AUTOMATIZACIÓN DE CIERRE DE PUERTOS Y ZONING

Automatización de peticiones en sistemas distribuidos de almacenamiento

Adrián Rodríguez Fajardo

Resumen– El siguiente informe presenta una investigación sobre la automatización del cierre de puertos y la baja de Zoning en una red SAN (Storage Area Network) con redundancia. En una red SAN, es fundamental garantizar la disponibilidad y la integridad de los datos almacenados, por lo que es importante contar con herramientas que permitan automatizar ciertas tareas de mantenimiento y gestión. En este sentido, el objetivo de este trabajo de fin de grado es analizar las posibilidades de automatización del cierre de puertos y la baja de zonificación en una red SAN con redundancia, evaluando las distintas opciones disponibles y proponiendo soluciones eficientes y seguras. Para ello, se han llevado a cabo pruebas en un entorno real y se han obtenido resultados que demuestran la viabilidad de la automatización en estos procesos.

Palabras clave– Almacenamiento, Automatización, Cabinas, CI/CD, CISCO, DevOps, Linux, SAN, Scripting, SQL, Switch, Puerto, Zoning.

Abstract– The following report presents research on the automation of port shutdown and zoning deactivation in a redundant Storage Area Network (SAN). In a SAN, it is crucial to ensure the availability and integrity of stored data, making it important to have tools that allow for the automation of certain maintenance and management tasks. The objective of this undergraduate thesis is to analyze the possibilities of automating port shutdown and zoning deactivation in a redundant SAN, evaluating different available options and proposing efficient and secure solutions. A set of tests have been conducted in a real environment and the obtained results demonstrate the feasibility of automation in these processes.

Keywords– Storage, Automation, Cabinets, CI/CD, Cisco, DevOps, Linux, SAN (Storage Area Network), Scripting, SQL, Switch, Port, Zoning.

1 INTRODUCCIÓN

NTT DATA Europe & Latam es una empresa japonesa especializada en la integración de sistemas de comunicaciones. Es una filial de Nippon Telegraph and Telephone (NTT), una de las empresas más importantes de Japón. Su actividad principal es la gestión de servicios tecnológicos y ocupa la quinta posición en el ranking mundial según Forbes Global 2000[9].

El trabajo de fin de grado se desarrollará en el departamento de Infraestructures Engineering de esta misma empresa. Este departamento se encarga de la planificación, construcción y mantenimiento de infraestructuras críticas para las empresas, tales como el almacenamiento y las copias de seguridad. Durante toda la etapa de desarrollo, contamos con la ayuda de dos miembros del equipo de DevOps de NTT Data, un supervisor senior y un desarrollador junior, y del asesoramiento de uno de los administradores que se encarga de realizar las peticiones que se plantean. El papel de estos tres compañeros consiste en guiarnos durante todo el proceso de automatización. Tendrán un papel consultivo y no directivo, pues seremos Francisco Javier y yo los que tomemos las decisiones.

El proyecto se realizará conjuntamente con otro estudiante de la mención, Francisco Javier Honrubia Ruiz. Debido a la envergadura del proyecto, sensibilidad de los datos y la densa cantidad de conocimientos que se necesitan para realizar los automatismos, se trata de un proyecto excesivamente exigente para un solo desarrollador.

• E-mail de contacto: adrianrodriguezfajardo@gmail.com
• Mención realizada: Ingeniería de Computadores
• Trabajo tutorizado por: Anna Sikora (Department of Computer Architecture and Operating Systems)
• Curso 2022/2023

El *Portal de Aprovisionamiento de Storage* o *PAS* es el nombre que recibe la plataforma web privada de la empresa, donde los administradores realizan tareas de monitorización y administración de las tecnologías de las que se dispone, como cabinas de IBM o switches de CISCO. Cuando se requiere ejecutar una acción sobre una de las tecnologías como, por ejemplo, abrir un puerto o dar de alta un disco duro, los administradores han de crear un cambio en la infraestructura a partir de las APIs de los dispositivos. Estos cambios se realizan a través del *PAS* y reciben el nombre de peticiones.

Estas peticiones tienen gran cantidad de metadatos asociados: administradores asignados, tecnología destino, fechas, logs, etc. Dependiendo de los pasos que tengan asociados, a grandes rasgos, las peticiones pueden clasificarse en: validadas, en proceso, en ejecución y finalizadas. Más adelante profundizaremos en la definición de estas fases. Lo importante es que se comprenda que es el administrador el que se encarga de avanzar de fase en fase las peticiones interactuando directamente con la página web *PAS* en lugar de con la infraestructura, de forma manual.

1.1. Estado del arte

En los últimos años, la automatización de tareas en redes SAN[1] ha ganado relevancia debido a la necesidad de garantizar la disponibilidad y la integridad de los datos almacenados, así como reducir el volumen de trabajo humano, lento e imperfecto. En este equipo de trabajo se han abordado previamente la automatización del cierre de puertos y la baja de Zoning en redes SAN con redundancia, con el objetivo de mejorar la eficiencia y la seguridad de estos procesos.

Se han propuesto distintas soluciones, como el uso de scripts y herramientas de scripting para interactuar con los dispositivos de almacenamiento y ejecutar comandos específicos. Estos enfoques han demostrado ser eficientes en entornos controlados, pero pueden requerir una configuración y mantenimiento manual.

Además, se habían desarrollado soluciones basadas en APIs proporcionadas por los fabricantes de los dispositivos de almacenamiento. Estas APIs permiten la comunicación y el control programático de los dispositivos, lo que facilita la automatización de tareas como el cierre de puertos y la baja de Zoning.

En este trabajo, se abordarán los desafíos y las posibilidades de automatización del cierre de puertos y la baja de Zoning en una red SAN con redundancia. Finalmente se implementará una solución basada en el scripting basada en la ejecución programada de comandos en los switches de CISCO.

En el método instaurado previo a nuestro proyecto, los administradores eran los encargados de reunir la información necesaria, realizar las comprobaciones previas, escribir a mano los comandos en las cabinas, validar los cambios y finalizar la petición.

Las principales desventajas del sistema manual son:

- Costo de tiempo elevado, de entre media hora para un técnico más experimentado a una hora para un aprendiz.
- Error humano: El operario asignado a la petición no solo ha de supervisar la petición, sino que también es el responsable de introducir y ejecutar los comandos en las cabinas.
- No se guarda ningún registro de los comandos ejecutados, outputs o mensajes de error resultantes de las ejecuciones (logs).

Con la solución que aporta nuestro proyecto, no solo perseguimos salvar estas desventajas sino también aportar mejoras significativas en materia de rendimiento, seguridad y registros.

2 OBJETIVOS

Los objetivos que se tratarán a lo largo de este proyecto se pueden clasificar en los siguientes dos ámbitos: el de desarrollo y el de administración, o Backend y Frontend. El grueso del proyecto se encuentra en la gestión y administración de las redes SAN con el propósito de automatizar la baja de puertos y de *Zoning* en el Backend. Estos cambios producirán importantes modificaciones en la web de los administradores de la empresa (el PAS), ya que en lugar de realizar ellos mismos las peticiones, solamente deberán ejecutar el automatismo y validar la respuesta al acabar éste.

2.1. Objetivos de Backend

Los objetivos de desarrollo o de Backend son los siguientes:

1. Automatizar las peticiones de cierre de puertos en una red SAN con redundancia.
2. Automatizar las peticiones de baja de Zoning en una red SAN con redundancia.
3. Implementar un sistema de logs que permita hallar el origen de un posible fallo o validar si el automatismo se ha completado sin errores.
4. El software creado deberá dividirse en distintos módulos, para que los pasos puedan ser reordenados en un futuro o insertar un paso intermedio sin que afecte a las otras funcionalidades del automatismo.
5. El automatismo será capaz de reunir la información necesaria para generar los comandos de las bases de datos o de las APIs que disponga el SAN.
6. El automatismo generará los comandos necesarios a partir de la información reunida.
7. El automatismo generará comandos de comprobación antes de ejecutar los comandos, para asegurar que se cumplan las condiciones necesarias para poder ejecutar en ese mismo momento.
8. Se enviarán los comandos a los switches de CISCO para ser ejecutados (tanto los del automatismo como los de comprobación) en el entorno de producción, recogiendo los logs y el output de éstos.

[1] Definidas en el apartado de 'Entorno'.

9. Después de la ejecución de los comandos, el automatismo enviará de nuevo los comandos de post-check o comprobación posterior, para comprobar que los cambios deseados en la infraestructura se han realizado correctamente.

10. Debido a la sensibilidad del sistema, todo el software creado será testeado en local, en el entorno de desarrollo, en Preproducción y, finalmente, en Producción.

11. Durante el desarrollo se utilizará un sistema de control de versiones de código para poder implementar los cambios de forma progresiva y mantener un historial de todos los cambios.

12. Durante todo el desarrollo se hará uso de VPN y máquinas virtuales por motivos de ciberseguridad ante ataques externos.

13. Los automatismos y el software se implementarán en Perl y Python, los lenguajes usados actualmente por la compañía en este tipo de desarrollos, para que los desarrolladores que ya estaban en el proyecto puedan interpretar o modificar su contenido en un futuro de forma más intuitiva para ellos.

14. Integrar los cambios de nuestro desarrollo al sistema actual de la empresa: bases de datos, programaciones y listados de peticiones.

15. Terminar los automatismos antes de mayo, para poder centrarnos en el testeo y la documentación.

16. Generar documentación de todo el software desarrollado.

2.2. Objetivos de Frontend

Los objetivos de administración o de Frontend son los siguientes:

17. El administrador responsable de la petición de baja de Zoning o cierre de puertos, una vez nuestro automatismo esté operativo, solo deberá de hacer click en el botón de “ejecutar” para que éste actúe, y revisar los posibles mensajes de error que pudieran salir.

18. El administrador podrá visualizar, antes, durante o después de la ejecución de los comandos, en qué estado se encuentra la petición y todas las fases del automatismo.

19. El administrador podrá consultar en el apartado de peticiones de la web, los logs y la información sobre las peticiones automatizadas.

20. Cuando una fase no pueda finalizar o finalice con errores, se mostrarán los mensajes de error.

2.3. Confidencialidad

La integración de estos scripts desarrollados con el *PAS* ha sido una de las tareas más largas e interesantes de este proyecto. Pero, debido a que la empresa ofrece *Outsourcing* de servicios a dos de los bancos más importantes de España, se firmó un contrato de confidencialidad que nos impide desvelar cierta información relativa a las APIs internas y su integración con las múltiples modificaciones realizadas a lo largo del proyecto.

Se recuerda entonces que, por motivos de confidencialidad y seguridad de la empresa en la que se desarrolla el proyecto, no se podrán mostrar capturas del código ni adjuntar el mismo en el proyecto final. Sin embargo, pueden consultarse los comandos de Cisco detallados en el apéndice de este artículo.

3 ENTORNO

3.1. Redes SAN

La definición de una red SAN puede variar dependiendo del enfoque de estudio. Según la documentación de IBM Corporation[1], una reconocida empresa tecnológica multinacional, *”Una red de área de almacenamiento (SAN) es una red dedicada adaptada a un entorno específico que combina servidores, sistemas de almacenamiento, conmutadores de red, software y servicios”*. Una definición más técnica y menos empresarial la encontramos de la mano de VMware Inc.[2], otra gran empresa tecnológica, que hace especial hincapié en las características tecnológicas: *”Una red de área de almacenamiento (SAN) es una red de alta velocidad independiente y dedicada que interconecta y suministra depósitos compartidos de dispositivos de almacenamiento a varios servidores.”*

En resumen, una red SAN es una solución de almacenamiento centralizada y de alto rendimiento que puede mejorar significativamente la eficiencia y disponibilidad de los datos en entornos empresariales.

3.2. Zoning

Una Storage Area Network o SAN se diferencia de las tradicionales redes LAN en que, haciendo uso de la alta velocidad de los canales de fibra óptica, los dispositivos de almacenamiento que la conforman no se pueden comunicar entre ellos. Los servidores de una red SAN solo pueden comunicarse y percibir la existencia de su propio dispositivo de almacenamiento. Esto recibe el nombre de Zonificación, aunque el término anglosajón *Zoning* está más extendido entre los profesionales del almacenamiento.

Según la documentación de IBM relativa a entornos con fibra óptica[3], el Zoning puede definirse como *”...una técnica de aislamiento de conexiones que se requiere al conectar hosts con el sistema de almacenamiento a través de Fibre Channel SAN”*. Dentro de cada switch se indica cuáles son las características de Zoning de los elementos que lo conforman, y los administradores definen los conjuntos de normas en los llamados *Zoneset*.

El Zoning persigue los siguientes objetivos:

- Crear una barrera entre diferentes entornos de red. Solo los miembros de la misma zona pueden comunicarse dentro de esa zona, y todas las comunicaciones externas están bloqueadas.

- Aislar cualquier adaptador de bus de host (HBA) individual por motivos de seguridad y fiabilidad.

- Permitir una segmentación más fina de la red de switches.

Es importante destacar que, para mantener la redundancia y accesibilidad de los datos, existen dos SAN (A y B) que trabajan en paralelo y que son capaces de soportar el tráfico de datos en caso de fallo de alguna de ellas. De esta forma, se garantiza la continuidad del servicio y la disponibilidad de los datos en todo momento.

3.3. Switch

Un *switch* es un dispositivo de red que permite la interconexión de múltiples dispositivos, como servidores y sistemas de almacenamiento, dentro de una red local. Estos dispositivos son de gran relevancia para el desarrollo de nuestros automatismos, ya que los comandos necesarios para realizar las modificaciones en la red SAN son ejecutados en estos elementos, que actúan como un intermediario con la infraestructura real.

En el contexto de una red SAN, los switches desempeñan varias funciones esenciales. En primer lugar, facilitan la comunicación entre los servidores y los sistemas de almacenamiento, permitiendo el acceso a los datos almacenados. Además, los switches proporcionan una mayor velocidad y ancho de banda, lo que garantiza un rendimiento óptimo en la transferencia de datos.

En el proyecto, la automatización del cierre de puertos y la baja de Zoning implica interactuar con los switches de la red SAN. Los switches desempeñan un papel crucial en el proyecto al permitir la comunicación entre los servidores y los sistemas de almacenamiento, así como al habilitar la configuración y control de los puertos necesarios. Su correcta gestión y configuración son esenciales para dar solución a este proyecto y para garantizar un funcionamiento óptimo y seguro de la red SAN.

3.4. Entornos de Pre y Post Producción

Es importante señalar que, para implementar la solución de la automatización, existen dos entornos de desarrollo independientes: el entorno de Preproducción y el entorno de Producción.

El entorno de preproducción (PRE) es un entorno de desarrollo completamente independiente del de producción (PRO) en el cual se facilitan herramientas y datos lo más similares al primero posible, para así asegurar que las pruebas sean precisas y realistas. El entorno de PRE se utiliza para desarrollar, probar y validar los cambios en la infraestructura de una forma más segura y controlada.

En el entorno de PRO se ejecuta la solución desarrollada en los servicios reales de trabajo, es decir, es la solución sobre la que operan todos los administradores y servicios de la empresa. Por lo tanto, no debe de ser modificada directamente, pues un error podría suponer pérdidas de confianza y económicas incalculables para la empresa. Es importante que el proceso de implementación se realice con sumo cuidado, y que se habiliten planes de respaldo como copias de seguridad y rollback de acciones importantes.

Existen múltiples herramientas CI/CD, como Jenkins, Git o Ansible, que se encargan de desplegar y validar, en el momento que se desee o de forma automática, todos los cambios desarrollados con origen PRE y destino PRO. Estas herramientas ya están correctamente configuradas en el entorno de trabajo, por lo que en este proyecto de TFG no se han tenido que modificar o utilizar, más allá que para desplegar los cambios en los entornos de PRE y PRO una vez testeados y validados.

4 METODOLOGÍA

4.1. Desarrollo ágil de software

La metodología elegida para el proyecto será el Desarrollo ágil de software, un enfoque de toma de decisiones en proyectos de desarrollo de ingeniería del software, basado en un proceso iterativo e incremental [4].

En esta metodología, los requisitos y soluciones que definen el problema a tratar pueden evolucionar a lo largo del proyecto. Debido a la complejidad y la poca tolerancia a fallos del sistema sobre el cual se opera, pueden surgir contratiempos a lo largo del desarrollo que tengan repercusiones en la planificación. Algunos ejemplos de estos contratiempos pueden ser: semanas críticas en las cuales no sea posible hacer cambios en el código fuente, nuevas tecnologías que agranden el problema o problemas administrativos y de permisos. Además, si el tiempo lo permitiera, podrían considerarse nuevas peticiones para ser automatizadas después de las dos ya mencionadas.

Por lo tanto, una metodología ágil de software, iterativa e incremental, es perfecta en el escenario planteado, pues los requisitos irán evolucionando según las necesidades del proyecto.

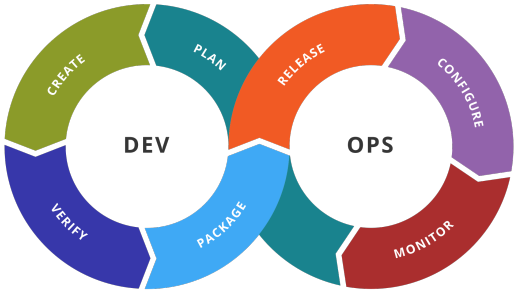
4.2. Cultura DevOps

DevOps[8] es un acrónimo en lengua inglesa que deriva de los términos Development (desarrollo) y Operations (operaciones), y se refiere a una corriente cultural o movimiento que se enfoca en la comunicación, la colaboración y la integración de los desarrolladores de software y los administradores de sistemas.

El enfoque de DevOps consiste en automatizar el proceso de entrega de software y los cambios en la infraestructura, con el propósito de establecer un entorno que permita desarrollar, probar y lanzar el software con mayor velocidad y fiabilidad.

Cabe destacar que las organizaciones como NTT Data que requieren realizar entregas de código de forma frecuente, necesitan de la implementación de las prácticas y herramientas asociadas a DevOps. Estas herramientas se distribuyen a lo largo de las distintas fases que componen el ciclo del DevOps, que pueden observarse en la Figura 1. El ciclo DevOps agrupa las siguientes fases: Desarrollo de

Fig. 1: Ciclo de vida DevOps [8]



código, Integración contínua (CI), Testing, Package, Gestión y automatización de versiones (release), Configuración de la infraestructura mediante IaaC (Infraestructure as a Code) y Monitorización de rendimiento.

Aunque este proyecto se centra en las fases del ámbito del desarrollo (código, CI, Testing y Package), se utilizan las herramientas de las fases administrativas (Release, IaaC y Monitoring) para analizar los resultados y rendimientos conseguidos, cerrando así el ciclo de vida del DevOps.

4.3. Desempeño de la metodología elegida

Se ha establecido un proceso de seguimiento que se realiza de manera continua, con el objetivo de detectar posibles problemas a tiempo. Para ello, se realizan reuniones diarias de seguimiento, dónde se revisa el progreso del proyecto, se analizan las tareas realizadas, el tiempo invertido, las dificultades encontradas y los próximos pasos a seguir.

Además, se han implementado diversas herramientas de monitorización y control de calidad, que permiten evaluar el desempeño de la metodología y detectar posibles desviaciones en el proceso de desarrollo. Estas herramientas incluyen pruebas automatizadas, integración continua, análisis estático de código, entre otras. Una de estas pruebas automatizadas consiste en comprobar si la zona o el puerto que se desea dar de baja tiene discos asignados todavía. En caso afirmativo se procede a paralizar el automatismo y mostrar el error correspondiente al administrador de sistemas.

En general, se ha observado que la metodología de Desarrollo Ágil de Software ha sido altamente efectiva en el proyecto, permitiendo una mayor colaboración con los equipos de desarrollo y operaciones, una mayor rapidez en la entrega de versión de software (entrega continua/despliegue continuo) y una mejora en la calidad del producto final. Los procesos de seguimiento y evaluación implementados han permitido detectar a tiempo posibles problemas y corregirlos, asegurando así el éxito del proyecto.

5 DESARROLLO

5.1. Automatización de baja de puertos

En una red de almacenamiento SAN, la gestión de los recursos físicos es una tarea clave para evitar cuellos de botella y, así, conseguir el rendimiento óptimo del sistema. Una de estas tareas es la asignación y liberación de los puertos que participan en las recurrentes conexiones entre dispositivos que conforman la red de almacenamiento.

En este punto se presentará el procedimiento desarrollado para dar de baja estos puertos cuando ya no se requiera de su operatividad. Aunque se describirán con más detalle en los próximos apartados, las distintas fases del automatismo se pueden observar en la figura 2, en la sección B del Apéndice.

5.1.1. Generación de asignaciones para cierre de puertos

En esta primera fase se reúne la información necesaria para, en las próximas fases, construir los comandos necesarios para cerrar los puertos desde las cabinas.

Con dicho propósito se procede a desarrollar un script en Perl con el ID de la petición como input, *CerrarPuertos_Asignacion.pl*, que persigue los siguientes objetivos:

1. El script accede a la base de datos y comprueba que la petición existe y los metadatos son correctos.

2. De entre otros datos, se recupera el nombre del host que contiene los puertos a cerrar, el estado de los puertos y la dirección del puerto.

3. Se guarda la información en una tabla en la base de datos. Esta es accesible desde el PAS y es permanente incluso después de finalizar la petición de cierre de puertos.

4. Se avanza la petición a la siguiente fase.

5.1.2. Generación de comandos para cierre de puertos

En esta segunda fase se utiliza la información almacenada en tabla de la fase anterior para construir los comandos a ejecutar en la cabina. Estos comandos[2] son idénticos a los utilizados por los administradores actualmente, con la diferencia de que son escritos y lanzados de forma automática.

Es importante recordar que todos los comandos se han de ejecutar dos veces debido a que toda la infraestructura está replicada, en SAN A y SAN B.

Podemos distinguir dos clases de comandos:

- Comandos de cierre: Comandos para cerrar los puertos en la cabina.

- Comandos de comprobaciones previas: Comandos para verificar que es viable y seguro proceder con el cierre. Los comandos servirán para verificar que el host asociado al puerto está listo y se puede proceder con el cierre.

- Comandos de comprobaciones posteriores: Comandos que comprueban que el cierre ha sido exitoso y se puede proceder a finalizar la petición.

Se desarrolla un script en Perl con el ID de la petición como input, *CerrarPuertos_Comandos.pl*, que realiza las siguientes tareas:

1. El script accede a la base de datos recupera los datos de la tabla donde se insertaron en la fase anterior.

2. Construye, a partir del nombre del servidor y la dirección interna del puerto, los comandos. No sólo los de cierre sino también los de comprobación.

3. El output de la ejecución de los comandos se guardan tanto en logs del sistema como en una tabla dedicada a este propósito.

4. Se avanza la petición a la siguiente fase.

[2]Los comandos se muestran y explican detalladamente más adelante en 'A ANEXO 1: Comandos de CISCO utilizados'.

5.1.3. Modificaciones en el Frontend para cierre de puertos

Es necesario hacer transparente a los administradores en qué fase del automatismo se encuentra la petición, para que puedan realizar un seguimiento de ésta, o pueda ser parada o modificada manualmente si fuera necesario.

Han sido modificados los archivos de PHP que facilitan al operario la información de seguimiento de la petición, pues no solo se ha de mostrar la información reunida y agrupada en tablas, sino que también se han de cambiar los pasos anteriores por los nuevos, para que sean coherentes con las fases de la automatización.

Después de esta fase de desarrollo, la web administrativa PAS muestra a tiempo real en qué estado se encuentra la petición y los resultados obtenidos de cada fase.

5.1.4. Scripts de ejecución de comandos para cierre de puertos

En esta fase se ha programado el script *CerrarPuertos_Ejecutar.py*, una rutina en Python que se encarga de realizar una conexión segura con la cabina habilitada, dónde se ejecutan las instrucciones de los comandos generados en fases previas. El output de las instrucciones es interpretado por una función independiente, que avanza o detiene la petición según el tipo de comando:

- Si la instrucción es un comando de baja, ejecuta el comando. Si se ejecuta el comando correctamente, se devuelve el valor de retorno exitoso para avanzar la petición.

- Si la instrucción es de comprobación, tanto anterior como posterior al automatismo, se analiza el output. Si después de terminar el test no se ha encontrado ningún error conocido, se devuelve el valor de retorno exitoso y se avanza la petición. En caso contrario, se devuelve el mensaje de error y se paraliza la petición.

En el caso de que el script no finalice con éxito, el equipo de administradores tendrá que revisar el historial de logs generado para averiguar qué fase ha fallado y por qué. Después, tomará las medidas que contemple para solucionar o proponer la ejecución de la petición.

Una finalización errónea del automatismo no es un fracaso. Al contrario. Detectar un error puede ser una tarea muy laboriosa para un administrador, mientras que el automatismo puede reconocerlo al instante e indicar las acciones a realizar.

5.1.5. Testeo de cierre de puertos en Preproducción

El testeo de los cambios realizados en el entorno de Preproducción es crucial para garantizar que los mismos sean implementados de manera efectiva y que no se desplieguen errores en el entorno de Producción. En el contexto de este proyecto, en el que se busca automatizar el cierre de puertos y la baja de Zoning en una red SAN, testear es de suma importancia debido a las graves repercusiones que podría tener un error en la infraestructura de un sistema bancario.

El proceso de testeo llevado a cabo en el entorno de Preproducción para verificar el correcto cierre de puertos y baja de Zoning incluye:

- El diseño de los casos de prueba.
- La ejecución de todas las fases del automatismo de forma individual a excepción de la fase de ejecución.
- El testeo de las fases de forma integrada.
- La corrección de los errores encontrados durante la fase de testeo y planteamiento de las soluciones que garanticen la calidad final.
- Finalmente, planear el salto a Producción y sus implicaciones.

5.1.6. Testeo de cierre de puertos en Producción

El testeo en Producción es una etapa crítica en el proceso de desarrollo. El testeo en Producción adquiere una importancia crucial debido a que el sistema debe operar de manera estable y confiable ya que, aunque se disponen de copias de seguridad de respaldo, ante una caída temporal de los servicios la empresa perdería credibilidad ante el cliente y podría causar algún error irreparable.

Al igual que en el testeo sobre el entorno de Preproducción se diseñarán los casos de prueba, se ejecutarán las mismas fases individual y conjuntamente, y se corregirán los errores. En esta fase, además, se propone la ejecución definitiva de los comandos en las cabinas de CISCO.

En esta fase tan crítica del desarrollo se ha de operar con máxima cautela. En primer lugar, se creará de forma manual una petición simulada de cierre de puertos, para poner a prueba el automatismo, ejecutando los comandos de cierre y las comprobaciones del sistema. Si la petición se completa exitosamente, se seleccionarán unas 10 peticiones extra reales para ser resueltas por el automatismo.

Si estas peticiones finalizan satisfactoriamente, la fase de testeo se dará por terminada y el proyecto avanzará a la siguiente petición. Sin embargo, el automatismo de cierre de puertos seguirá siendo validado por compañeros administradores que, una vez acabada la fase de testeo, usarán el automatismo y nos darán feedback.

5.1.7. Feedback de Producción

Finalizada la fase de testeo de cierre de puertos, el desarrollo se da por concluido salvo posibles modificaciones que pudieran ser necesarias, ya sea por características que no se tuvieron en cuenta en la reunión de requisitos o porque los administradores detectan que se podría mejorar algún elemento del automatismo.

Además, debido al cuidadoso sistema de logs, errores y alertas del automatismo, se puede consultar periódicamente la ejecución de las peticiones de cierre de puertos para asegurar el correcto funcionamiento.

Así pues, el automatismo de cierre de puertos en redes SAN incluye dos fuentes principales de feedback:

- Feedback directo del automatismo: Consiste en alertas y correos que se envían automáticamente cuando falla una de las fases del programa.
- Feedback indirecto del automatismo: Se puede consultar en cualquier momento el funcionamiento del automatismo, en qué fase se encuentra cada petición y los

datos intermedios generados, así como las salidas de la ejecución de comandos en las cabinas.

- Feedback de los administradores: A lo largo del ciclo del proyecto, se llevan a cabo varias reuniones semanales con el equipo de administradores de la empresa. Estas reuniones son claves para que nos transmitan funcionamientos inesperados o inquietudes respecto a alguna funcionalidad concreta del programa.

Cuando se recibe un feedback negativo de alguna característica del software, se establece una fecha límite de resolución, un nivel de urgencia y se resuelve con la mayor brevedad posible.

El feedback, tanto del automatismo como de los administradores, es de gran importancia en cuanto a la calidad final del producto que se brinda a la empresa, así como para asegurar que se cumplen los requisitos y objetivos definidos en las fases primerizas de este proyecto final de grado.

5.2. Automatización de baja de Zoning

El Zoning es una técnica utilizada en redes de almacenamiento de consideradas dimensiones para agrupar los dispositivos (discos físicos, discos virtuales y cintas) que las forman en unidades lógicas e independientes. Esta práctica busca incrementar la seguridad y el rendimiento, así como facilitar el manejo de todos estos dispositivos.

El objetivo principal del Zoning es limitar la comunicación entre los dispositivos de almacenamiento y los servidores o switches que intentar acceder a ellos. De esta forma se evita el envío innecesario o malintencionado de información .

En este punto se presentará el automatismo desarrollado en este proyecto para desmantelar estos grupos virtuales o zonas cuando ya no sean necesarias. El principal motivo para dar de baja una zona, es la eliminación de un disco, pero pueden haber otros.

El lector apreciará que los pasos son los mismos que para el automatismo de puertos. Esto se debe a que se ha de respetar la forma en la que funcionan las peticiones de aprovisionamiento en la web de la empresa (PAS). Las distintas fases del automatismo se pueden observar en la figura 3, en la sección B del Apéndice.

5.2.1. Generación de asignaciones para baja de Zoning

En esta primera fase se reúne la información necesaria para, en las próximas fases, construir los comandos necesarios para dar de baja las zonas indicadas.

Con dicho propósito se procede a desarrollar un script en Perl con el ID de la petición como input, *BajaZoning_Asignacion.pl*, que persigue los siguientes objetivos:

1. El script accede a la base de datos y comprueba que la petición existe y los metadatos son correctos.
2. De entre otros datos, se recupera el alias del dispositivo de la zona que se quiere dar de baja, el nombre del switch, el estado de los puertos del dispositivo y su dirección, y el tipo de Zoning (Smart o Regular).
3. Se guarda la información en una tabla en la base de datos. Esta es accesible desde el PAS y es permanente incluso después de finalizar la petición.
4. Se avanza la petición a la siguiente fase.

5.2.2. Generación de comandos para baja de Zoning

En esta segunda fase se utiliza la información almacenada en tabla de la fase anterior para construir los comandos a ejecutar en la cabina. Estos comandos[3] son idénticos a los utilizados por los administradores actualmente, con la diferencia de que son escritos y lanzados de forma automática.

Es importante recordar que todos los comandos se han de ejecutar dos veces debido a que toda la infraestructura está replicada, en SAN A y SAN B, al igual que en el cierre de puertos.

Podemos distinguir dos clases de comandos:

- Comandos de cierre: Comandos eliminar las zonas deseadas y aplicar los cambios.
- Comandos de comprobaciones previas: Comandos para verificar que es viable y seguro proceder con el cierre, que el Zoning del dispositivo es candidato a ser dado de baja y que antes de eliminar una zona no queda ningún dispositivo asociado a ella.
- Comandos de comprobaciones posteriores: Comandos que comprueban que la zona ha sido dada de baja exitosamente.

Se desarrolla un script en Perl con el ID de la petición como input, *BajaZonning_Comandos.pl*, que realiza las siguientes tareas:

1. El script accede a la base de datos, recupera los datos de la tabla donde se insertaron en la fase anterior.
2. Identifica el tipo de Zonning, Smart o Regular.
3. Según el tipo, construye todos los comandos a partir del nombre del switch y el alias y el puerto del dispositivo.
4. Se avanza la petición a la siguiente fase.

5.2.3. Modificaciones en el Frontend para baja de Zoning

Al igual que para el automatismo de cierre de puertos, se han modificado los archivos de PHP que facilitan al operario la información de seguimiento de la petición:

- Se muestra la información reunida y agrupada en tablas.
- Se modifican los pasos manuales anterior por los nuevos.
- Toda la información visualizable en el PAS es actual y se actualiza en tiempo real.

[3]Los comandos se muestran y explican detalladamente en el último apartado de 'A ANEXO 1: Comandos de CISCO'.

5.2.4. Scripts de ejecución de comandos para baja de Zoning

En esta fase se ha programado el script *BajaZoning_Ejecutar.py*, una rutina en Python que se encarga de realizar una conexión segura con la cabina habilitada, dónde se ejecutan las instrucciones de los comandos generados en fases previas. El output de las instrucciones es interpretado por una función independiente, que avanza o detiene la petición de la misma forma que en el apartado *"6.1.4 Scripts de ejecución de comandos para cierre de puertos"*, según si es un comando de cierre o de comprobación. Si la ejecución es exitosa y la validación no falla, la petición avanza al estado de finalizada.

En el caso de que el script no finalice con éxito, el equipo de administradores tendrá que revisar el historial de logs generado para averiguar qué fase ha fallado y por qué. Después, tomará las medidas que contemple para solucionar o proponer la ejecución de la petición.

5.2.5. Testeo de baja de Zoning en Preproducción

El proceso de testeo llevado a cabo en el entorno de Preproducción para verificar la eficacia del automatismo de baja de Zoning ha sido el mismo que el detallado para el de cierre de puertos. Se han seguido los mismos pasos y usado los mismos programas y técnicas.

5.2.6. Testeo de baja de Zoning en Producción

El proceso de testeo llevado a cabo en el entorno de Producción para verificar la eficacia del automatismo de baja de Zoning ha sido el mismo que el detallado para el de cierre de puertos. Se han seguido los mismos pasos y usado los mismos programas y técnicas.

5.2.7. Feedback de Producción

Finalizada la fase de testeo de cierre de zonas, el desarrollo se da por concluido salvo posibles modificaciones que pudieran ser necesarias, ya sea por características que no se tuvieron en cuenta en la reunión de requisitos o porque los administradores detectan que se podría mejorar algún elemento del automatismo.

Las fuentes de feedback del automatismo, detalladas en la sección equivalente de cierre de puertos, ha sido de gran importancia para esta petición en particular, tal y como se explicará en el apartado de resultados de este informe.

6 PRESENTACIÓN Y DISCUSIÓN DE RESULTADOS

El presente apartado se enfoca en la presentación e interpretación de los principales hallazgos. Además, se discutirá si los resultados esperados de este trabajo eran o no alcanzables desde un principio con los recursos disponibles y qué conocimientos adquiridos en los estudios de ingeniería informática han sido claves para resolver este proyecto.

Desde antes de empezar el trabajo de fin de grado, ya se había pactado con la empresa y la universidad los objetivos. Y es que la hora de manipular la infraestructura de almacenamiento de un banco se ha de operar con suma cautela a cada paso, planificando los tests necesarios y sin desplegar en el entorno de Producción hasta que cada linea de código haya sido analizada. Sin embargo, aunque el reto y la responsabilidad iba más allá de un trabajo universitario habitual, creo que la experiencia acumulada en el grado ha permitido que los resultados esperados fueran factibles con una preparación previa adecuada.

Los resultados esperados del proyecto iban más allá de la finalización del código y su correcto funcionamiento. Las soluciones de software debían de estar testeadas, bien documentadas (tanto para la empresa como para la universidad), satisfacer unos criterios de rendimiento y seguridad. Además hubo que adaptarlas a la estructura de la empresa y al lenguaje de programación habitual de los compañeros.

Los principales resultados de la solución desarrollada son los siguientes:

- Tras recibir el feedback de los administradores que han hecho uso de los automatismos del proyecto, podemos confirmar que éstos funcionan correctamente, cumpliendo con todos los objetivos detallados al inicio del proyecto.
- El tiempo empleado por la corporación para resolver una de estas peticiones ha bajado drásticamente de los 30 minutos (aproximadamente y dependiendo del nivel de conocimiento del administrador) a apenas unos segundos.
- El automatismo no solo es instantáneo sino que, además, elimina el error humano.
- El elaborado sistema de logs ha permitido en varias ocasiones detectar errores no solo nuestros, sino también otros errores en los recursos que utilizábamos.
- El software resultante ha sido testeado mediante pruebas unitarias y de integración, para asegurar la calidad del producto entregado.
- La solución ha sido creada y probada primero en un entorno local y aislado, después en el entorno real de Preproducción y, finalmente, probado en el entorno de Producción.
- Las fechas de las fases indicadas en la planificación se han cumplido y al finalizar la estancia de prácticas en la empresa los dos automatismos se encuentran operativos.
- El trabajo ha sido realizado en su completitud de forma autónoma, adoptando nuestro superior un puesto de supervisor. Esto ha sido posible gracias a los conocimientos adquiridos en las estancias de prácticas curriculares previas, tanto por mi parte como por parte de F. Javier Honrubia, el otro estudiante que ha participado en este desarrollo.
- El proyecto se ha resuelto utilizando los recursos y programas pactados en el inicio de este. No ha sido necesario adquirir material adicional.
- Se ha aplicado la metodología ágil y la filosofía DevOps, y utilizado las herramientas propias de cada fase del desarrollo.

En resumen, el proyecto se ha efectuado con los recursos y tiempo planificados siendo fiel a la metodología de desarrollo ágil de software y a la filosofía DevOps. Se ha realizado testing y utilizado el feedback después para asegurar la calidad del producto entregado. Además de entregar un producto de calidad siguiendo la planificación a la perfección, se ha diseñado una herramienta muy útil para la empresa, que sentará un precedente al ahorrar tiempo y problemas derivados de la resolución de estas peticiones.

Para una hipotética retirada de una cabina, se pueden crear unas 30 peticiones aproximadamente, entre cierre de los puertos de la cabina y la baja de las zonas asignadas a los dispositivos y puertos. Si utilizamos el tiempo promedio de 30 minutos por petición [4] el tiempo total asciende a las 15 horas, o dos días completos de un administrador trabajando de forma continua y sin descansos. Tal y como se muestra en la Tabla 1, la mejora ha sido notable en términos de tiempo y uso de recursos. Exactamente ha supuesto un *SpeedUp* de x180, además de las mejoras en calidad de seguridad y logs ya mencionadas.

TABLA 1: TIEMPO NECESARIO PARA DAR DE BAJA EL ZONING Y LOS PUERTOS AL DESACTIVAR UNA CABINA

Sistema	Tiempo aprox. de resolución
Antiguo	15 horas
Nuevos automatismos	5 minutos

7 CONCLUSIONES Y TRABAJO FUTURO

En esta sección se expondrá de forma sintética la oferta de valor de este proyecto: qué ha aportado a la empresa y a mí, personalmente. Seguidamente se revisarán los objetivos y su resolución. Finalmente se explorarán los posibles puntos de mejora y extensiones al trabajo.

7.1. Revisión de objetivos

En este apartado se detallará como se han cumplido los objetivos a grandes rasgos y sin profundizar en la solución, ya que la resolución del objetivo es, en la mayoría de los puntos, una solución software que no puede ser mostrado por temas de confidencialidad. Los objetivos que cumplen estas características son, nombrados por su número en el apartado 4 de este artículo: 1, 2, 4, 5, 6, 7, 8, 9, 13 y 14 de los objetivos de Backend y 17, 18, 19 y 20 de Frontend. Todos estos objetivos han sido diseñados, codificados, validados y testeados con éxito.

En cuanto al sistema de logs y gestión de errores presentado en el apartado de resultados, se ha usado una clase desarrollada en Python que crea y modifica ficheros ".log" en un directorio del repositorio de la empresa para después mostrarlos al administrador si este los requiere. La información mostrada en los logs son metadatos de la petición, mensajes de error, salidas de los comandos y línea exacta de fallo en caso de error entre otros.

[4] Este tiempo fue consensuado en múltiples ocasiones por los administradores en las numerosas reuniones llevadas a cabo, aunque el tiempo no ha sido medido en el proyecto.

Para gestionar las nuevas funcionalidades hemos tenido que adaptar y crear archivos .php en un ambiente MVC (Modelo-Vista-Controlador) gestionado por un framework llamado Yii2, que agiliza el proceso de renderizar y pasar datos a las vistas desde el modelo y pasando por el controlador. Así hemos conseguido hacer visible el avance de nuestros automatismos para los administradores de la empresa.

La documentación realizada para la empresa, además del apartado de desarrollo y apéndice de este informe, cuenta con múltiples capturas y anotaciones del código, y servirá no solo para revisión y comprensión del automatismo, sino también para replicarlo en futuros automatismos, de forma más ágil.

7.2. Extensiones

En primer lugar, un punto de mejora para el proyecto hubiera sido realizar una análisis de rendimiento exhaustivo y metódico del automatismo. Si el tiempo lo hubiera permitido y el equipo nos hubiera asignado un administrador unas cuántas horas para la tarea, sería interesante haber cronometrado cuánto tiempo tarda el operario en realizar cada una de las peticiones y como incrementa o decrementa el tiempo según si la petición es de cierre de puertos o de Zoning o si la baja de Zoning es de tipo Smart o Regular. Después hacer un recuento de todas las peticiones que han entrado y así calcular cuantas horas y dinero aproximado se está ahorrando mensual y anualmente para cuantificar el valor en la empresa de nuestro proyecto.

En cuanto a futuras vías de desarrollo, un añadido al automatismo implementado, aunque algo complejo de realizar dados los recursos disponibles y los conocimientos actuales, consiste en ejecutar las peticiones de forma automática, ya que actualmente la tarea de empezar el automatismo (pulsando un botón) es todavía responsabilidad del administrador. La idea sería crear una ventana o programación de peticiones, que las ejecutara todas de golpe en una fecha y hora determinadas. Con esta modificación estaríamos hablando ya de una tarea totalmente automatizada, donde salvo excepción no contemplada que terminara en error, el operario no debería de intervenir.

7.3. Aportaciones del TFG

Al realizar el proyecto dentro del marco laboral de una empresa multinacional, la elaboración de este TFG no ha aportado valor únicamente al estudiante, sino que la empresa ha salido beneficiada. A continuación se detallarán las aportaciones de este proyecto para ambas partes.

7.3.1. Aportaciones para el estudiante

En cuanto a la experiencia personal y crecimiento como estudiante en prácticas, puedo afirmar que ha sido muy notable. En primer lugar he trabajando siguiendo la filosofía DevOps, haciendo uso de sus herramientas más extendidas como *GitHub*, *Jenkins* y *Nagios*, aplicando desde técnicas como el *scripting* y la automatización, hasta conceptos como el desarrollo incremental, despliegue continuo o integración continua. Esta experiencia me ha convertido en un

trabajador DevOps muy capaz y he podido encontrar mi pasión y un espacio en su plantilla como Junior al terminar la carrera.

Además, haber realizado el trabajo en tándem con el compañero F. Javier Honrubia, que se había especializado en la rama de administración, mientras que yo me especialicé en la de desarrollo, nos ha permitido trabajar con un punto de vista global tanto del equipo como del panorama DevOps.

7.3.2. Aportaciones para la empresa

La empresa se ha visto beneficiada de varias maneras. En primer lugar, nuestro punto de vista ha servido para plantear como se estaban realizando algunas tareas de programación y diseño, así como la necesidad de actualizar el código *legacy* de la empresa.

Sin embargo, el principal valor de los dos automatismos creados para la empresa es ahorro de tiempo y, por lo tanto, de dinero invertido en horas de trabajo de un administrador especializado en resolver este tipo de peticiones. Desde que se puso en funcionamiento la versión final los automatismos, se han ahorrado entre treinta minutos y una hora por cada petición resuelta automáticamente por el software, pues la tarea del administrador es pulsar el botón del portal para avanzar la petición y revisar que no hayan errores. Estas horas son recuperadas por el equipo y pueden ser utilizadas en otras tareas de monitorización.

Por otro lado, la solución de Software no solo se ejecuta de forma instantánea, sino que además aporta otros beneficios para la empresa eliminando el fallo humano y dejando un historial detallado de logs. Gracias al histórico de logs que genera el programa podemos saber no solo si la fase fue exitosa o falló, sino el porque falló y en qué linea exacta del código.

En resumen, la solución entregada a la empresa automatiza la resolución de las dos peticiones, ahorrando tiempo y recursos. Elimina el fallo humano y cuenta con un sistema de errores y logs que aseguran la recuperabilidad en caso de fallada, ya sea una fallada del automatismo, de la conexión con una base de datos o por la dependencia con cualquier otro módulo.

Referencias

[1] International Business Machines Corporation. *¿Qué es una red de área de almacenamiento?*. Recuperado de URL de la página web el 06/03/2023. https://www.ibm.com/es-es/topics/storage-area-network

[2] VMware, Inc. *Red de área de almacenamiento (SAN)*. Recuperado de URL de la página web el 06/03/2023. https://www.vmware.com/es/topics/glossary/content/storage-area-network-san.html

[3] International Business Machines Corporation. *Fibre Channel zoning*. Recuperado de URL de la página web el 06/03/2023. https://www.ibm.com/docs/en/flashsystem-a9000/12.3.2?topic=connectivity-fibre-channel-zoning

[4] Fundación Wikimedia, Inc. *Desarrollo ágil de software*. Recuperado de URL de la página web el 12/03/2023. https://es.wikipedia.org/wiki/Desarrollo_ágil_de_software

[5] CISCO. *CISCO VSAN*. Recuperado de URL de la página web el 16/03/2023. https://www.cisco.com/c/en/us/td/docs/switches/datacenter/mds9000/sw/6_2/configuration/guides/fabric/nx-os/nx_os_fabric/vsan.html

[6] CISCO. *CISCO Zoning Comandos*. Recuperado de URL de la página web el 16/03/2023. https://www.cisco.com/c/en/us/td/docs/switches/datacenter/mds9000/sw/6_2/configuration/guides/fabric/nx-os/nx_os_fabric/zone.html

[7] NAVARRO CADAVID, ANDRÉS; FERNÁNDEZ MARTÍNEZ, JUAN DANIEL; MORALES VÉLEZ, JONATHAN *Revisión de metodologías ágiles para el desarrollo de software PROSPECTIVA, vol. 11, núm. 2, julio-diciembre, 2013, pp. 30-39. Universidad Autónoma del Caribe*

[8] Fundación Wikimedia, Inc. *DevOps toolchain*. Imagen recuperada de URL de la página web el 15/04/2023. https://en.wikipedia.org/wiki/DevOps_toolchain

[9] Fundación Wikimedia, Inc. *NTT Data*. Recuperado de URL de la página web el 12/03/2023. https://es.wikipedia.org/wiki/NTT_Data

Agradecimientos

Me gustaría expresar mi sincero agradecimiento a la tutora del TFG por su invaluable ayuda y dedicación a lo largo de este proceso. Su compromiso con mi proyecto y su disposición para tutorizar no solo mi trabajo, sino también el de otro estudiante, es verdaderamente admirable. Su orientación y apoyo continuo han sido fundamentales para el éxito de mi investigación. Estoy enormemente agradecido por su generosidad y compromiso con nuestra formación académica.

Para terminar esta sección, me gustaría hacer énfasis en el gran trabajo de mi compañero F. Javier Honrubia. Sus conocimientos y experiencia en el equipo de administradores han sido claves para agilizar la comunicación con los operarios y ha cumplido su parte del trabajo de forma profesional y disciplinada. Además, la combinación de aptitudes y talentos ha sido una tarea dinámica y constructiva.

ANEXO

A ANEXO 1: COMANDOS DE CISCO

En esta sección, se muestran los comandos que el automatismo presentado es capaz de lanzar en las cabinas para conseguir los objetivos y garantizar el correcto funcionamiento del programa. Como fuente de información se ha utilizado principalmente la documentación interna de la empresa y la documentación en línea de CISCO[6].

A.1. Comandos de posicionamiento

Los comandos de posicionamiento como ”configure terminal” o ”interface fc[Blade/Port]” son comandos utilizados en dispositivos de red de Cisco para acceder a diferentes modos de configuración y seleccionar interfaces específicas para su configuración.

Estos comandos permiten a los administradores de red acceder y modificar la configuración de diferentes partes del dispositivo de red, como interfaces de red, protocolos de enrutamiento, políticas de seguridad y otras configuraciones de red.

A.1.1. Entrar en la configuración del terminal

configure terminal;

Cuando se utiliza el comando ”configure terminal”, se abre el modo de configuración global en el dispositivo de red, lo que permite al usuario configurar los parámetros globales del dispositivo, como las direcciones IP, las rutas predeterminadas, la configuración de los protocolos de enrutamiento, las políticas de seguridad, etc.

A.1.2. Entrar en la configuración de un puerto

interface fc[blade/port];

Al utilizar el comando ”interface fc[blade/port]”, los administradores de red pueden seleccionar una interfaz FC específica en un dispositivo de red de Cisco y acceder a ella para realizar cambios en su configuración. Esto puede incluir la configuración de la velocidad de la interfaz, la activación o desactivación de la interfaz, la configuración de la dirección MAC de la interfaz y otros ajustes.

A.1.3. Entrar en la base de datos de device alias

device-alias database;

El comando *device-alias database;* permite acceder y gestionar la base de datos de alias de dispositivos en un switch de almacenamiento.

La base de datos de alias de dispositivos es una colección de nombres alias que se asignan a los dispositivos conectados al switch. En lugar de utilizar la dirección de este dispositivo o un número identificador, se utiliza el *device alias*, una cadena de texto de pocos carácteres útil para manipular el dispositivo de forma más intuitiva.

A.2. Comandos de cierre de puertos

Rutina completa de comandos para cerrar un puerto:

configure terminal;
interface fc[blade/port];
show interface fc[blade]/[port];
shutdown;
no switchport description;
show interface fc[blade]/[port];

A.2.1. Comprobaciones previas

configure terminal;
interface fc[blade/port];
show interface fc[blade]/[port]

El comando ”show interface fc[blade]/[port]” en un dispositivo Cisco se utiliza para mostrar la configuración y el estado de un puerto específico en una red de área de almacenamiento (SAN) que utiliza el protocolo Fibre Channel.

Este comando es útil para verificar la configuración de un puerto Fibre Channel específico, como el estado del enlace, la velocidad del enlace, la configuración de la VLAN, la configuración de QoS, etc. Antes de dar de baja el puerto, debemos comprobar que el puerto se encuentra desconectado y listo para ser dado de baja, para ello podemos buscar en el output de este comando el estado, y éste debe ser ”LINK FAILURE” o ”NOT-CONNECTED”.

Sin embargo, este comando en sí no sirve para cerrar un puerto en una red SAN. Para cerrar un puerto Fibre Channel específico, se puede utilizar el comando ”shutdown” en la configuración del puerto correspondiente. Es importante tener en cuenta que cerrar un puerto en una red SAN puede tener implicaciones significativas en el rendimiento de la red y debe hacerse con cuidado y sólo cuando sea necesario.

A.2.2. Comandos de cierre

configure terminal;
interface fc[blade/port];
shutdown;

El comando ”shutdown” en un dispositivo de red de Cisco se utiliza para desactivar un puerto de red específico en un interfaz determinado.

Es importante tener en cuenta que el comando ”shutdown” desactiva completamente el puerto, lo que significa que no se puede enviar ni recibir tráfico a través de él hasta que se vuelva a activar manualmente utilizando el comando ”no shutdown”. Si se desactiva accidentalmente un puerto mediante el comando ”shutdown”, es importante recordar volver a activarlo una vez que se hayan completado las tareas de mantenimiento o solución de problemas en ese puerto.

A.2.3. Borrar la descripción del puerto

configure terminal;
interface fc[blade/port];
no switchport description;

El comando ”no switchport description” es utilizado en la configuración de un puerto en un dispositivo de red de

Cisco para eliminar la descripción asociada con un puerto que ha sido configurado en modo switchport.

La descripción de un puerto proporciona información adicional sobre la función o ubicación del puerto, lo que puede ayudar a los administradores de red a identificar y solucionar problemas en la red. Al utilizar el comando "no switchport description", se elimina esta información de la configuración del puerto.

Es importante tener en cuenta que este comando no tiene ningún efecto sobre un puerto que no está configurado en modo switchport, ya que la descripción de ese puerto no está asociada con el comando "switchport". En resumen, el comando "no switchport description" se utiliza específicamente para eliminar la descripción asociada con un puerto configurado en modo switchport en un dispositivo de red de Cisco.

A.2.4. Comprobaciones posteriores

configure terminal;
interface fc[blade/port];
show interface fc[blade]/[port]

Volvemos a utilizar el comando "show interface fc[blade]/[port]" en la cabina para mostrar la configuración y el estado de un puerto específico para comprobar que el puerto ha sido dado de baja. Cuando el puerto se encuentra dado de bajo el estado que muestra la salida del comando es "Administratively Down".

A.3. Comandos de baja de Zoning

Rutina completa de comandos para baja de Zoning Regular:

configure terminal;
interface fc[blade/port];
show interface fc[blade]/[port]
show zoneset active vsan [san] | i [deviceAlias] ;
device-alias database;
no device-alias name [deviceAlias];
device-alias commit;
no zone name [zona] vsan [san];
zone commit vsan [san];
show zoneset active vsan [san] | i [deviceAlias] ;
end;

Rutina modificada de comandos para baja de Zoning Smart:

...
device-alias commit;
zone name [zona] vsan [san];
no member device-alias [deviceAlias] init;
zoneset activate name zs-produccion-[san] vsan [san];
zone commit vsan [san];
...

A.3.1. Comprobaciones previas

configure terminal;
interface fc[blade/port];
show interface fc[blade]/[port]
show zoneset active vsan [san] | i [deviceAlias] ;

Para procesar la baja de Zoning usaremos previamente el comando "show interface fc[blade]/[port]", al igual que en las comprobaciones de cierre de puertos, para mostrar la configuración y el estado de un puerto específico en una red de área de almacenamiento (SAN) y así poder verificar que la configuración del puerto es la correcta.

Antes de dar de baja el Zoning en un puerto en particular, debemos comprobar que dicho puerto se encuentra desconectado y listo para ser dado de baja, para ello podemos buscar en el output de este comando el estado, y éste debe ser "LINK FAILURE" o "NOT-CONNECTED".

Además, el comando "show zoneset active vsan [san] — i [deviceAlias]" es utilizado en los dispositivos Cisco para mostrar información específica sobre un alias de dispositivo dentro de una zona en un Virtual Storage Area Network (VSAN). Permite verificar la configuración de Zoning activa en una VSAN determinada (*san*) y filtrar la salida para mostrar solo las líneas que contienen el alias del dispositivo especificado (*deviceAlias*).

El output mostrado al ejecutar este último comando será cotejado en busca de ciertos patrones dentro de la información sobre los alias. Estos patrones o reglas son propias de cada política e indican características de disponibilidad o no de estos alias. Sin embargo, por motivos de confidencialidad no se indican estas reglas, pero a grandes rasgos consisten en buscar ciertas palabras clave dentro del output. Según si estas palabras son o no encontradas, se procederá con la baja del Zoning o, al contrario, se detendrá el automatismo mostrando el error correspondiente al administrador responsable.

Después de comprobar el estado del puerto y de asegurar la configuración ideal de los dispositivos, el automatismo procede a ejecutar los comandos de baja. Estos comandos de cierre variarán según si el Zoning es de tipo *Smart* o bien *Regular*.

A.3.2. Comandos de baja de Zoning Smart

El Zoning de tipo Smart nos permite agrupa un conjunto de alias bajo un mismo grupo. Esto nos permite ejecutar comandos sobre todo ese grupo además de muchas otras funciones. Esto hace más sencillo su manejo, aunque se han de ejecutar más comandos que con el Zoning Regular.

configure terminal;
interface fc[blade/port];
device-alias database;
no device-alias name [deviceAlias];
device-alias commit;
zone name [zona] vsan [san];
no member device-alias [deviceAlias] init;
zoneset activate name zs-produccion-[san] vsan [san];
zone commit vsan [san];
end;

En primer lugar, el comando *no device-alias name [deviceAlias]* se encarga de eliminar del switch el alias *deviceAlias* de la base de datos. Esto implica eliminar el alias asignado a un dispositivo de almacenamiento. Finalmente aplicamos y confirmamos los cambios realizados en la base de datos con el comando *device-alias commit;*.

Los dispositivos asociados al switch pertenecen a una zona única (de aquí el término *Zoning*) que comparten con

otros dispositivos. Para eliminarlo de la zona, se crea una zona auxiliar con el mismo nombre y después se reemplazará por la original.

Los pasos para retirarlo de esa zona son:

- *zone name [zona] vsan [san];*: Crea la zona en la red San que corresponde.
- *no member device-alias [deviceAlias] init;*: Retira el dispositivo con el alias *deviceAlias* de la zona recién creada.
- *zoneset activate name zs-produccion-[san] vsan [san];*: Activa el grupo con todas las zonas de la red San (incluída la modificada).
- *zone commit vsan [san];*: Aplicamos y confirmamos los cambios en el grupo de zonas de la red San.

A.3.3. Comandos de baja de Zoning Regular

En el caso del Zoning de tipo Regular, las zonas no se encuentran agrupadas. Entonces, se puede dar de baja la zona con un solo comando y aplicar los cambios.

configure terminal;
interface fc[blade/port];
device-alias database;
no device-alias name [deviceAlias];
device-alias commit;
no zone name [zona] vsan [san];
zone commit vsan [san];
end;

En primer lugar, el comando *no device-alias name [deviceAlias]* se encarga de eliminar del switch el alias *deviceAlias* de la base de datos al igual que en el caso anterior. De nuevo, aplicamos los cambios en la base de datos con el comando *device-alias commit;*.

Como la zona no pertenece a ningún grupo, puede ser dada de baja de forma independiente con el comando *zone name [zona] vsan [san];*.

Para terminar aplicamos los cambios en las zonas de la red San con *zoneset activate name zs-produccion-[san] vsan [san];* y confirmamos los cambios en el grupo de zonas de la red San.

A.3.4. Comprobaciones posteriores

configure terminal;
interface fc[blade/port];
show zoneset active vsan [san] | i [deviceAlias] ;

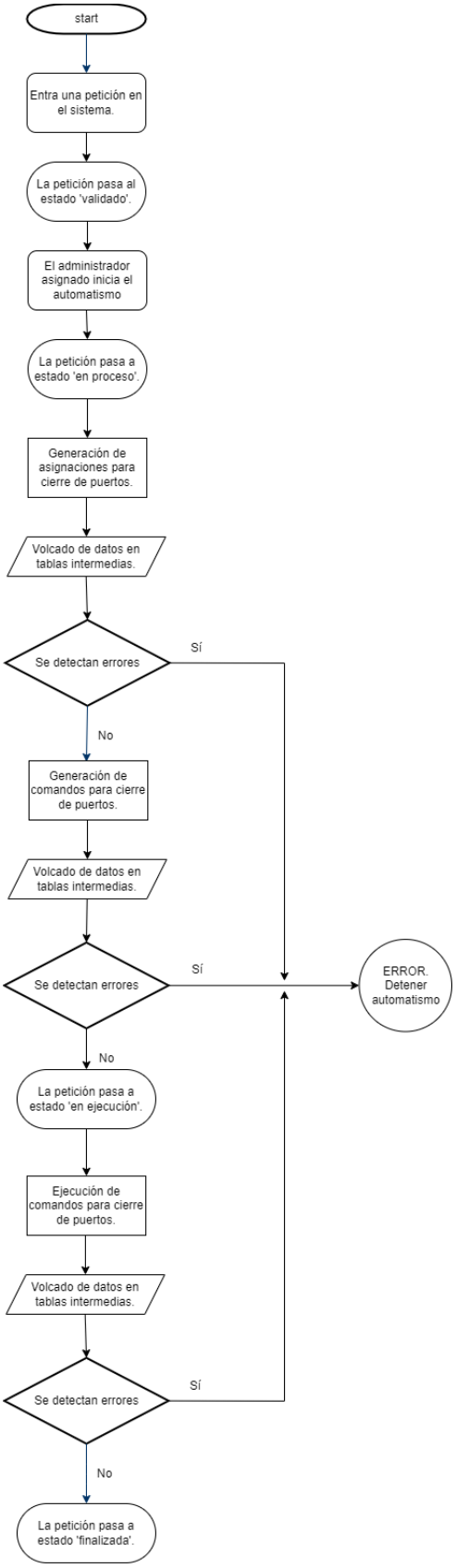
Ejecutamos este comando de nuevo, para mostrar la configuración del dispositivo que, si la baja de Zoning ha resultado exitosa, habrá sido modificada. Se analizará el output del comando para comprobar que el Zoning ha sido dado de baja correctamente.

En el caso de que la salida no fuera correcta, se detendría el automatismo, mostrando un mensaje de error y la salida del comando, para que el administrador pueda decidir como proceder.

B ANEXO 2: DIAGRAMAS DE FLUJO

B.1. Cierre de puertos

Fig. 2: Automatismo de cierre de puertos.



B.2. Baja de Zoning

Fig. 3: Automatismo de cierre de puertos.