



Business and technology working as one

NEP@L

Network Programmability

Abstraction Layer

Network Programmability Abstraction Layer

Introducción

La automatización en el ambiente tecnológico, como la mayoría de los tipos de automatización, es pensada o considerada como una manera de “hacer las cosas más rápido”. Mientras que hacer las cosas más rápido es bueno, reducir el tiempo de las implementaciones y los cambios de configuración no siempre representan un problema prioritario ni que debe resolverse.

NEP@L® permite escribir código para resolver las necesidades de programabilidad y automatización, considerando los estándares de la industria open source mas destacados.

Definición

Conceptualmente, “NEP@L® es un framework de desarrollo que brinda respuesta a una variedad de problemas a los que se enfrenta una organización cuando decide implementar automatización”.

Compuesta por los módulos NEP@L Network® y NEP@L IT®, la arquitectura model-driven de NEP@L® comprende una serie de componentes estratégicamente relacionados para optimizar y potenciar el éxito de las tareas de automatización, a fin de agilizar el time-to-market, aumentar la confiabilidad, mantener los sistemas bajo compliance y brindar mejoras continuas de forma documentada.

Desde el punto de vista del usuario, NEP@L® es el punto de entrada para la formulación, definición y resolución de casos de usos de automatización, y en el otro extremo, obtener los resultados deseados y necesarios, llamados beneficios



Figura 1. Flujo de información en NEP@L

Network Programmability Abstraction Layer

Arquitectura

NEP@L[®] fue construido siguiendo una arquitectura totalmente modular. Está compuesta por proyectos (Templating, On-Boarding, Compliance, Roll Back, etc), por APIs de interconexión (norte y sur), un conector hacia el este para integración con la plataforma de gestión de Logicalis, y un conector hacia el oeste para conectar con orquestadores y plataformas de automatización. Esta concepción es la que permite a NEP@L tener un alto nivel de integración, un diferencial fundamental cuando se lo compara con soluciones similares.

La metodología de integración continua (CI/CD) actúa como un orquestador de funcionamiento de todos los módulos al momento de la generación de código, integración, testing y delivery.

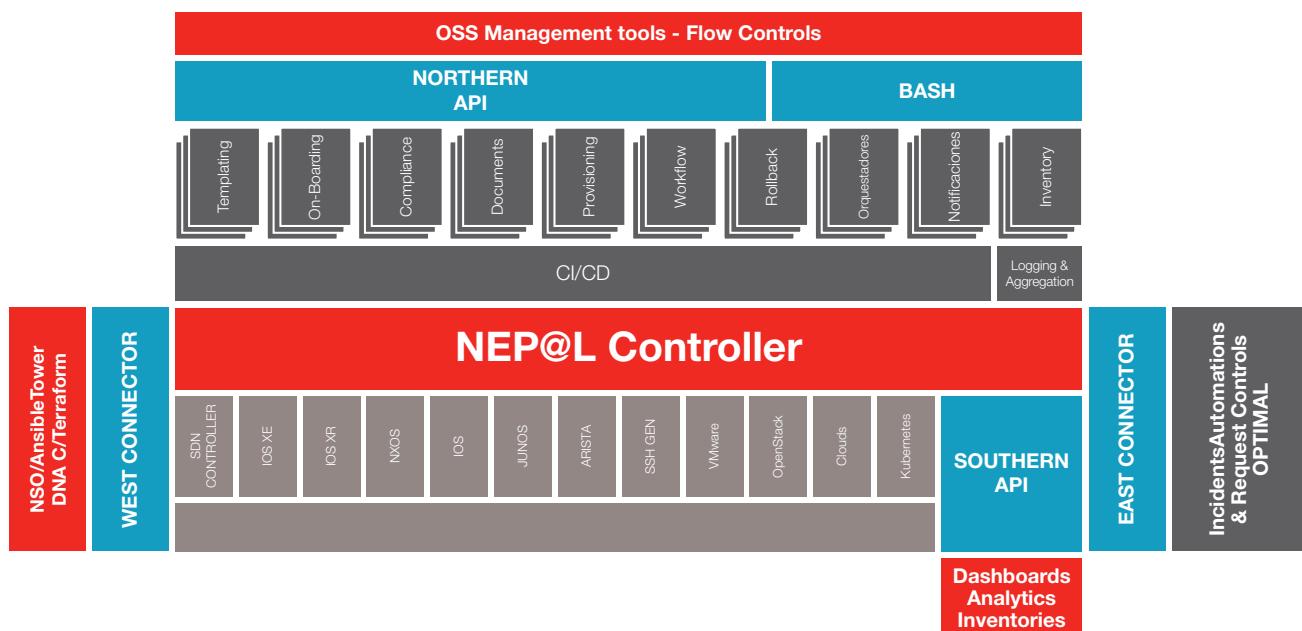
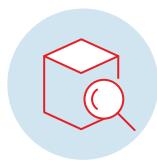


Figura 2. Arquitectura del framework NEP@L

Los proyectos comprenden piezas de código construidas en los distintos lenguajes que componen el framework (Ansible, Python, Jinja2, NodeJS, Puppet, etc.). Las APIs se construyen según la necesidad de cada caso. Esta visión programática no solo permite interactuar con las plataformas más conocidas (Cisco, Juniper, Arista, SDN Controllers, Nestconf, Restconf, etc.), sino que también con cualquier plataforma gestionable vía SSH o API (http/Non http), y con otras plataformas de automatización o controllers, tales como ServiceNow, Cisco NSO, Ansible Tower, DNA Center, Terraform y NEP@L Control.



Consultoría

Al ser un servicio, Logicalis NEP@L® tiene como objetivo reducir los tiempos de respuesta que la tecnología impone al negocio, al mismo tiempo que aumenta la confiabilidad. Por este motivo, el primer paso debe ser un proceso consultivo que entienda tanto el estadio tecnológico del cliente como así también los casos de uso de valor para automatizar. Una vez detectados dichos casos de uso, se avanzará a la siguiente etapa teniendo en cuenta el contexto tecnológico del cliente.



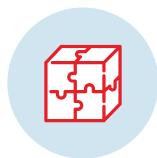
Planeamiento

El siguiente paso es lograr que los pre-requisitos tecnológicos sean debidamente cumplimentados, algo que seguramente implicará un proceso de adaptación. El objetivo es realizar los cambios e implementaciones en la infraestructura actual del cliente, para llevar adelante los procesos de automatización requeridos.



Codificación

La generación de templates es crucial para en el proceso de automatización. A través de estos, se definirá el conjunto de bloques de código de configuración necesarios. En esta etapa también se definirán los modelos de servicios o parámetros de compliance a seguir. Este enfoque permite mantener o preservar lineamientos durante todo el proceso de automatización y brindar los reportes necesarios.



Testing

Antes del despliegue, todo el código generado pasa por un proceso de testing virtual, durante el cual se revisan no solo la sintaxis y la semántica, sino también la configuración o despliegue, documentación y delivery del caso de uso. Solo en caso de que el proceso completo de testing (Integración Continua) finalice apropiadamente, el código de automatización quedará disponible para su despliegue.



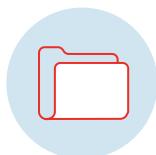
Despliegue

Durante esta etapa es donde se impactan la configuraciones definidas durante el proceso de codificación. Solo durante este pequeño lapso, la infraestructura involucrada queda expuesta a posible acciones disruptivas (network/data center outage). Durante la etapa de despliegue, no solo se activan los servicios requeridos o descriptos en el caso de uso, sino que también se despliega la estrategia de rollback mas apropiada.



Mejora continua

En los procesos de automatización es importante que se detecten de forma constante las oportunidades de mejora de dichos procesos para brindar mayores beneficios para el negocio. Es por esto que NEP@L[®] considera a la mejora continua como uno de sus pilares. Los módulos de consultoría y codificación, como parte integrantes del framework de desarrollo, están concebidos para satisfacer esta demanda.



Documentación

Uno de los mayores desafíos de las compañías dinámicas desde el punto de vista tecnológico es, sin dudas, estar en un estadio de constante evolución y desarrollo, manteniendo en todo momento un marco de documentación apropiado de los procesos de cambios. Para ello, y mediante la integración de distintas herramientas, NEP@L[®] mantiene un marco documental controlado y actualizado durante la misma etapa de deployment.

Network Programmability Abstraction Layer

NEP@L

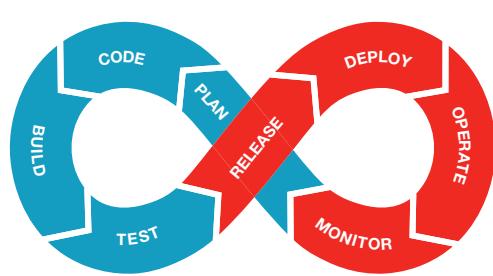
conocimiento, tecnologías y metodologías

Desde una perspectiva técnica, NEP@L® brinda sus servicios sobre una variedad de tecnologías que gozan de un alto nivel de adopción, lo que permite mantener la inversión realizada por nuestros clientes.

Haciendo uso de las más avanzadas herramientas de automatización, programación, versionado, repositorios, integración continua, notificación, simulación y documentación, NEP@L® transforma el paradigma de configuración en programación.

Integración Continua (CI/CD)

La integración continua es una práctica común especialmente en el ámbito del desarrollo de software. El objetivo de este moderno enfoque es trabajar en pasos con el fin de lograr un proceso de desarrollo más efectivo, poder reaccionar con más flexibilidad ante los cambios y optimizar el time-to-market.



De manera innovadora, NEP@L® incorpora esta técnica tanto al contexto de las redes como al de data center. Cada vez que un ingeniero cambia o crea una porción de código, que será utilizada para resolver un caso de uso de automatización, dicha pieza de código es sometida a las etapas del proceso CI/CD. Solo si todas las etapas que componen el proceso (code review, implementación, testing, delivery, etc) sin error, el mismo estará disponible para su implementación o deployment.

Todo el proceso de CI/CD se realiza de forma virtual y no hay ningún impacto sobre la infraestructura real. Por tal motivo, la probabilidad de que se produzcan errores al momento de deployment sobre la infraestructura real, se reduce a su mínima expresión. Desde el punto de vista de NEP@L®, CI/CD es escribir software que verifique software.

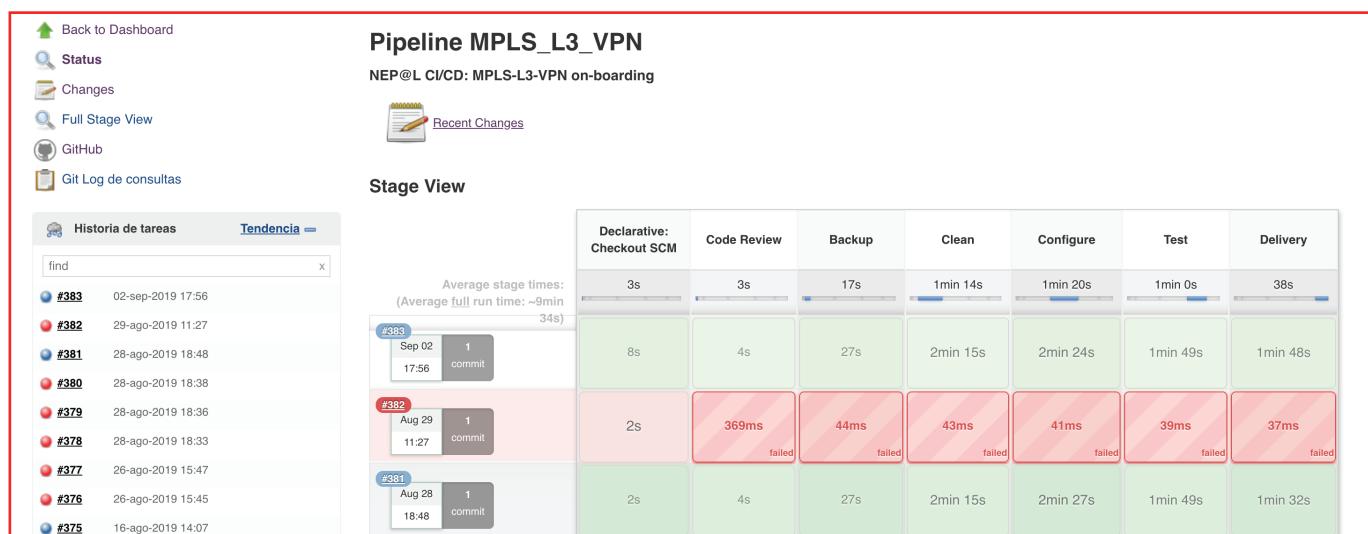


Figura 3. Pipeline CI/CD aplicado a proceso de MPLS-L3_VPN

Network Programmability Abstraction Layer

1. Integración de tecnologías

El proceso completo de integración de tecnologías, que definen el funcionamiento de NEP@L®, se muestra en la figura 4.

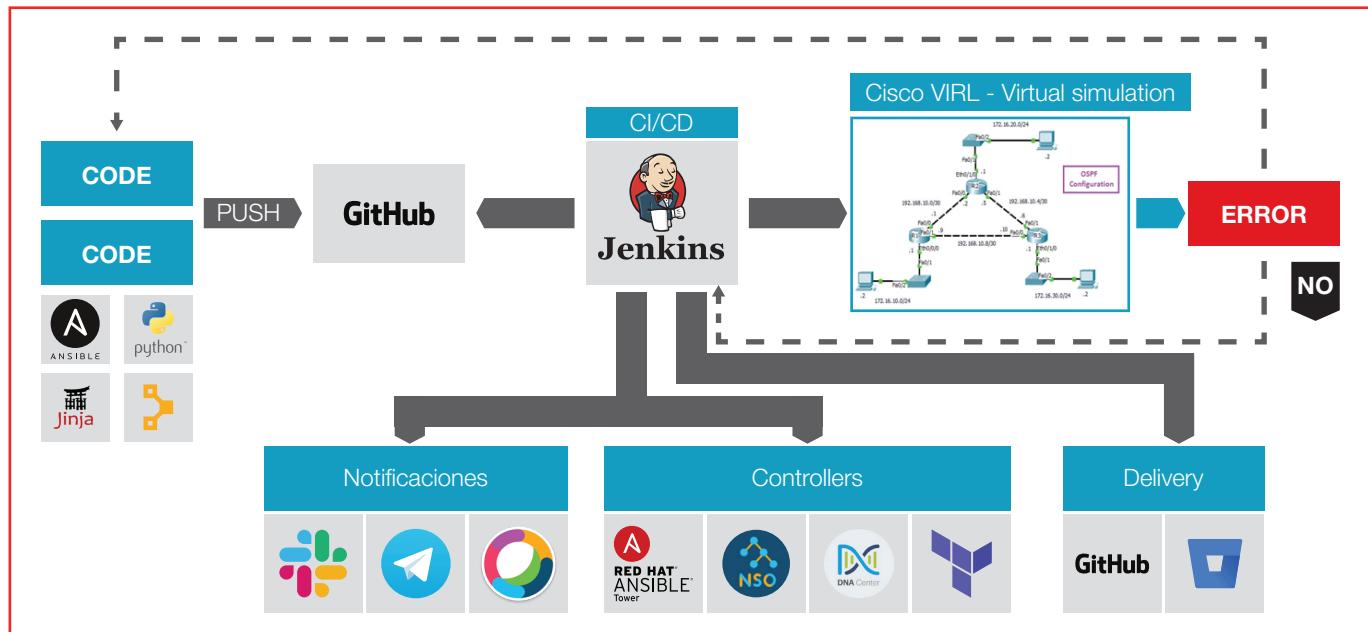


Figura 4. NEP@L: Tecnologías integradas

Secuencia de integración

1.1. El ingeniero desarrolla el código necesario para resolver el requerimiento de programabilidad y/o automatización y submite el mismo a repositorios Git (GitHub/GitLab). El repositorio garantiza la integridad del código en cuanto a versionado y a distintos sitios/unidades de desarrollos (branches). Los lenguajes que forman parte de NEP@L® son Ansible, Python, Jinja2 y Puppet.

1.2. Una vez que el código está en el repositorio, es tomado por el nodo central de NEP@L®: la herramienta de integración continua CI/CD, Jenkins.

1.3. Jenkins aplica el código diseñado para testear la solución completa, utilizando para esto un simulador de networking llamado VIRL y distintas VMs. Si hubiera algún error, el proceso vuelve al inicio para que los desarrolladores lo corrijan.

1.4. Una vez libre de errores, Jenkins deposita el código en repositorios en formato de librería, listo para su aplicación en la fase llamada deployment. Esta etapa se denomina Delivery.

1.5. Una vez finalizado el proceso de integración, Jenkins crea en distintos controllers tales como AnsibleTower, Cisco NSO (Network Services Orchestrator), DNA C, Terraform y NEP@L® Controller los objetos necesarios para su ejecución.

1.6. Finalmente, Jenkins notifica al grupo de personas involucradas en la codificación del caso de uso de programabilidad o automatización sobre los resultados de la integración.

2. Deployment

Los ingenieros ejecutan el código depositado en Ansible Tower y/o NSO. Como el código pasó previamente por el proceso de integración continua, está libre de errores y, por lo tanto, la probabilidad de que exista un problema en esta etapa es prácticamente nula.

3. Conclusión

La concepción de la configuración tanto de los equipos de networking como de data center como código (network e infraestructura as a code), sumada a la adopción de metodologías y tecnologías antes mencionadas, y la definición de técnicas de desarrollo de software específicas, definen conceptual y funcionalmente a NEP@L®.

NEP@L® es la herramienta a través de la cual Logicalis se transforma y adhiere en un todo a la cultura DevOps Infrastructure y NetDevOps.



Business and technology working as one