LIBRO CISCO CAPITULO 10

**Desafíos de Automatización en Redes:**

* **Escalabilidad:** A medida que las redes crecen en tamaño y complejidad, la administración manual de dispositivos se vuelve impracticable. La automatización ayuda a gestionar y configurar grandes volúmenes de dispositivos sin intervención manual constante.
* **Diversidad de Equipos:** Las redes modernas incluyen una amplia variedad de dispositivos y tecnologías, desde routers y switches hasta firewalls y puntos de acceso inalámbricos. La automatización ayuda a estandarizar la gestión de estos dispositivos diversos mediante plantillas y modelos.
* **Interacción Manual:** Las interacciones manuales con la red a través de interfaces de línea de comandos (CLI) son propensas a errores y no son escalables. La automatización reduce estos errores y mejora la consistencia.
* **Velocidad y Eficiencia:** En entornos de alta demanda, la velocidad de provisión de servicios y cambios en la red debe ser rápida. La automatización permite realizar cambios de forma rápida y eficiente, algo crucial para las operaciones de red ágiles.
* **Gestión de Configuración:** La automatización ayuda a gestionar las configuraciones de red de manera más eficaz, aplicando políticas de manera uniforme y evitando inconsistencias entre dispositivos.

**Redes Definidas por Software (SDN)**

* **Definición y Propósito:** Las Redes Definidas por Software (SDN) son un enfoque que separa el plano de control del plano de datos en redes. Esto permite una gestión más centralizada y flexible de la red. SDN proporciona una visión global de la red, facilitando la programación y la automatización.
* **Génesis de SDN:** SDN surgió como una respuesta a la creciente complejidad de las redes y la necesidad de una gestión más dinámica. Permitió la virtualización de redes y la creación de redes lógicas sobre una infraestructura física común.
* **Valor Obtenido:** SDN mejora la eficiencia operativa al permitir la automatización y la programación centralizada. Facilita la implementación de políticas de red de manera consistente y simplifica la gestión de redes complejas.

**3. Interfaces de Programación de Aplicaciones (API)**

* **Definición de API:** Las API (Interfaces de Programación de Aplicaciones) permiten que diferentes aplicaciones se comuniquen entre sí. En el contexto de redes, las API permiten la automatización y la integración de herramientas de gestión y monitoreo.
* **Beneficios de las API:** Las API facilitan la integración de diferentes sistemas, permiten la automatización de tareas repetitivas y reducen la necesidad de interacción manual. Permiten a los administradores de red crear scripts y herramientas personalizados para manejar tareas específicas.
* **Resistencia al Cambio:** A pesar de los beneficios, algunas organizaciones aún resisten el cambio hacia el uso de API debido a la falta de familiaridad o la complejidad percibida.

**4. API REST**

* **Definición y Funcionamiento:** REST (Representational State Transfer) es un estilo arquitectónico para diseñar servicios web que utiliza HTTP y métodos estándar como GET, POST, PUT y DELETE para interactuar con recursos. Las API REST son ampliamente utilizadas en la programación moderna debido a su simplicidad y compatibilidad con HTTP.
* **Beneficios de REST:** Las API REST son ligeras y fáciles de usar. Permiten la comunicación entre sistemas de forma simple y eficiente, utilizando formatos de datos estándar como JSON o XML.
* **Uso Común:** Las API REST se utilizan para una variedad de tareas, desde consultar datos hasta realizar operaciones complejas en aplicaciones web y móviles.

**5. Orquestación entre Dominios e Independiente de la Tecnología**

* **Orquestación:** La orquestación se refiere a la coordinación y gestión de múltiples componentes o sistemas. En el contexto de redes, la orquestación puede implicar la gestión de diferentes tecnologías y dominios de red, garantizando que trabajen juntos de manera eficiente.
* **Independencia de Tecnología:** La orquestación debe ser independiente de las tecnologías subyacentes para ser efectiva. Esto permite que diferentes tecnologías y plataformas se integren sin problemas y se gestionen de manera uniforme.
* **Consideraciones para la Transformación:** Aunque esta sección no está incluida en todas las certificaciones, entender la orquestación es importante para la transformación digital y la integración de nuevas tecnologías en redes.

**6. Impacto en la Seguridad y la Gestión de Servicios de TI**

* **Seguridad:** La automatización puede influir en la seguridad de la red, ya que una mala configuración o errores en los scripts pueden introducir vulnerabilidades. Es crucial implementar buenas prácticas de seguridad al automatizar procesos.
* **Gestión de Servicios de TI:** La automatización afecta la gestión de servicios de TI al permitir una supervisión y administración más proactiva. La integración con marcos de gestión como ITIL y TOGAF ayuda a alinear las operaciones de TI con las mejores prácticas y estándares de la industria.

La automatización en redes se enfrenta a desafíos significativos debido al crecimiento y la complejidad de las infraestructuras. SDN y las API, incluyendo REST, juegan un papel crucial en la simplificación y estandarización de la gestión de redes. La orquestación y la seguridad son consideraciones adicionales importantes, especialmente cuando se implementan nuevas tecnologías y metodologías ágiles en el entorno de TI.

**1. Escala en Redes**

**Desafíos de la Escala en Redes:**

La escalabilidad en redes se refiere a la capacidad de una red para manejar el crecimiento en tamaño, tráfico y número de usuarios sin sacrificar el rendimiento ni la eficiencia. A medida que una red se expande, los enfoques manuales, como la configuración y el mantenimiento a través de CLI (Command Line Interface), pueden volverse inadecuados.

**Problemas con CLI y Métodos de Pegado:**

* **Control de Flujo y Eficiencia:** Las metodologías basadas en CLI, especialmente cuando se utilizan para pegar grandes bloques de texto de configuración, a menudo enfrentan problemas de control de flujo. Esto puede causar lentitud y errores en la implementación debido a la necesidad de esperar que cada comando se procese antes de continuar con el siguiente.
* **Ejemplo del Cliente:** En el ejemplo que compartiste, un cliente enfrentaba problemas con listas de acceso de seguridad extensas. La configuración manual a través de CLI y la copia y pegado de grandes cantidades de datos llevaron a largos tiempos de implementación. La alternativa fue transferir y fusionar archivos de configuración, lo que resultó en una mejora significativa en la eficiencia.

**Uso de Protocolos NETCONF/RESTCONF:**

* **Protocolos NETCONF/RESTCONF:** Estos protocolos permiten una gestión programática de la red, lo que mejora la eficiencia en comparación con la CLI. NETCONF (Network Configuration Protocol) y RESTCONF (RESTful Configuration Protocol) permiten realizar cambios y recopilar información de manera más eficiente mediante el uso de interfaces programáticas.

**Preguntas para Evaluar Escala y Automatización:**

* **Número de Dispositivos y Servicios:** Evaluar cuántos dispositivos, nodos y servicios se deben implementar ayuda a planificar la escala de la automatización.
* **Dependencias y Escalonamiento:** Identificar dependencias entre dispositivos y servicios es crucial para implementar cambios de manera que se minimicen los riesgos.
* **Ventanas de Cambio:** Considerar el tiempo disponible para implementar cambios es importante, especialmente si la red no tiene ventanas de mantenimiento.
* **Capacidad de Reversión:** Planificar cómo revertir cambios en caso de errores inesperados es esencial para mantener la estabilidad de la red.

**Manejo del Riesgo:**

* **Evaluación del Riesgo:** Determinar cuánto riesgo estás dispuesto a asumir es crucial. Realizar cambios en etapas y realizar pruebas exhaustivas ayuda a minimizar el impacto de problemas inesperados.

**Enfoque por Fases:**

* **Pruebas y Implementación por Fases:** Implementar cambios en etapas y realizar pruebas en entornos controlados permite identificar problemas antes de una implementación completa. Este enfoque reduce el riesgo y asegura una transición más suave.

**2. Concurrencia y Paralelismo**

**Concurrencia:**

* **Definición:** La concurrencia se refiere a la capacidad de un sistema para gestionar múltiples tareas en progreso al mismo tiempo, aunque no necesariamente de manera simultánea. Las tareas se inician, ejecutan y completan en períodos de tiempo superpuestos.
* **Ejemplo de Concurrencia:** Imagina un proceso de actualización de múltiples dispositivos donde cada dispositivo se configura secuencialmente, pero algunos procesos pueden comenzar antes de que otros hayan terminado. Esto es adecuado cuando las tareas no tienen dependencia estricta entre sí.

**Paralelismo:**

* **Definición:** El paralelismo implica la ejecución simultánea de múltiples tareas, utilizando varios núcleos de CPU o procesadores. A diferencia de la concurrencia, el paralelismo se enfoca en realizar múltiples tareas al mismo tiempo en paralelo.
* **Ejemplo de Paralelismo:** En un sistema con múltiples núcleos, puedes configurar varios dispositivos simultáneamente, cada uno en un núcleo diferente. Esto acelera el proceso al aprovechar el hardware disponible para ejecutar tareas en paralelo.

**Aplicación Práctica:**

* **API Meraki Dashboard:** Al usar la API Meraki, el paralelismo puede mejorar la eficiencia al enviar múltiples solicitudes API simultáneamente en lugar de procesarlas de manera secuencial. Esto reduce el tiempo total necesario para completar todas las solicitudes.

**Uso de Asyncio en Python:**

* **Asyncio y Semáforos:** La biblioteca asyncio en Python permite realizar operaciones de entrada/salida de manera asíncrona, mientras que los semáforos ayudan a limitar el número de tareas concurrentes. Esto es útil para manejar múltiples solicitudes a una API de manera eficiente.

**3. Automatización y Eficiencia**

**Presiones Comerciales:**

* **Centro de Costos vs. Centro de Ganancias:** Las empresas pueden ver la TI de red como un centro de costos, buscando reducir gastos, o como un centro de ganancias, buscando mejorar márgenes. La automatización ayuda a ambos enfoques al reducir costos operativos y aumentar la eficiencia.

**Automatización en Redes:**

* **Ventajas de la Automatización:** La automatización reduce la carga de trabajo manual, minimiza errores y permite a los equipos de TI concentrarse en tareas más estratégicas. La medición del tiempo y los costos ahorrados a través de la automatización proporciona una justificación sólida para su implementación.

**ROI de la Automatización:**

* **Medición del ROI:** Proporcionar informes sobre los ahorros de tiempo y costos asociados con la automatización ayuda a los tomadores de decisiones a comprender los beneficios y justificar la inversión en herramientas y procesos automatizados.

**4. Redes Definidas por Software (SDN)**

**Origen y Concepto:**

* **Programa Clean Slate:** El Programa Clean Slate de la Universidad de Stanford en 2008 impulsó la idea de SDN, replanteando cómo se podrían diseñar redes desde cero, sin las limitaciones de las tecnologías existentes.
* **Definiciones de SDN:**
  + **Desacoplamiento del Plano de Control y Datos:** SDN separa la toma de decisiones sobre el tráfico (plano de control) de la transmisión real de datos (plano de datos), centralizando la inteligencia y el control.
  + **Virtualización de Red:** La infraestructura de red se abstrae, permitiendo a las aplicaciones gestionar la red sin preocuparse por los detalles del hardware subyacente.
  + **Interfaces Programáticas:** Las API permiten a los sistemas externos interactuar con la red, facilitando la provisión y la gestión de servicios.

**Transformación de Redes:**

* **Impacto de SDN:** SDN transforma la forma en que se diseñan, implementan y gestionan las redes. Ofrece nuevas formas de interactuar con el hardware y los servicios, proporcionando flexibilidad y escalabilidad.

**Comparación con Tecnologías Anteriores:**

* **Tecnologías Preexistentes:** Aunque SDN representa un avance significativo, no es el primer intento de transformar las redes. Tecnologías anteriores como MPLS, VPN y SNMP también han contribuido a la evolución de la red.

**Entidades No Tradicionales y SDN**

**Entidades No Tradicionales en Redes**

Las **entidades no tradicionales** en el contexto de redes se refieren a dispositivos, aplicaciones o servicios que no eran comunes en las redes tradicionales. Estas entidades incluyen aplicaciones de red programables, dispositivos de red con interfaces API, y soluciones que no se ajustan al modelo clásico de hardware de red.

**Impacto de las Entidades No Tradicionales en la Red**

1. **Programación de Redes**: La llegada de nuevas interfaces programáticas permitió a los programadores que no eran ingenieros de redes tradicionales comenzar a desarrollar aplicaciones que interactúan con redes. Este cambio aumentó la demanda de habilidades en programación de redes y permitió a los desarrolladores diseñar redes más adaptadas a sus aplicaciones específicas en lugar de depender de equipos de aprovisionamiento de red.
2. **Seguridad**: Con la introducción de nuevas interfaces y APIs, la seguridad se convirtió en una preocupación mayor. Las nuevas interfaces necesitan ser protegidas con políticas de seguridad robustas para evitar daños no intencionales por parte de usuarios inexpertos y ataques maliciosos.

**Impacto en la Industria**

La **separación del plano de control y el plano de datos** revolucionó la industria de redes. Los **controladores centralizados** como el Cisco ACI APIC permitieron una administración más eficiente al centralizar la supervisión y configuración de la red, en lugar de manejar cada dispositivo por separado. Esto facilitó la gestión de redes grandes y complejas, permitiendo una configuración y monitoreo unificado.

**Nuevos Métodos y Protocolos**

La adopción de SDN trajo consigo nuevos métodos y protocolos:

1. **API y Protocolos**: La proliferación de APIs como NETCONF y RESTCONF ha sido significativa. Estas APIs permiten una comunicación más eficiente con los dispositivos de red y facilitó la administración y configuración centralizada.
2. **Interfaces de Red Centralizadas**: Las interfaces y controladores centralizados (por ejemplo, Cisco ACI) han demostrado ser eficaces para la administración de redes, haciendo más sencillo el aprovisionamiento y monitoreo de dispositivos a gran escala.

**Normalización**

SDN ha impulsado la **normalización** en la red al estandarizar las interfaces y protocolos de comunicación. La especificación OpenAPI, derivada del proyecto Swagger, ha simplificado la creación y documentación de APIs REST, proporcionando una base más consistente para la interacción entre dispositivos y aplicaciones.

**Habilitación de Operaciones**

La **automatización y la programación** han transformado las operaciones de red, permitiendo a los operadores utilizar métodos basados en API REST y scripts para manejar la red en lugar de depender de métodos manuales. La introducción de tecnologías como gRPC, OpenConfig, y YANG también ha contribuido a una administración más eficiente y menos propensa a errores.

**Habilitación de Opciones Profesionales**

SDN ha abierto nuevas oportunidades para los **ingenieros de redes tradicionales**, permitiéndoles aplicar su experiencia a través de interfaces programáticas y mejorar su capacidad para implementar servicios a gran escala.

**Impacto de SDN**

El impacto de SDN en la industria de redes puede ser comparado con el de la telefonía IP, donde la integración entre telecomunicaciones y conmutación de paquetes fue un desafío. La habilidad para combinar conocimientos tradicionales con nuevas tecnologías es crucial para el éxito de SDN.

**Casos de Uso y Problemas Resueltos con SDN**

**Casos de Uso de SDN**

SDN ha abordado una variedad de casos de uso en diferentes sectores:

1. **Academia e Investigación**: Permite experimentar con nuevos protocolos de enrutamiento y redes sin alterar las normas establecidas.
2. **Centros de Datos**: Facilita la separación del plano de control y datos, proporcionando visibilidad profunda y gestión de tráfico.
3. **Proveedores de Nube**: Ofrece aprovisionamiento automatizado y redes programables.
4. **Proveedores de Servicios**: Permite el control y análisis basados en políticas para optimizar y monetizar servicios.
5. **Redes Empresariales**: Ofrece virtualización, segmentación y orquestación de perfiles de seguridad.

**Beneficios de SDN**

1. **Configuración Centralizada**: Facilita el control de todos los dispositivos de red desde un único punto.
2. **Adaptación de Algoritmos**: Permite personalizar los algoritmos de reenvío según necesidades específicas.
3. **Integración Externa**: Las aplicaciones externas pueden influir en la red.
4. **Implementación Escalable**: Mejora la velocidad y precisión en la implementación de servicios de red.

**Protocolos y Soluciones Relacionados**

* **OpenStack**: Virtualización de cargas de trabajo.
* **OMI**: Normalización de la gestión con un modelo común.
* **Puppet y Ansible**: Gestión de configuración basada en agentes y sin agentes, respectivamente.
* **NETCONF, YANG**: Normalización de configuración y modelado de datos.

**Descripción General de los Controladores de Red**

**Plano de Control vs. Plano de Datos**

* **Plano de Control**: Toma decisiones sobre cómo se deben reenviar los datos. Protocolos como OSPF y BGP operan en este plano.
* **Plano de Datos**: Ejecuta las decisiones de reenvío. Protocolos como LACP y tablas de direcciones MAC operan aquí.

**Separación de Planos**

La separación de estos planos permite centralizar la toma de decisiones en un controlador mientras que los dispositivos manejan el reenvío de datos. Esto ha permitido una administración más eficiente y la introducción de nuevas funciones como superposiciones de red y host.

**Ejemplos de Controladores y Superposiciones**

* **Cisco ACI**: Utiliza el controlador APIC para la administración centralizada de dispositivos Nexus.
* **OTV, VXLAN**: Ejemplos de superposiciones que proporcionan túneles virtuales y aislamiento de tráfico.

**Interfaces de Programación de Aplicaciones (API)**

**API en Redes**

Las **APIs** permiten interactuar con dispositivos y aplicaciones de red. Aunque la interfaz de línea de comandos (CLI) era la norma, las APIs han permitido una interacción más flexible y programática.

**Tipos de API**

1. **REST (Representational State Transfer)**: Utiliza HTTP para operaciones de lectura y escritura, es sin estado y permite una comunicación sencilla entre sistemas.
2. **SOAP (Simple Object Access Protocol)**: Utiliza XML y es más estructurado en comparación con REST.

**Métodos de API REST**

* **GET**: Recupera datos.
* **POST**: Envía datos para crear nuevos recursos.
* **PUT**: Actualiza datos existentes.
* **DELETE**: Elimina datos.

**Autenticación y Seguridad**

Las APIs REST pueden utilizar diversos métodos de autenticación como autenticación básica, tokens de portador y OAuth. La seguridad es crucial para proteger las interacciones con la API.

**Paginación**

La paginación ayuda a manejar grandes volúmenes de datos limitando la cantidad de datos devueltos en una sola solicitud. Esto evita sobrecargar al servidor y permite una gestión más eficiente de los resultados.

**Formatos de Datos: JSON y XML**

* **JSON (JavaScript Object Notation)**: Formato ligero y fácil de leer que se utiliza ampliamente para la comunicación entre sistemas.
* **XML (eXtensible Markup Language)**: Más estructurado y verbose, utilizado para describir y compartir datos de manera programática.

**1. JSON (JavaScript Object Notation)**

**JSON** es un formato ligero de intercambio de datos que es fácil de leer y escribir para los humanos y fácil de analizar y generar para las máquinas. JSON se basa en una notación de objetos JavaScript, pero es independiente del lenguaje de programación, lo que lo hace muy versátil y popular en diversos contextos, incluyendo el desarrollo web y las APIs.

**Características de JSON:**

* **Sintaxis Simple:** JSON utiliza una sintaxis de pares nombre-valor que es fácil de entender. La estructura básica consiste en objetos (también llamados registros o diccionarios) y arreglos (listas o matrices).
* **Objetos:** Se definen entre llaves { } y contienen pares nombre-valor separados por comas. El nombre debe estar entre comillas dobles, mientras que el valor puede ser una cadena (entre comillas dobles), un número, un booleano (true/false), un arreglo, otro objeto o null.
* **Tipos de Valores en JSON:**
  + **Cadenas:** Deben estar entre comillas dobles. Ejemplo: "texto".
  + **Números:** No requieren comillas. Ejemplo: 123.
  + **Booleanos:** true o false, sin comillas.
  + **Null:** Representado por null, sin comillas.
  + **Objetos:** Definidos entre { }.
  + **Arreglos:** Definidos entre [ ].

**Ventajas de JSON sobre XML:**

* **Menor Peso:** JSON suele ser más compacto y menos verbose comparado con XML.
* **Facilidad de Lectura y Escritura:** La sintaxis es más simple y fácil de comprender.
* **Integración Natural con JavaScript:** JSON se integra perfectamente con JavaScript, haciendo más sencilla la manipulación de datos en aplicaciones web.

**Uso en APIs:**

* **Negociación de Contenidos:** Muchas APIs permiten negociar el formato de los datos. Los servidores pueden enviar datos en formato JSON o XML, dependiendo de los encabezados de la solicitud (como Accept) o la extensión del URI (como .json o .xml).

**2. Orquestación Multidominio e Independiente de la Tecnología (CDTAO)**

La **orquestación multidominio** se refiere a la capacidad de coordinar y gestionar múltiples dominios tecnológicos desde una perspectiva unificada. Esto incluye integrar diversas herramientas y aplicaciones para obtener una visión consolidada de la infraestructura de TI.

**Desafíos en la Gestión Multidominio:**

* **Diversidad de Herramientas:** Las herramientas y tecnologías utilizadas en diferentes dominios (redes, almacenamiento, seguridad, etc.) pueden ser variadas y no siempre se integran bien.
* **Visión Fragmentada:** Puede ser difícil obtener una visión global del estado y el rendimiento de la infraestructura si se utilizan múltiples herramientas que no se comunican entre sí.

**Beneficios de la Orquestación:**

* **Visión Unificada:** Permite consolidar información de diferentes herramientas en un único panel de control.
* **Automatización y Eficiencia:** Facilita la automatización de tareas y procesos que abarcan múltiples dominios tecnológicos.

**Uso de APIs en la Orquestación:**

* **Interoperabilidad:** Las APIs proporcionan un mecanismo para conectar y comunicar diferentes herramientas y sistemas. Permiten que las herramientas se integren y compartan datos y funcionalidades.
* **Ejemplo de API en Orquestación:** Un sistema de orquestación puede usar APIs para obtener datos de estado de un sistema de red, controladores de almacenamiento, y herramientas de seguridad, y luego consolidar esa información en un panel de control unificado.

**Aplicaciones de la Orquestación Multidominio:**

* **Salud:** Integración de sistemas conectados como monitores de presión arterial y equipos de emergencia.
* **Finanzas:** Gestión de cajeros automáticos y sistemas de bóvedas.
* **Comercio Minorista:** Supervisión de terminales de tarjetas de crédito y equipos de punto de venta.
* **Educación:** Control de equipos como proyectores y cámaras.

**3. Impacto en la Seguridad y la Gestión de Servicios de TI**

El **impacto en la seguridad y la gestión de servicios de TI** se refiere a cómo las nuevas tecnologías, como las redes definidas por software (SDN) y la programación de redes, afectan la seguridad y la administración de la infraestructura de TI.

**Seguridad en Entornos Programables:**

* **Riesgos de Exposición:** Las interfaces programáticas (APIs) y los controladores tienen un alcance más amplio y pueden ser objetivos para ataques si no se protegen adecuadamente.
* **Prácticas de Seguridad:**
  + **Protección de Controladores:** Asegurarse de que los controladores y sistemas programables estén protegidos mediante autenticación fuerte y control de acceso.
  + **Registro y Monitoreo:** Implementar y revisar el registro y la contabilidad para detectar actividades sospechosas y asegurar la integridad de las operaciones.

**Prácticas de Seguridad Específicas:**

* **Interfaces Programáticas:** Asegúrese de que todas las interfaces tengan controles de acceso adecuados y se mantengan seguras.
* **Equipos Críticos:** Priorizar la seguridad en los equipos de alta prioridad cuya disponibilidad impacte directamente en el servicio o en los ingresos.

JSON es una herramienta esencial para el intercambio de datos en aplicaciones modernas debido a su simplicidad y compatibilidad. La orquestación multidominio y la gestión de servicios de TI deben considerar la integración y seguridad de diversos sistemas y herramientas para lograr una visión unificada y segura.