hizo para las tres versiones de Red Hat que se usan en el ambiente de pruebas contando así con un repositorio de actualizaciones para cada una de estas versiones. Adicionalmente se configuró un servidor Apache para exponer estos directorios y así los paquetes pudieran ser descargados desde cada uno de los servidores.

2. Establecer conectividad entre el servidor Satellite y los servidores que recibirán las actualizaciones:

Se validó que hubiera un puente de conexión entre todos los servidores y el servidor Satellite y que el puerto 80 estuviera abierto y escuchando para el caso del servidor Satellite.

3. Configurar el gestor de paquetes en cada una de las máquinas clientes para que al ejecutarlo apunte al servidor Satellite:

En el directorio de configuración del gestor de paquetes yum */etc/yum.repos.d/* se encuentra un archivo que contiene todos los parámetros del repositorio que usa este gestor y que ahora corresponde al servidor Satellite. En este archivo se colocó la dirección IP del servidor Satellite y la ubicación de los paquetes para la versión del SO del servidor. El archivo de configuración resultante es similar a lo que se observa en la Figura 11.

```
# vi /etc/yum.repos.d/rhel7-server.repo
[InstallMedia]
name=Red Hat Enterprise Linux 7.2
mediaid=1446216863.790260
metadata_expire=-1
gpgcheck=0
cost=500
baseurl=file:///media/rhel7-server/
enabled=1
```

Figura 11. Configuración del repositorio para el gestor de paquetes YUM [19]

El primer paso de la etapa de pruebas fue ejecutar el Playbook de parchado sobre el ambiente de pruebas de diferentes versiones de Red Hat usando RBEA y observar finalmente el resultado de ejecución en esa misma plataforma y mediante los mensajes de Slack que se fueron recibiendo en el canal correspondiente a medida que se ejecutaba cada tarea del Playbook.

Los resultados de estas pruebas arrojaron que para los servidores Red Hat versión 6 y 7 el proceso de parchado se realizaba satisfactoriamente, pero en el caso de Red Hat 8 siempre se presentaba un error como el que se muestra en la Figura 12.

Figura 12. Error de parchado para los servidores RHEL8

Este problema se intentó resolver de acuerdo con lo que indicaba el mensaje de error instalando diferentes versiones de Python y usando diferentes paquetes yum y dnf sin ningún éxito por lo que se investigó a mayor profundidad las diferencias entre esta versión de sistema operativo y las versiones 6 y 7 de Red Hat respecto al gestor de paquetes, qué versión de Python utilizaba y qué problemas de incompatibilidad tenía con la versión de Python utilizada por Ansible.

En primer lugar, se instaló Python versión 2.7, ya que Ansible indicaba que se requería el módulo yum para Python 2 y se configuró el intérprete en el sistema de manera que apuntara a la ruta donde se encontraba Python 2.7 (Por defecto Red Hat 8 utiliza Python 3.6). También se revisó que yum estuviera instalado y configurado para funcionar con Python 2. Pese a esto, al volver a ejecutar el Playbook el error de la Figura 12 persistió.

Al indagar la causa de que este error continúe apareciendo, incluso con la especificación del intérprete mencionado, se encuentra que, aunque se ejecute el módulo yum este

archivo apunta en realidad a la carpeta que corresponde al módulo dnf, como se observa en la Fig. 2, el cual sólo funciona con Python 3.

```
[root@SDCPRURHEL8 ~]# readlink -f /etc/yum.conf
/etc/dnf/dnf.conf
[root@SDCPRURHEL8 ~]#
```

Figura 13. Enlace simbólico mediante el cual yum apunta al módulo dnf

Por otro lado, observando en el Playbook las tareas que listan e instalan las actualizaciones se identifica que para listar y actualizar los paquetes se especifica el uso de yum como backend, como se muestra en la Figura 14.

```
---
- block:

    - name: List all packages
      yum:
        list: updates
        update_only: true
        update_cache: true
        use_backend: yum
      become: true
      register: list_update

- name: Update all packages
  yum:
    name: '*'
    state: present
    update_only: true
    update_cache: true
    use_backend: yum
  become: true
  register: update
```

Figura 14. Módulo yum especificado como backend para listado y actualización de paquetes en el Playbook

Esto no es conveniente ya que, como se dijo anteriormente, lo ideal es utilizar el módulo dnf para evitar el error de la figura 12. Consecuentemente, se decide cambiar esto para no forzar el uso de yum y que en vez de esto al ejecutar el módulo yum este apunte al módulo dnf como se espera según la figura 13.

Luego de hacer este cambio se ejecuta el Playbook sin ningún problema aparente. Sin embargo, al verificar el log que muestra las actualizaciones hechas, no se evidencia ningún paquete instalado como se observa en la figura 15.

```
TASK [Update all packages] ****************************************************
ok: [SDCPRURHEL8]

TASK [Show Update Result] *****************************************************
ok: [SDCPRURHEL8] => {
    "msg": {
        "changed": false,
        "failed": false,
        "msg": "Nothing to do",
        "rc": 0,
        "results": [
            "Packages providing * not installed due to update_only specified"
        ]
    }
}
```

Figura 15. Playbook ejecutado exitosamente desde un nodo de control configurado para pruebas. Resultado: ningún paquete instalado

Al realizar varias pruebas se logra determinar que esto sucede ya que en la tarea de actualización de paquetes con el módulo yum (usando dnf en el backend) no reconoce correctamente el wildcard '*' como parámetro para representar todos los paquetes.

Para solucionar esto se decidió reemplazar el asterisco ('*') por un arreglo construido a partir de la lista de paquetes obtenida al ejecutar la tarea que lista los paquetes disponibles, de la cual toma el atributo "name" como nombre del archivo y se organiza en forma de lista. A continuación, se muestra la línea que realiza la construcción del arreglo de paquetes.

```yaml
- name: Update all packages
  yum:
    name: "{{ list_update.results | map(attribute='name') | list }}"
    state: present
    update_only: true
    update_cache: true
  become: true
  register: update
```

Figura 16. Reemplazo de '' por el arreglo o lista que contiene los paquetes a instalar*

Una vez hecho este cambio, se ejecuta la prueba con éxito verificando que todos los paquetes se hayan instalado como se muestra a continuación.

Figura 17. Prueba de parchado exitosa para un servidor RHEL8. Lista de paquetes instalados completamente.

Hasta este punto con las modificaciones realizadas ya fue posible realizar el parchado de las máquinas RHEL 8 de la misma manera que funcionaba para las RHEL 6 y 7. Sin embargo, ya que el gestor de paquetes que instala las actualizaciones cambió para estos servidores también se hace necesario hacer algunas correcciones para hacer el reporte de los resultados del parchado a partir de los logs generados por dnf.

Estos resultados se envían tanto por Slack, como se mencionó anteriormente, como por correo electrónico y son un insumo importante para el administrador de sistemas que le permite conocer qué paquetes se instalaron en cada una de las máquinas y si es necesario instalar algún paquete faltante de manera manual si no fue proveído en la versión requerida por el servidor Satellite.

Teniendo en cuenta que el archivo log donde se guardan los paquetes instalados para ser enviados por mensaje a Slack, para el módulo yum es */var/log/yum.log* y para el dnf es */var/log/dnf.log* se realiza una modificación en el bloque del Playbook donde se evalúa la versión de la máquina y según esto se especifican los archivos de log y el formato correspondientes.

No obstante, luego de realizar pruebas utilizando este archivo para reportar los resultados del parchado se identificó que el gestor de paquetes dnf no siempre actualiza este archivo de logs cuando se ejecuta desde Ansible (Este problema fue identificado por la comunidad y fue abierto un issue al respecto en el repositorio de Ansible).

Por esto y para resolver la necesidad de generar un reporte de los paquetes modificados para enviar por Slack y correo electrónico se usó la salida de la tarea que ejecuta el *yum update*.

```
always:
  # Send Error/Completed to Slack
  - name: Set pkg variable
    shell: grep "$(date +'%b %d')" /var/log/yum.log
    become: true
    register: pkg_inst
    when:
      - update is defined and list_update is defined and update.skipped is not defined
      - pkg_manager == 'yum'

  # Get patch report from RHEL8 hosts. Logs in /var/log/dnf.rpm.log are not always generated.
  - name: Set dnf result message
    set_fact:
      pkg_inst: '{ "stdout": "{{ update.results | join("\n") }}" }'
    when:
      - update is defined and list_update is defined and update.skipped is not defined
      - pkg_manager == 'dnf'
```

Figura 18. Modificación para verificar la versión del S.O. Si es RHEL8 asigna la ruta de archivo log correspondiente con el módulo dnf. Si es menor a esta versión se asigna el archivo log correspondiente con el módulo yum.

Finalmente se verifica que para todas las versiones de Red Hat funcionen correctamente los cambios realizados ejecutando pruebas desde el ambiente Sandbox de NEXT y se observa que tanto por Slack como por correo electrónica se reporta el resultado del parchado correctamente como se muestra en las Figuras 19 y 20.

```
server: sdcprurhel8-1
status: Installation and Restart performed successfully. Packages installed:
Installed: httpd-2.4.37-30.module+el8.3.0+7001+0766b9e7.x86_64
Installed: httpd-filesystem-2.4.37-30.module+el8.3.0+7001+0766b9e7.noarch
Installed: httpd-manual-2.4.37-30.module+el8.3.0+7001+0766b9e7.noarch
Installed: httpd-tools-2.4.37-30.module+el8.3.0+7001+0766b9e7.x86_64
Installed: udisks2-2.9.0-3.el8.x86_64
Installed: udisks2-iscsi-2.9.0-3.el8.x86_64
Installed: udisks2-lvm2-2.9.0-3.el8.x86_64
Installed: unbound-libs-1.7.3-14.el8.x86_64
Installed: vim-common-2:8.0.1763-15.el8.x86_64
Installed: vim-enhanced-2:8.0.1763-15.el8.x86_64
Installed: vim-filesystem-2:8.0.1763-15.el8.noarch
Installed: wget-1.19.5-10.el8.x86_64
```

Figura 19. Prueba de parchado exitosa para RHEL8-1. Se reporta la lista de paquetes instalados con formato arreglado de manera que se pareciera lo más posible a los logs proporcionado por el yum.log para versiones anteriores de Red Hat.

```
server: sdcprurhel7-2
status: Installation and Restart performed successfully. Packages installed:
Apr 21 17:12:18 Updated: libgcc-4.8.5-44.el7.x86_64
Apr 21 17:12:19 Updated: 1:grub2-common-2.02-0.87.el7.noarch
Apr 21 17:12:19 Updated: redhat-release-server-7.9-3.el7.x86_64
Apr 21 17:12:19 Updated: setup-2.8.71-11.el7.noarch
Apr 21 17:12:19 Updated: langtable-0.0.31-4.el7.noarch
Apr 21 17:12:19 Updated: 32:bind-license-9.11.4-26.P2.el7.noarch
Apr 21 17:12:19 Updated: libreport-filesystem-2.1.11-53.el7.x86_64
Apr 21 17:12:19 Updated: tzdata-java-2020a-1.el7.noarch
Apr 21 17:12:19 Updated: langtable-data-0.0.31-4.el7.noarch
Apr 21 17:12:20 Updated: 1:grub2-pc-modules-2.02-0.87.el7.noarch
Apr 21 17:12:20 Updated: 2:vim-filesystem-7.4.629-7.el7.x86_64
Apr 21 17:12:20 Updated: libX11-common-1.6.7-2.el7.noarch
Apr 21 17:12:20 Updated: subscription-manager-rhsm-certificates-1.24.42-1.el7.x86_64
Apr 21 17:12:20 Updated: 1:emacs-filesystem-24.3-23.el7.noarch
Apr 21 17:12:21 Updated: tzdata-2020a-1.el7.noarch
Apr 21 17:12:21 Updated: bash-4.2.46-34.el7.x86_64
```

Figura 20. Prueba de parchado exitosa para RHEL7-2. Se reporta la lista de paquetes instalados tal y como se hacía en la versión anterior de este Playbook.

10.3. Desplegar automatizaciones tanto en ambientes de desarrollo como en ambientes productivos.

Para realizar las pruebas de los cambios realizados sobre el Playbook de parchado se utilizó un nodo de control de Ansible configurado en una de las máquinas del ambiente de pruebas de Red Hat. Para esto, se instaló Ansible y se creó un inventario con algunos servidores Windows y Linux del ambiente de pruebas. Adicionalmente fue necesario instalar algunos paquetes adicionales para la conectividad con los servidores Windows por WinRM y copiar llaves SSH para los servidores Linux.

Desde este nodo de control se ejecutaron los Playbooks como primer filtro para definir los cambios que debían realizarse. Una vez el Playbook se ejecutaba como se esperaba se hacían estos mismos cambios en el repositorio de LA Innovation en una nueva rama y se solicitaba la aprobación del equipo de I+D en Colombia para combinarlos en la rama de sandbox. Con el código disponible en la rama de sandbox, la plataforma sandbox de RBEA se actualizaba automáticamente para reflejar estos cambios e hizo posible ejecutar el Playbook con los cambios de la misma manera que se realiza en RBEA sobre ambientes productivos.

Este proceso se repitió para cada uno de los cambios introducidos en los Playbooks de manera que, una vez se contó con una nueva versión del Playbook probado sobre

diferentes servidores con diferentes condiciones (servidores con y sin actualizaciones, con diferentes versiones de Python, entre otras) se subieron en una rama que pudiera ser combinada en la rama de desarrollo. Esta es la rama por defecto en la que trabajan todo el personal de IBM que contribuye a las automatizaciones y que es continuamente revisada por el equipo de LA Innovation para combinar estos cambios en la rama máster (rama principal).

Todos los cambios descritos en el numeral anterior fueron finalmente subidos a este repositorio en GitHub y aprobados por los desarrolladores de IBM Brasil. Luego de esto fueron combinados a la rama principal del repositorio por lo que quedaron disponibles desde RBEA para ser utilizados por los administradores de sistemas o cualquier persona de IBM Latinoamérica.

Con los cambios disponibles desde RBEA se programaron pruebas sobre ambientes productivos con servidores previamente aprobados por los clientes para este fin y se confirmó el correcto funcionamiento de los Playbooks de parchado.

10.4. Otras actividades realizadas

En paralelo al cumplimiento de los objetivos planteados en el presente trabajo también se apoyaron otras tareas tanto de automatización como de la labor diaria del equipo de sistemas operativos. Entre estas actividades realizadas se destacan las siguientes:

- Desarrollo de tareas de automatización de bajada y subida de servicios previo al parchado del sistema. Esto se realizó, en una primera etapa, con los servicios de SQL Server para evitar afectar a las instancias de bases de datos si no se detenía correctamente un servicio antes de realizar actualizaciones del sistema operativo. Para esto se escribieron tareas en Ansible que ejecutaran scripts en cada servidor de acuerdo con su sistema operativo para detener o arrancar estos servicios según la etapa del parchado en la que encuentre, una vez ejecutados se obtiene la salida de estos scripts para tomar decisiones respecto al flujo de la automatización y en caso de presentarse errores se detiene el proceso de parchado para evitar llevar

el sistema a un estado de inestabilidad. El resultado de ejecución de estos scripts sea exitoso o no se reporta en tiempo real por medio de Slack. Esta actividad implicó escribir los scripts en Bash y Powershell para interactuar con los servicios de manera que la automatización desarrollada pueda ser usada para detener e iniciar servicios de cualquier tipo adaptando los scripts en cada máquina y sin tocar el código de Ansible.

- Alistamiento y registro de servidores de la línea base del SDC de la misma manera que se realizó para el ambiente de pruebas. Esto se hizo a partir de un cruce del inventario de máquinas para el cliente SDC Bogotá para las cuales se identificaron problemas de conexión, información de registro desactualizada o problemas de configuración y de software que se resolvieron o se escalaron a otros equipos. Adicionalmente se hizo un monitoreo diario desde Kibana del estado de conexión de las más de 150 máquinas registradas para el mismo cliente y se hicieron las correcciones necesarias para mantener al mínimo el porcentaje de máquinas no alcanzadas por la herramienta.

- Ajustes al manual de alistamiento de servidores para registro en NEXT para agregar detalles que no fueron mencionados y fue necesario consultar durante el proceso y para corregir algunos pasos que podrían ser interpretados de forma incorrecta. Esta nueva versión del manual fue puesta a disposición del autor del documento original.

- Configuración de máquinas VIOS para autenticación por LDAP de manera que todos los usuarios registrados en el directorio activo de Microsoft puedan iniciar sesión en estos servidores tanto en el modo VIO como en el modo AIX. Gracias a esto también es posible hacer una conocer la autoría de cada uno de los cambios que se realicen sobre estos servidores. Esta configuración se realizó sobre un total de 74 máquinas y a partir de esto se completó un manual como guía para la configuración de otras máquinas en el futuro.

- Diligenciamiento de Health checks para servidores VIOS, AIX y Windows Server de manera manual para el aseguramiento de parámetros de seguridad en cada una de las máquinas de acuerdo con los estándares recomendados por sistema operativo.

11. CONCLUSIONES

- Se logró un avance significativo en la adopción y despliegue de automatizaciones de parchado a nivel de IBM Colombia gracias al trabajo conjunto que se realizó durante la práctica con los equipos de sistemas operativo y de I+D. Esto se evidenció en los ambientes de prueba puestos a disposición de todo el equipo y las mejoras aplicadas para ampliar la compatibilidad y las funcionalidades relacionadas con tareas pre y post parchado, entre otras.

- La automatización de tareas de administración de sistemas implica la aplicación de conocimientos de diferentes ramas de la Ingeniería de sistemas tales como las redes y telecomunicaciones, sistemas operativos y desarrollo de software para el análisis de requerimientos, diseño y creación de scripts u otras piezas de software que permitan reemplazar o apoyar este tipo de labores técnicas.

- Ansible se presenta como una opción de automatización que puede adaptarse a una infraestructura de cualquier escala y es muy flexible en términos de configuración y seguridad. Gracias a estas características pudo ser utilizado en diferentes ambientes tanto de desarrollo como productivos.

- La sintaxis utilizada por Ansible expresada en formato YAML, compatible con código en Python y con librerías de Python para la manipulación de datos hace que la curva de aprendizaje sea corta si ya se está familiarizado con la programación, lo cual se permitió entender y escribir nuevas tareas para los Playbooks existentes después de un corto tiempo de aprendizaje.

REFERENCIAS BIBLIOGRÁFICAS

[1] INTERNATIONAL BUSINESS MACHINES CORPORATION, «ANNUAL REPORT,» 26 Febrero 2019. [En línea]. Available: https://www.sec.gov/Archives/edgar/data/51143/000104746919000712/a2237254z10-k.htm. [Último acceso: 1 Marzo 2021].

[2] Portafolio, «IBM Colombia siete décadas de innovación A su llegada, la compañía ofreció en el país relojes, balanzas y máquinas de escribir y de tabulación.,» *Portafolio,* 2007.

[3] IBM, «Directory of worldwide contacts,» IBM, 2021. [En línea]. Available: https://www.ibm.com/planetwide/co/. [Último acceso: 15 Enero 2021].

[4] IBM, «Productos,» 2021. [En línea]. Available: https://www.ibm.com/co-es/products. [Último acceso: 5 Febrero 2021].

[5] IBM Research, «About,» 2019. [En línea]. Available: https://www.draco.res.ibm.com/about?lnk=nav. [Último acceso: 26 Febrero 2021].

[6] IBM, «Annual Report,» New York, 2019.

[7] IBM, «Chronological History of IBM,» [En línea]. Available: https://www.ibm.com/ibm/history/history/history_intro.html. [Último acceso: 13 Enero 2021].

[8] IBM GTS Colombia, «Inducción GTS Colombia,» 2021. [En línea]. Available: https://w3.ibm.com/w3publisher/inducci-n-gts-colombia. [Último acceso: 11 Enero 2021].

[9] Red Hat, «¿Qué es la automatización de la infraestructura?,» [En línea]. Available: https://www.redhat.com/es/topics/automation/what-is-infrastructure-automation. [Último acceso: Junio 2021].

[10] VMWare, «What is IT automation?,» [En línea]. Available: https://www.vmware.com/topics/glossary/content/it-automation. [Último acceso: Junio 2021].

[11] Red Hat, «What's IT automation?,» [En línea]. Available: https://www.redhat.com/en/topics/automation/whats-it-automation. [Último acceso: Junio 2021].

[12] Red Hat Ansible, «How Ansible Works,» [En línea]. Available: https://www.ansible.com/overview/how-ansible-works. [Último acceso: Julio 2021].

[13] Red Hat Ansible, «Introduction to modules,» [En línea]. Available: https://docs.ansible.com/ansible/latest/user_guide/modules_intro.html. [Último acceso: Abril 2021].

[14] Red Hat Ansible, «Intro to playbooks,» [En línea]. Available: https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html. [Último acceso: Marzo 2021].

[15] Rapid7, «Patch Management: Benefits and Best Practices,» [En línea]. Available: https://www.rapid7.com/fundamentals/patch-management/. [Último acceso: Junio 2021].

[16] Microsoft, «Windows Server Update Services (WSUS),» 22 Mayo 2017. [En línea]. Available: https://docs.microsoft.com/en-us/windows-server/administration/windows-server-update-services/get-started/windows-server-update-services-wsus. [Último acceso: Julio 2021].

[17] Red Hat, «Red Hat Satellite. Chapter 7. Patching Your Systems,» [En línea]. Available: https://access.redhat.com/documentation/en-us/red_hat_satellite/6.2/html/quick_start_guide/patching_your_systems. [Último acceso: Junio 2021].

[18] IBM, «Network Installation Management,» [En línea]. Available: https://www.ibm.com/docs/en/aix/7.1?topic=installing-network-installation-management. [Último acceso: Agosto 2021].

[19] Red Hat, «CHAPTER 3. INSTALLING SATELLITE SERVER,» [En línea]. Available: https://access.redhat.com/documentation/en-us/red_hat_satellite/6.2/html/installation_guide/installing_satellite_server. [Último acceso: 10 Septiembre 2021].

[20] F. C. P. Díaz, IBM: El camino hacía las soluciones cognitivas, Bogotá: Fundación Universitaria Jorge Tadeo Lozano, 2017.

[21] Red Hat, «How to build your inventory,» [En línea]. Available: https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html. [Último acceso: Mayo 2021].

[22] ManageEngine, «What is patch management?,» [En línea]. Available: https://www.manageengine.com/patch-management/what-is-patch-management.html. [Último acceso: Julio 2021].