ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

AUTOMATIZACIÓN DE REDES UTILIZADAS PARA EOT

AUTOMATIZACIÓN DE LA GESTIÓN DE REDES UTILIZADAS EN EOT

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERA EN TECNOLOGÍAS DE LA INFORMACIÓN

IBETH VERÓNICA BASTIDAS TOSCANO

ibeth.bastidas@epn.edu.ec

DIRECTOR: CARLOS ROBERTO EGAS ACOSTA

carlos.egas@epn.edu.ec

DMQ, SEPTIEMBRE DE 2022

CERTIFICACIONES

Yo, Ibeth Verónica Bastidas Toscano declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

IBETH VERONICA BASTIDAS TOSCANO

Certifico que el presente trabajo de integración curricular fue desarrollado por Ibeth Verónica Bastidas Toscano, bajo mi supervisión.

CARLOS ROBERTO EGAS ACOSTA

DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

IBETH VERÓNICA BASTIDAS TOSCANO

CARLOS ROBERTO EGAS ACOSTA

DEDICATORIA

A mi padre y madre que me dieron la oportunidad de prepararme y estudiar.

AGRADECIMIENTO

A mi madre, hermanos y amigos que estuvieron junto a mí y me apoyaron incondicionalmente a lo largo de toda mi vida universitaria.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

La gestión de redes hace referencia a la instalación, aprovisionamiento, configuración, monitoreo y garantía de la infraestructura de red, esta gestión es realizada por un administrador de red mediante varios protocolos y herramientas. Las redes del Internet de Todo (EoT) o Internet de Todas las Cosas (IoE) y el Internet de las Cosas (IoT), están compuestas por un número de nodos de red heterogéneos imposibles de gestionar de forma manual, por lo que la automatización de la gestión de red resulta la opción más viable para la gestión de redes IoT e IoE.

El presente documento describe los conceptos básicos de la gestión de red tradicional, conceptos de redes IoT e IoE, automatización de la gestión de redes IoT, herramientas y protocolos utilizados en la automatización de gestión de red así como sus características, componentes, ventajas y desventajas; se plantea una comparativa entre las herramientas de automatización de gestión de redes IoT (Ansible, Puppet, SaltStack y Chef) y se exponen entornos de trabajo desarrollados mediante la computación en la nube y el aprendizaje de máquina que proponen un punto de partida para la automatización de la gestión de redes IoE. Finalmente se realiza un análisis de las ventajas y desventajas de las herramientas de automatización de gestión de red, sus retos de implementación y escenarios de implementación con sus respectivos requerimientos.

PALABRAS CLAVE: automatización, Internet de Todas las Cosas (*Internet of Everything*, IoE), Internet de las Cosas (*Internet of Things*, IoT), gestión de red, automatización de la gestión de red, Puppet, Ansible, Chef, SaltStack.

ABSTRACT

Network management refers to the installation, provisioning, configuration, monitoring and assurance of the network infrastructure. Network Management is done by a system administrator through protocols and different tools. Internet of Things and Internet of Everything (IoT and IoE) networks are made up of a huge number of heterogeneous nodes that are impossible to manage manually. That is why network management automation is the most viable option for IoT and IoE network management.

This document describes the basic concepts of traditional network management, IoT and IoE network concepts, IoT network management automation, tools and protocols used in network management automation as well as their characteristics, components, advantages and disadvantages; a comparison between the IoT network management automation tools (Ansible, Puppet, SaltStack and Chef) is proposed and work environments developed through cloud computing and machine learning are exposed, those environments propose a starting point for automation of IoE network management. Finally, an analysis of the advantages and disadvantages of network management automation tools, their implementation challenges and implementation scenarios with their necessary requirements is carried out.

KEYWORDS: automation, Internet of Everything (IoE), Internet of Things (IoT), network management, network management automation, Puppet, Ansible, Chef, SaltStack.

.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

La tecnología se encuentra en continuo desarrollo permitiendo que cada vez más cosas se conecten al Internet, el Internet de todas las Cosas (*Internet of Everything*, IoE) o Internet de todo (EoT) ya no solo se refiere a la conexión y la comunicación entre dispositivos en Internet, como lo es el Internet de las cosas (*Internet of Things*, IoT), sino que además de cosas abarca personas, procesos y datos consiguiendo que las conexiones de red sean más valiosas.

Con el aumento de la información que se obtiene de estas conexiones la gestión de redes incrementa su complejidad representando un flujo de trabajo más complicado, y, a medida que más dispositivos se integran a la red, los errores humanos en la configuración, administración y monitoreo de red pueden llegar a alcanzar costos exorbitantes para las empresas y reducir su productividad.

Los sistemas de gestión de red tradicionales se idearon principalmente para funcionar en hardware, pero debido al continuo incremento de dispositivos heterogéneos conectados a Internet gracias al IoT y en general el IoE, la automatización de redes se vuelve necesaria para la optimización de la red, reducción de costos, disminuir errores humanos y permitir la escalabilidad en la gestión de redes complejas.

La administración tradicional de red no logrará alcanzar las expectativas de servicio esperado si el ritmo de crecimiento y desarrollo tecnológico se mantiene. Al automatizar una red el monitoreo, configuración y aprovisionamiento de red ya no requerirá de la presencia física de un administrador sino solamente del software con las configuraciones respectivas y el acceso remoto a las redes.

En el presente documento se describen inicialmente los conceptos principales relativos al IoE, diferencias respecto al IoT y su relación con el *Cloud Computing*, se habla de los protocolos utilizados en la gestión de red y las tecnologías existentes para la gestión remota de redes tomando en cuenta el aprendizaje de máquina las redes definas por software (*Software Defined Network*, SDN) y la función de virtualización de red (*Network Function Virtualization*, NFV) y se realiza una breve descripción del estado actual de la automatización de gestión de redes.

En la metodología se presenta un análisis comparativo de las herramientas utilizadas en la automatización de la gestión de redes IoT e IoE, se mencionan los leguajes populares utilizados para la elaboración de scripts y códigos de monitoreo de red y se realiza una comparativa entre los mismos.

Finalmente se presenta un análisis de las ventajas y desventajas de automatizar la gestión de redes extensas describiendo varios escenarios de implementación de automatización y sus respectivos requisitos.

1.1 Objetivo general

Estudiar la automatización de redes utilizadas para el Internet de Todo (IoE) mediante el análisis conceptual del IoE y la descripción de las técnicas y herramientas de automatización de la gestión red para optimizar los procesos de administración y monitoreo de red permitiendo que se puedan adaptar a la cambiante demanda de las redes IoE.

1.2 Objetivos específicos

1. Describir los conceptos de redes IoE y las diferencias respecto a las redes IoT.
2. Estudiar conceptos de automatización de redes y las tecnologías para la Gestión Remota de Redes IoE.
3. Investigar sobre el estado actual de la automatización de la gestión de redes IoE y las herramientas para la automatización de su administración.
4. Describir los escenarios de implementación y sus requerimientos para la automatización de la gestión de redes IoE.
5. Enumerar los beneficios de la automatización de gestión de redes IoE, así como también los retos en su implementación.
6. Realizar un análisis comparativo de las herramientas para la automatización de la administración de redes IoE.

1.3 Alcance

El presente proyecto pretende estudiar la automatización de la gestión de redes utilizadas para IoE por medio de un análisis conceptual de las redes IoE y la automatización de gestión de redes.

Se describirán los procesos, técnicas, herramientas, protocolos, requisitos y retos de la automatización de gestión de redes IoE y sus escenarios de implementación para lo que se requiere un conocimiento del estado actual de la automatización de las redes, como por ejemplo los softwares y protocolos ya existentes que monitorizan y gestionan redes sin la necesidad de un administrador de red, los programas generalmente son creados por marcas en específico para sus propios productos. Se analizará también los conceptos de IoE describiendo las principales diferenciaciones respecto a IoT.

A manera de implementación se realizará una revisión de las herramientas y protocolos que permitan la automatización de la gestión de redes mediante la elaboración de scripts haciendo uso de lenguajes tales como Python y Ruby, así como también se presentarán escenarios en los que la implementación de automatización de la gestión de redes IoE represente una mejora en la eficiencia y productividad de las actividades.

Finalmente se realizará un análisis comparativo de los beneficios en la implementación de la automatización de la gestión de redes IoE obteniendo como resultado un estudio teórico basado en un análisis conceptual.

1.4 Marco teórico

Automatización de la red

Conceptos de automatización de red

Automatizar implica reducir al máximo las operaciones manuales permitiendo mejorar la eficiencia de los procesos.

La automatización de red refiere a la habilidad de una red para gestionarse sola, haciendo uso de programas o *scripts* previamente elaborados por Ingenieros para la administración, monitoreo y demás tareas relacionadas con la gestión de redes. Esto puede lograrse aplicando las tecnologías de separación del plano de control y de datos de las redes, así como el aprendizaje de máquina a través de *scripts* realizados utilizando lenguajes como Python.

Herramientas de automatización de la gestión de red

Las herramientas líderes de automatización de la gestión de configuración de red que además hacen uso de la infraestructura como servicio son Puppet, Ansible, Chef y SaltStack todas estas herramientas comparten un objetivo: gestionar eficientemente, y, con la casi nula necesidad de intervención humana, redes a gran escala. Estas herramientas están diseñadas para el despliegue, configuración y gestión de redes. En la sección 2.1 del presente documento se describe a profundidad cada una de estas herramientas, su funcionamiento y características; la sección 2.2 y 2.3 describen respectivamente entornos de automatización de red basados en la nube y entornos basados en el aprendizaje de máquina, estos entornos no están completamente desarrollados para la automatización de redes extensas, pero contienen la base para el desarrollo de la automatización.

Cómo se automatiza la gestión de red

La gestión tradicional de red requiere de un agente y una estación de gestión de red, los agentes recolectan información de los dispositivos de red y la estación de gestión analiza esta información y emite alertas a un administrador de red para solventar fallas.

El proceso de automatización de gestión de red consiste en aplicar aprendizaje de máquina en conjunto con alguna de las herramientas de automatización de gestión de red mencionadas previamente (Puppet, SaltStack, Chef, Ansible), utilizando *scripts* escritos en Ruby o Python, en donde se especifiquen todas las acciones a realizarse. Al automatizar el proceso de gestión de red, los agentes obtienen de la estación de gestión los archivos

de configuración para cada nodo de la red, permitiendo así una configuración y despliegue automático.

Gestión de redes

Conceptos de gestión de redes

Gestionar una red consiste en ejecutar un conjunto de operaciones de administración y monitoreo de red tales como la administración de seguridad, enrutamiento y fallas, con el objetivo de mantener una red eficiente y de buen rendimiento.

Según lo especificado en [1] un sistema de gestión de red cumple con lo siguiente:

- **Gestión de configuración de red:** permite configurar una red de acuerdo con requerimientos específicos.
- **Gestión de topología:** refiere a mantener la conectividad de la red al añadir nuevos dispositivos siempre asegurando un buen rendimiento de red.
- **Gestión de seguridad:** ayuda a prevenir accesos no autorizados mediante cifrado y métodos de autenticación.
- **Gestión de calidad de servicio:** mantiene un rendimiento de red adecuado mediante la clasificación del tráfico y su priorización, evitando pérdida de paquetes y latencia.
- **Gestión de fallas:** permite detectar fallas y resolverlas sin afectar el desempeño de la red.
- **Mantenimiento de red:** operaciones que permiten mantener la red estable y activa, tales como el mantenimiento de software, actualizaciones, etc.

Protocolos de la gestión de red tradicional

En [1] se habla también de los protocolos comúnmente utilizados para la gestión tradicional de red, entre los más destacados se tiene:

- **SNMP (*System Network Management Protocol*):** SNMP involucra gestores, agentes y nodos de red gestionados. Es un protocolo utilizado para el monitoreo remoto de dispositivos IP permitiendo además del monitoreo la configuración de sus parámetros, fue desarrollado por el Grupo de Trabajo de Ingeniería de Internet (*Internet Engineering Task Force*, IETF) y se basa en una Base de Información

Gestionada (*Management Information Base*, MIB) que se encarga de seleccionar la base de datos utilizada para la gestión de dispositivos de red, y, una Estructura de Información Gestionada (*Structure of Management Information*, SMI) que define la estructura y tipos de objetos almacenados en una MIB.

- **CMIP (*Common Management Information Protocol*):** este protocolo permite la gestión de fallas, seguridad y monitoreo de rendimiento, extendiendo las capacidades de SNMP. CMIP es responsable de la comunicación entre el gestor y el dispositivo gestionado y puede usarse en Sistemas de Interconexión de Sistemas Abiertos (*Open Systems Interconection*, OSI)

- **NETCONF (*Network Configuration Protocol*):** es una mejora de SNMP pues implementa nuevas características, entre ellas está la configuración de múltiples almacenes de datos, distinción de datos de configuración y datos de estado y codificación de los datos haciendo uso del Lenguaje de Marcado Extensible (*Extensible Markup Language*, XML). Para extender las capacidades del protocolo NETCONF se diseñó el protocolo RESTCONF definido por la IETF en 2017, este protocolo permite gestionar la red y ejecutar las operaciones de creación, lectura, actualización y eliminación de datos (*Create, Retrieve, Update, Delete*, CRUD) mediante aplicaciones web.

- **OMA-DM (*Open Mobile Alliance - Device Management*):** es un protocolo desarrollado por el grupo de trabajo de la sincronización de datos y gestión de dispositivos de la *Open Mobile Alliance* (OMA), permite entre otras cosas la actualización de software, configuración de dispositivo y manejo de fallas.

Los protocolos contemplados para redes complejas permiten la escalabilidad, seguridad, tolerancia a fallos, auto configuración y calidad de servicio (Quality of Service, QoS) para el monitoreo y gestión de sus dispositivos. Existen protocolos de gestión para redes IoT de bajo consumo de energía diseñados para permitir un buen desempeño de red haciendo uso mínimo de recursos para las operaciones de gestión de red, algunos de estos protocolos según [1] se detallan a continuación:

- **LWM2M (*Light Weight Machine to Machine*):** es un protocolo diseñado por la *Open Mobile Alliance* (OMA) basado en los estándares de protocolo y de seguridad de la IETF. LWM2M se utiliza para la gestión de dispositivos IoT de bajo consumo de energía, funciona en modelos cliente-servidor y entre sus características se destaca la conectividad, monitoreo, seguimiento de recursos y la actualización de *firmware*.

- **CoMI (*CoAP Management Interface*):** CoMI es una interfaz de gestión de dispositivos IoT de bajo consumo de energía. El protocolo CoAP habilita la ejecución de operaciones de gestión para recursos de dispositivos IoT especificados en YANG.
- **6LowPAN-SNMP:** es una adaptación del protocolo SNMP para redes IPv6 inalámbricas de área personal de bajo consumo de energía (IPv6 *Low-Power Wireless Personal Area Network*, 6LowPAN) para dispositivos con recursos limitados, la compatibilidad entre estos protocolos se asegura mediante un proxy de reenvío que convierte mensajes SNMP en mensajes 6LowPAN-SNMP.
- **NETCONF light:** protocolo desarrollado por la IETF para dispositivos con recursos limitados, provee herramientas de instalación y manipulación de la configuración de los dispositivos

Los protocolos mencionados no son autosuficientes y requieren de la gestión de un administrador de red.

FCAPS

La administración de red abarca la gestión de fallas, configuración, rendimiento, contabilidad y seguridad, por sus siglas en inglés FCAPS (*Fault, Configuration, Accounting, Performance, and Security Management*). A continuación, se describe brevemente cada área de las FCAPS:

- **Gestión de fallas**: se encarga de recolectar y analizar alarmas y fallas en los servicios. La gestión de fallas filtra los mensajes de error y los coordina de modo que los eventos actuales reflejen la condición real del sistema.
- **Gestión de configuración**: permite la estandarización de la activación y desactivación de servicios de forma controlada. Provee inventarios, localizaciones, instrucciones y mantenimiento de los agentes de servicio sus componentes y configuraciones, esta información se recolecta continuamente para ser monitoreada.
- **Gestión de contabilidad:** se asegura que no se excedan las cuotas establecidas y no se comentan fraudes, monitoreando parámetros que definen el límite de utilización, monitoreo y costo de uso. Recolecta la información de contabilidad y uso para enviar reportes a otros servicios.

- **Gestión de rendimiento:** se encarga de medir y analizar el rendimiento al recolectar datos de rendimiento, evaluarlos y levantar alertas cuando el rendimiento varíe y no se encuentre dentro de los límites establecidos.
- **Gestión de seguridad**: controla el acceso y control de seguridad y utilización de servicios, manteniendo la privacidad, confidencialidad e integridad de la información.

Internet de todas las cosas

Conceptos de redes IoT y redes IoE

El internet de las cosas (IoT) se refiere a la capacidad que tienen los objetos de conectarse a internet e interactuar entre ellos, entre varias de sus aplicaciones se tienen los teléfonos inteligentes, domótica, agricultura inteligente, dispositivos electrónicos que se usan en el cuerpo humano (*wearable*), sistemas inteligentes de transporte, etc. El IoT describe un mundo que interconecta dispositivos y sensores que permiten obtener la información necesaria para la gestión y monitoreo de un amplio catálogo de sistemas dedicados a actividades cotidianas con el fin de automatizar y facilitar el trabajo humano, sin embargo, el IoT es únicamente una rama del Internet de Todas las Cosas.

Según Cisco en [2], el IoE además de la interconexión de dispositivos inteligentes, se refiere a la conexión y comunicación de personas, procesos y datos estableciendo un enlace extremo a extremo, esto crea un gran valor y oportunidades para organizaciones, comunidades, países e individuos en general. Las interacciones en el IoE pueden ser máquina a máquina (*machine-to-machine*, M2M), máquina a personas (*machine-to-people*, M2P) y personas asistidas por tecnología a personas (*Technology-assisted people-to-people*, P2P). De acuerdo con D. Vaya and T. Hadpawat en [3] cada componente que se interconecta en el IoE puede definirse como se explica brevemente a continuación:

- **Personas:** las personas tienen varias formas de conectarse a la red, por ejemplo, los relojes inteligentes, sensores que pueden utilizarse en la ropa e incluso las redes sociales permiten que una persona actúe a manera de un nodo que constantemente transmite datos.
- **Datos:** la información recolectada por los dispositivos de red se envía a un repositorio central para su pronto procesamiento y análisis, permitiendo que su almacenamiento sea de corta duración.
- **Procesos:** es responsable de la interacción de datos, personas y cosas.

- **Cosas:** son los dispositivos que interactúan entre sí y recolectan la información a ser procesada y la comparten, ayudando en la toma de decisiones, por ejemplo, dispositivos inteligentes de medición de electricidad que comparten energía consumida.

Diferencias entre IoT e IoE

El internet de Todas las Cosas (IoE) comprende varias tecnologías de transición entre las que se encuentra Internet de las personas (*Internet of People*, IoP), Internet de lo Digital, Internet de las Cosas Industrial (IIoT) y el IoT.

La principal diferencia entre las redes IoT e IoE es su alcance, el internet de las cosas conecta únicamente dispositivos a la red, mientras que el internet de todo considera personas, datos y procesos por lo que se toman en cuenta más interacciones y se permite el análisis de datos para ofrecer servicios más personalizados y automatizados a los usuarios. [3]

El término IoE ha estado siendo usado principalmente por Cisco, sin embargo, varios autores utilizan el término IoT para referirse también al IoE, este es el caso de Qualcomm en [4]. Además, al ser el IoT parte del IoE, las tecnologías, protocolos, arquitecturas y plataformas consideradas para la automatización de redes IoT serán muy similares para las redes IoE, pero a escalas mayores, ya que todos estos avances se han considerado para redes heterogéneas y a gran escala, por esta razón se analizará la información obtenida para automatización de la gestión de redes IoT.

Redes IoE y el Cloud Computing

Los datos recolectados por los dispositivos inteligentes en redes IoE proporcionan información valiosa que requiere ser procesada, debido a lo extenso de estas redes la cantidad de datos que se obtiene es enorme por lo que su análisis y procesamiento requiere de plataformas capaces de administrar grandes cantidades de datos.

F. Hussain en [5] define a la computación en la nube (*Cloud Computing*) como una plataforma capaz de manejar y administrar grandes cantidades de datos, permite ofrecer escalabilidad, reducir costos, y mejorar el desempeño y mantenimiento de redes, además permite el acceso a la información almacenada en cualquier lugar y desde cualquier dispositivo inteligente.

Automatización de la gestión de Redes IoT e IoE

La gestión de redes es un proceso que se realiza mayoritariamente en forma manual, según Cisco [6] actualmente hasta el 95 por ciento de los cambios en las redes se realizan de forma manual, esto no solo implica un incremento en los tiempos de configuración e integración de nuevos equipos a la red, sino también una mayor probabilidad de cometer errores de configuración que afecten el desempeño de la red y, por lo tanto, la disponibilidad de los servicios que funcionan sobre la misma.

El desarrollo continuo y el rápido crecimiento del mundo tecnológico requiere la búsqueda de soluciones para satisfacer una demanda tecnológica cambiante y agilizar procesos tales como la planificación y monitoreo de red; integración, mantenimiento y actualización de equipos, entre otros.

La administración tradicional de red no logrará alcanzar las expectativas de servicio esperado si el ritmo de crecimiento y desarrollo tecnológico se mantiene. Reemplazar los procesos manuales y automatizarlos puede lograrse haciendo uso de programas, *scripts*, comandos, inteligencia artificial mediante el aprendizaje automático, entre otros. Mejorando tanto la eficiencia de la red como la de los trabajadores.

La automatización supone la optimización de recursos económicos pues reduce los costos de operación en la gestión manual de las redes, facilita el trabajo a gran escala y asegura una red con desempeño confiable ya que los errores humanos no tienen lugar al ser automatizados los procesos de gestión de red.

A continuación, se mencionan algunas tecnologías para la gestión de red tradicional, así como también tecnologías que son la base de la automatización de redes extensas.

Tecnologías para la gestión remota de redes

Los sistemas remotos de gestión de red ayudan a optimizar la administración de los recursos de red, en [7] I. Peksens y V. Zagursky detallan tecnologías de gestión remota de redes como CNSRM, DMI y WBM que se detallan a continuación:

- **Gestión remota de sistemas y redes informáticas (*Computer Network and System Remote Management*, CNSRM)**

  Se compone de un dispositivo gestionado y un agente que se comunican a través del protocolo SNMP basado en una base de información gestionada (MIB) y una estructura de información gestionada (SMI) que define la estructura y tipos de objetos almacenados en una MIB. La gestión de los dispositivos se realiza en

tiempos establecidos y mediante comandos previamente configurados por el administrador de red.

- **Interfaz para administración de escritorio (*Desktop Management Interface*, DMI)**

  Esta tecnología garantiza un acceso transparente al sistema remoto y permitiendo gestionar el hardware, software y el sistema operativo (*Operating System*, OS), es capaz de procesar eventos sincrónicos y asíncronos.

  DMI se compone de un gestor y un agente, la comunicación entre ambos se realiza mediante el protocolo SNMP o RPC (*Remote Procedure Call*). Cada cliente DMI contiene archivos con parámetros de lectura y escritura que juntos conforman la base de datos de archivos de información de gestión (*Management Information Files*, MIF), esta base de datos consta de una interfaz de gestión (*Management Interface*, MI), un bloque de sincronización y una interfaz de componentes (*Component Interface*, CI). El cliente permite al gestor acceder remotamente a la base de datos MIF para los propósitos administrativos que sean requeridos.

- **Gestión basada en la Web (*Web-Based Managemet*, WBM)**

  Esta tecnología de gestión está basada en el lenguaje de marcado de hipertexto (*Hyper Text Markup Language*, HTML) y en los principios de la web por lo que se relaciona muy de cerca con el crecimiento de Internet. Posibilita la unión de todas las tecnologías existentes y así garantizar una interfaz de usuario que permita ejecutar funciones para gestión.

  Es compatible con diversas arquitecturas debido a que existen variantes que se acoplan a cada caso, entre las que se tiene a la gestión empresarial basada en Web (*Web-Based Enterprise Management*, WBEM) que surge de la cooperación de Cisco, Intel y BMC; Interfaz de administración de Windows (*Windows Management Interface*, WMI) que se adapta a los productos de Microsoft; Interfaz Java de programación de aplicaciones de gestión (*Java Management Application Programming Interface*, JMAPI) que contiene clases especialmente diseñadas para tareas de gestión.

  WBM se compone de un gestor y un agente que se comunican utilizando el protocolo de transferencia de hipertexto (*Hyper Text Transfer Protocol*, HTTP) o el protocolo de administración de hipermedios (*Hyper Media Management Protocol*,

HMMP), sin embargo, HTTP no soporta eventos de notificación asíncrona lo que se soluciona haciendo uso de HMMP y una arquitectura de gestión distribuida.

Tecnologías para la gestión remota de redes IoT e IoE

Redes definidas por software (SDN) y virtualización de la función de red (NFV) en la automatización de la gestión de red

SDN y NFV son tecnologías complementarias que posibilitan que elementos de hardware puedan se implementados con software, lo que se denomina *softwarización* de red.

- **SDN:** Las redes definidas por software separan el plano de control del plano de datos de una red, permitiendo que el plano de control pueda ser programable, de esta forma funciones realizadas por hardware como por ejemplo el ruteo, pueden ser implementadas en software [8]. Esto permite que la automatización de gestión de redes dinámicas y extensas sea más sencilla.

  En [9] se propone una arquitectura virtualizada de automatización de gestión de red basada en SDN, SDNVA por sus siglas en inglés (*SDN-based network virtualization architecture with autonomic management*) esta arquitectura asegura una separación entre los recursos virtuales de red usando un módulo de virtualización y proporcionando una gestión automatizada jerárquica entre los recursos físicos y virtuales.

- **NFV:** La virtualización de función de red provee flexibilidad y programabilidad en la orquestación dinámica de red. NFV separa las funciones del hardware y las ejecuta en software utilizando máquinas virtuales o contenedores. [8]

Aprendizaje de máquina (*Machine Learning*, ML) y Python

El aprendizaje de máquina consiste en extraer conocimiento, utilizarlo y mejorarlo a partir de un conjunto de datos de entrenamiento y prueba. Esto lo hace a través de un conjunto de técnicas entre las que se puede destacar K-N vecinos más cercanos, árboles de decisiones, redes Bayesianas, entre otras. El ML se clasifica en:

- **Aprendizaje supervisado:** hace uso de conjuntos de datos para crear modelos que relacionen entradas con sus respectivas salidas.
- **Aprendizaje no supervisado:** hace uso de conjuntos de datos para crear modelos que busquen estructuras o patrones en los datos.

- **Aprendizaje de refuerzo:** proceso iterativo que utiliza la retroalimentación del ambiente para corregir la secuencia de acciones.

El ML puede ser aplicado para automatizar la gestión de red, en [10] se menciona que varias de las técnicas utilizadas para ML son útiles para FCAPS.

El ML puede ser implementado en cualquier lenguaje de programación, por su simplicidad la tendencia es el uso de Python.

Según R. Gonzales [11] Python es un lenguaje interpretado, es decir, se ejecuta haciendo uso de un programa intermedio denominado intérprete permitiendo que su ejecución sea más rápida; es de tipado dinámico por lo que no se necesita declarar el tipo de dato que almacenará una determinada variable; está fuertemente tipado, lo que significa que no se puede tratar una variable como un tipo distinto de dato; es multiplataforma pues está disponible en plataformas como UNIX, DOS, Mac OS, Windows, Linux, entre otros; y, es orientado a objetos paradigma en el que los conceptos del mundo real son transformados en clases y objetos dentro del programa que se está elaborando.

Estado actual de la automatización de la gestión de redes IoE

La automatización de red surge de un manifiesto en 2001 del *International Business Machines Corporation* (IBM) [12] en el que se habla sobre la auto gestión de red y las propiedades *"self -*"* que tiene un sistema autónomo, donde "*" se refiere a optimización, protección, configuración y reparación autónoma, permitiendo que la red se adapte a cualquier cambio de su entorno sin intervención humana.

Según lo detallado en [12] esto se logra con un lazo autónomo de control (*Autonomic Control Loop*, ACL) que podría implementarse para recolectar grandes cantidades de datos, monitorearlos, analizarlos, planificar y ejecutar respuestas dinámicas a las demandas evolutivas de los servicios. Esto es precisamente lo que hace el modelo de referencia MAPE-K (*monitor, analyze, plan, execute, knowledge*) que surgió del manifiesto de IBM, MAPE-K se basa en una ACL entre los elementos de red donde se ejecuta el monitoreo de datos para luego analizarlos, planificar y ejecutar las decisiones tomadas.

Actualmente existen numerosas áreas de investigación dedicadas a la automatización de red [8], sin embargo, aquellas que más se adecúan para la implementación de automatización de gestión de redes extensas, como los son las redes IoT e IoE, son aquellas que utilizan *softwarización* de la red basándose aprendizaje de máquina (*Machine Learning*, ML), redes definidas por software (SDN) y virtualización de la función de red (NFV) pues en conjunto hacen que la red se vuelva flexible e innovadora con la

virtualización y programabilidad de red, posibilitando que las configuraciones, monitoreo, control y segmentación de red sean dinámicos. El ML también ya es aplicado en el manejo de recursos de red, predicción de tráfico, seguridad y calidad de servicio.

Las organizaciones de estandarización mantienen una investigación continua y han implementado soluciones para la automatización de redes extensas, algunos de estos avances se listan a continuación:

- El Instituto Europeo de Normas de Telecomunicaciones (*European Telecommunications Standards Institute*, ETSI) tiene un rol importante en cuanto a las arquitecturas de redes automatizadas y ha establecido un grupo de especificación industrial (*Industry Specification Group*, ISG) para la ingeniería de red de auto gestión del Internet del futuro (*Autonomic network engineering for the self-managing Future Internet*, AFI).

  La arquitectura de referencia estándar desarrollada por ETSI se denomina GANA (*Generic Autonomic Network Architecture*) y una de sus versiones, el ETSI AFI GANA descrito en [13] es un modelo de referencia para la gestión automatizada y control, su implementación considera redes cognitivas, la auto gestión y la arquitectura GANA.

  GANA comprende elementos de toma de decisiones (*Decision-making elements*, DEs) en cuatro niveles y, elementos gestionados (*Managed Elements*, MEs) que forman un lazo de control jerárquico (*Hierarchical Control Loop*, HCL) que tiene políticas de elementos superiores denominadas comportamiento autónomo (*autonomic behaviors*, ABs). Los DEs se comunican entre sí de acuerdo con HCL recolectando datos, evaluándolos y generando las instrucciones luego de un proceso de decisión [8].

- La Fuerza de Tareas de Investigaciones de Internet (*Internet Research Task Force*, IRTF) está trabajando en el desarrollo de estándares para permitir una mejor y eficiente gestión de redes a través de la automatización. IRTF basado en el RFC 7575 y el RFC 7576 creó el grupo de investigación ANIMA [14] (*Autonomic Networking Integrated Model and Approach*) encargado de definir protocolos para la infraestructura de red autónoma.

  El grupo de trabajo de ingeniería de Internet (*Internet Engineering Task Force*, IETF) trabaja en la estandarización de protocolos que habiliten la auto gestión, auto configuración, auto protección y auto corrección de redes puesto que los protocolos

de gestión de redes tradicionales como SNMP no están contemplados para redes complejas.

Ejemplos de implementación de automatización de gestión de redes IoT se pueden encontrar en [15] donde se habla de una arquitectura de control y gestión de redes de transporte óptico adoptando las funciones del plano de control de ETSI NFV.

En [16] se presenta una abstracción y control de redes de ingeniería de tráfico (*Traffic Engineering*, TE) mediante la configuración de redes virtuales con las operaciones necesarias para controlar y gestionar redes TE a larga escala, multi dominio, multi capa y de diferentes proveedores.

La automatización de gestión de redes sigue desarrollando áreas de investigación de acuerdo con la demanda y crecimiento de las redes. Los artículos y autores que ofrecen soluciones para automatización de gestión de redes IoT se considerarán y utilizarán para el desarrollo del presente trabajo ya que IoT es parte del IoE y la información es relevante para redes IoE.

2 METODOLOGÍA

En el presente componente se describe un análisis comparativo entre las herramientas, lenguajes y protocolos utilizados en la automatización de redes IoE, en este sentido, se ha planteado un método inductivo con el fin de obtener conclusiones a partir del análisis de la información investigada; y, debido a la naturaleza de datos se mantuvo un enfoque cualitativo para un trabajo de tipo descriptivo mediante una técnica de recolección de información de análisis documental para lo cual se realizó una investigación profunda de la bibliografía y artículos científicos disponibles en Internet.

La investigación descriptiva según F. Morales [17] consiste en la caracterización de las situaciones concretas a ser analizadas, indicando sus rasgos principales, peculiaridades y aspectos diferenciadores. No se trata únicamente de recolectar datos, sino que busca la identificación de relaciones existentes entre dos o más variables, en este tipo de investigación se recopila la información acorde al tema bajo estudio para luego analizarla y extraer conclusiones que favorezcan el conocimiento.

El análisis de la información obtenida siguió el procedimiento detallado a continuación basado en [18] :

- **Revisión de documentación y reajuste de datos:** en esta etapa, posterior a la recolección de la información relevante para el estudio de este componente, se realiza una simplificación o resumen de toda la bibliografía y artículos científicos consultados, mediante procedimientos analíticos generales y listando las principales características de cada ítem a ser comparado.
- **Transformación de datos y análisis de contenido:** aquí se han ilustrado las relaciones entre los conceptos analizados mediante matrices descriptivas que permitan obtener una visión global de la información con el objetivo de facilitar el análisis y realizar comparaciones. Las matrices obtenidas se enfocan a temas en específico contrastando siempre los mismos ítems de estudio, desde diferentes puntos de comparación.
- **Obtención de conclusiones:** basado en el análisis tabular y comparativo de la información recolectada se llega a una comprensión amplia del tema bajo estudio y se puede de esta forma destacar las características, falencias y virtudes de los conceptos bajo comparación

2.1 Herramientas para la automatización de la administración de redes IoE

Las herramientas de gestión y configuración de red requieren generalmente de agentes, un agente es una pieza de software que debe ser instalado en los dispositivos a ser gestionados en una red, sin embargo, en redes IoT e IoE existen nodos simples que no permiten la carga de software general y mucho menos la ejecución de agentes, en estos casos se puede hacer uso de herramientas que soporten agentes proxy o herramientas que no requieran de agentes.

Según P. Gargano [19] existen 3 tipos de herramientas para la administración y gestión de red:

- **Configuración basada en agentes:** en este caso un agente debe ser instalado obligatoriamente en todos los dispositivos a ser gestionados. Como ejemplo de una configuración basada en agentes se tiene a las herramientas de gestión Puppet y Chef.
- **Configuración sin agente:** no requiere de agentes instalados en los dispositivos a ser gestionados, utilizan interfaces de programación de aplicaciones (*Application Programming Interfaces*, APIs) que soporte el dispositivo a gestionarse o SSH (*Secure Shell*) para la comunicación. Ansible es una herramienta de gestión de configuración sin agente.
- **Configuración de agente proxy:** no requiere de agentes en cada dispositivo a gestionarse, sin embargo, necesita algún tipo de proceso que permita la comunicación entre el servidor maestro y el dispositivo remoto.

  Salt es una herramienta de configuración basada en agente, cada dispositivo gestionado usando Salt debe tener un agente instalado denominado Salt minion, sin embargo, si el dispositivo no admite la instalación de agentes puede utilizarse un proxy minion que no necesita ser instalado en el dispositivo y corre un software intermedio que permite la comunicación entre el maestro Salt y el Salt minion.

Estas herramientas de gestión funcionan ejecutando instrucciones en lenguaje de dominio específico (*domain-specific languaje*, DSL) o en lenguajes de propósito general. DSL está diseñado para expresar soluciones a problemas presentadas en dominios específicos, un

ejemplo de esto es YAML. DSL es limitado respecto a lenguajes de propósito general como lo son Python y Ruby.

En la **Figura 2. 1** se puede apreciar una comparativa entre los 3 tipos de herramientas de automatización de gestión de red descritos previamente.

Como se muestra en la imagen las herramientas de configuración basadas en agente como Puppet y Chef utilizan SSH para comunicar al maestro con los dispositivos gestionados que tienen un agente Puppet y un agente Chef respectivamente.

SaltStack puede actuar como una herramienta de configuración sin agente en la que el maestro se comunica con sus dispositivos gestionados haciendo uso de un proxy minion que actúa como intermediario entre el maestro y el dispositivo gestionado, por lo que se elimina la necesidad de que exista un agente instalado en cada dispositivo, sin embargo, dependiendo de las características del dispositivo a gestionarse un agente puede ser instalado y de esta forma SaltStack funcionará como una herramienta basada en agente.

Por último, se observa como Ansible permite la comunicación entre el controlador y el dispositivo a gestionarse mediante el uso de APIs y SSH sin la necesidad de agentes.

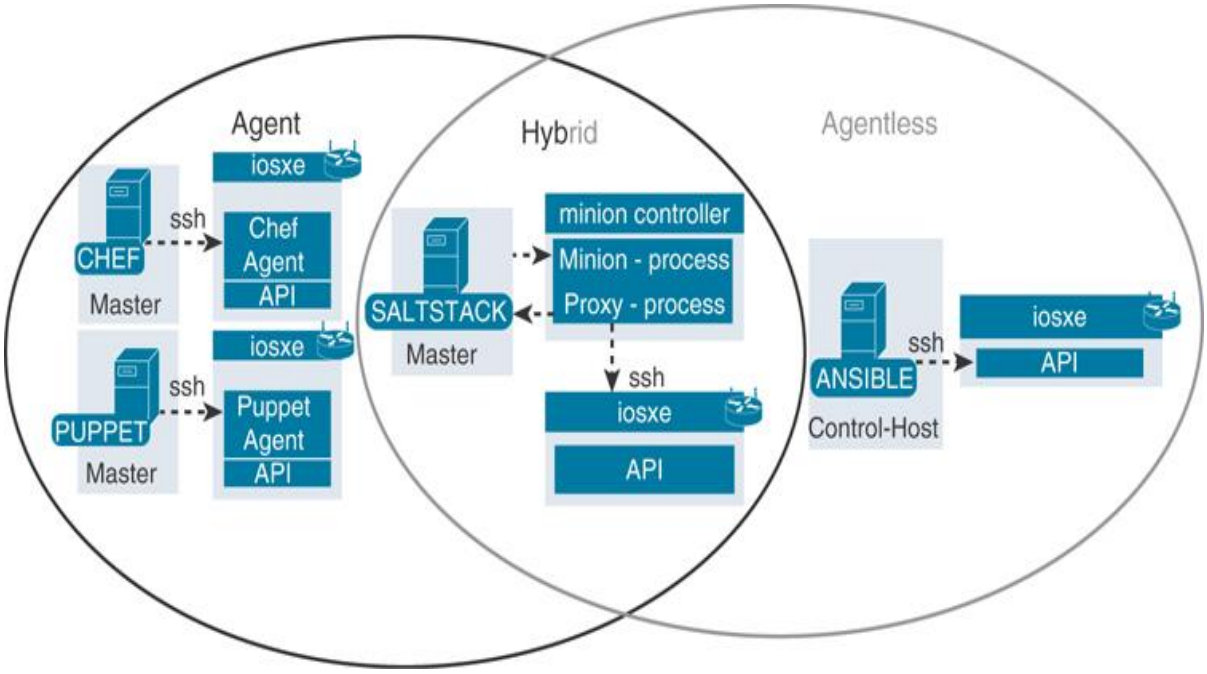

Figura 2. 1 Comparación de los tipos de herramientas para automatización de gestión y configuración de redes. (Fuente: [19])

Ansible

Es una herramienta o plataforma de automatización de código abierto que permite la gestión y configuración centralizada. Fue originalmente escrita por Michael DeHann de AnsibleWorks adquirida por Red Hat en 2015 y por IBM en 2019 [19]. Ansible se incluye como parte de Fedora y está disponible en RHEL, CentOS, Scientific Linux, entre otros, es una herramienta de automatización de gestión que no necesita agentes instalados en sus dispositivos gestionados. Su propósito principal es configurar tres tipos de tareas en routers, switches y servidores, según A. Ratan en [20] estas tareas son:

- **Gestión de configuración:** se utiliza para buscar y enviar configuraciones en varios dispositivos que son llamados inventarios en Ansible, dependiendo del tipo de inventario pueden enviarse configuraciones masivas específicas o completas.
- **Implementación de aplicaciones:** Ansible se encarga del despliegue y actualización de las aplicaciones en los servidores que lo necesiten, incluso puede configurar las aplicaciones para tareas específicas y configurarlas basado en los dispositivos del inventario.
- **Automatización de tareas:** permite que se realicen tareas previamente escritas por un administrador de red, en uno o varios dispositivos. Ansible puede configurarse para ejecutar estas tareas una sola vez o de forma periódica.

Componentes de Ansible

- **Estación de control Ansible:** es la estación de gestión, no es un servidor dedicado o hardware elaborado pues Ansible no lo requiere, puede ser incluso un computador personal.
- **Módulo Ansible:** colección de comandos escritos en un lenguaje de scripting estándar como Python, puede usarse cualquier lenguaje que soporte notación de objetos JavaScript (*JavaScript Object Notation*, JSON).
- ***Playbooks* (libros de jugadas):** se refiere a un conjunto de instrucciones que se crean para que Ansible configure, despliegue y maneje los nodos gestionados en grupos o de forma individual. Están escritos en un lenguaje básico de texto que no es de marcado (*Ain't Markup Language*, YAML). En la Figura 2. 2 se puede apreciar la estructura de un *playbook* y las terminologías utilizadas en éstos se detallan en la Tabla 2. 1

Tabla 2. 1 Terminologías del libro de jugadas de Ansible. (Fuente: *[20]*)

Término	Descripción
Módulo	Código escrito en cualquier lenguaje de scripting para ejecutar alguna acción en un dispositivo gestionado.
Tarea (*task*)	Acción que referencia a un módulo para ejecutarse con argumentos de entrada.
Jugada (*play*)	Conjunto de tareas para un host o grupos de host.
Libro de jugadas (*playbook*)	Archivo escrito en YAML que incluye una o más jugadas.
Rol	Conjunto de libros de jugadas para ejecutar configuraciones estándar de forma repetitiva. Uno o múltiples roles pueden asignarse a un solo host.

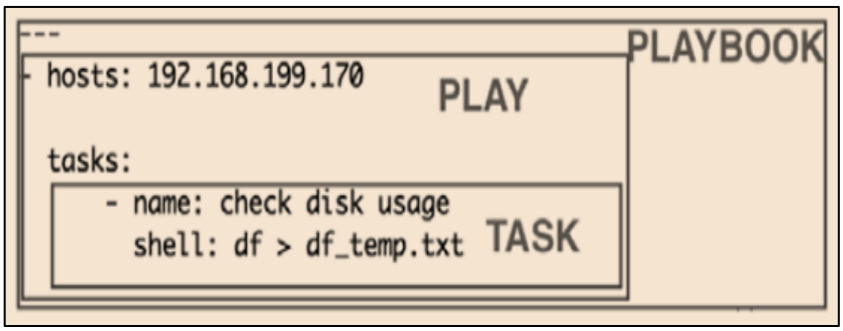

Figura 2. 2 Libro de jugadas de Ansible. (Fuente: *[20]*)

- **Archivos de inventario:** contiene sistemas manejados por Ansible, el administrador usa el inventario para agrupar sistemas gestionados. Pueden ser construidos dinámicamente haciendo uso de una API remota que responda con una respuesta JSON válida.
- **Archivos YAML:** llamados también archivos de configuración y destino para los datos almacenados, es un formato de datos transportados sobre SSH y con soporte de PowerShell para nodos Windows.
- **Ansible Tower:** es una consola de servicio web que sigue el estándar REST para programabilidad.

Ansible utiliza el modelo *push* para enviar los comandos a los nodos a través de SSH, Ansible no requiere la presencia de agentes en los nodos gestionados, generalmente una estación de trabajo centralizada se utiliza para distribuir los mensajes a los dispositivos administrados, como se muestra en la Figura 2. 3 el nodo de gestión Ansible está cargado con los *playbooks* que contienen los archivos de configuración de los nodos a ser gestionados, y, el inventario que muestra una lista de los nodos en la red a ser gestionados. El nodo de gestión se comunica con los nodos gestionados mediante SSH y con el modelo push les envía las configuraciones respectivas.

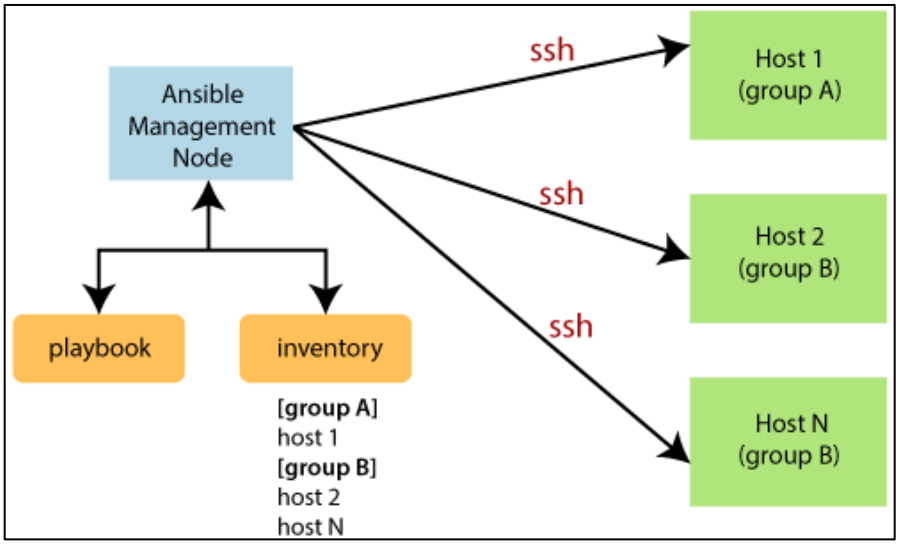

Figura 2. 3 Esquema de trabajo Ansible. (Fuente: *[21]*)

Ventajas

De acuerdo con E. Chou en [22] entre las ventajas de Ansible se pueden listar las siguientes:

- Ansible no requiere de un modelo maestro-esclavo por lo que no se requiere que un software específico o agente sea instalado en el dispositivo gestionado que se comunica con la controladora fuera del intérprete de Python.
- No utiliza agentes remotos en los nodos gestionados, usa SSH o llamadas a APIs para aplicar los cambios requeridos en el host remoto, lo que reduce la necesidad de un intérprete Python en los nodos gestionados.
- Ansible es idempotente, lo que significa que ejecutará un libro de jugadas las veces necesarias, comprobando primero si los cambios descritos existen evitando sobrecargas.
- Ansible cuenta con un amplio número de módulos desarrollados por la comunidad.

Ansible en redes IoE

Algunas de las recomendaciones en [23] que pueden aplicarse para utilizar libros de jugada de Ansible en redes IoE se listan a continuación:

- **Habilitar almacenamiento en caché:** esto permite que los libros de jugada se desplieguen rápidamente pues ya no se debe extraer la misma información ni se tiene que correr la configuración en cada ejecución del libro de jugadas. Los trabajos pueden ser programados y así las configuraciones pueden ser almacenadas para ejecuciones posteriores.
- **Datos de etapa:** generalmente un libro de jugada copia el firmware y luego ejecuta una actualización, el nuevo enfoque es tener un libro de jugadas para copiar el firmware y otro para ejecutar la instalación.
- **Validar datos de la etapa:** se debe validar la suma de comprobación del archivo de no tener este método de verificación se debe validar como mínimo el tamaño de los archivos.
- **Limitar los recursos del sistema:** utilizar usuarios Ansible separados y crear grupos para administrar grandes cantidades de dispositivos.
- **Trabajar con dispositivos integrados:** si los dispositivos a ser administrados no cuentan con un sistema operativo se los trata como dispositivos Ansible de red para lo que se requiere escribir módulos de Ansible personalizados para el dispositivo.

Puppet

Es una herramienta de configuración y gestión de red de código abierto, escrita en C++, Clojure y Ruby. Puppet se utiliza para automatizar el despliegue de configuración en dispositivos y puede ser usada por varias distribuciones de Linux, sistemas UNIX y Windows.

Componentes de Puppet

Puppet tiene una arquitectura cliente/servidor donde el cliente es llamado agente y el servidor es llamado maestro. El maestro controla completamente las configuraciones mientras que el agente periódicamente obtiene las configuraciones del maestro.

Puppet es una herramienta para automatización de gestión de red basada en agente, sin embargo, puede utilizar proxys para comunicarse con dispositivos que no admiten la instalación de agentes, el proxy se encarga de establecer comunicaciones con el maestro,

conectarse al dispositivo gestionado y aplicar la configuración. Algunos conceptos básicos para entender Puppet según J. Rhett en [24] son los siguientes:

- **Maestro Puppet:** servidor en el que el software de Puppet está instalado, es el encargado de gestionar toda la información de configuración para los nodos haciendo uso de manifiestos.
- **Agente Puppet:** software instalado en los nodos a ser gestionados por Puppet para la recolección de información.
    - **Factor:** es una parte de los nodos que contienen el agente Puppet, este ayuda a Puppet a descubrir los nodos.
- **MCollective (*Marionette Collective*):** es una herramienta de orquestación (de Puppet 4) que controla el agente Puppet en los nodos de la red, esta herramienta consulta, inicia, detiene y reinicia el agente Puppet. Utiliza ActiveMQ para proporcionar un software intermedio entre el agente y el servidor Puppet y soporta protocoles de conexión como capa de sockets seguros (Secure Socket Layer, SSL).
- **Recurso:** la entidad más elemental de Puppet, un recurso es una porción del código de Puppet que describe un aspecto particular del sistema.
- **Clase:** es una colección de recursos que pueden reusarse en múltiples ocasiones.
- **Manifiesto:** archivo de configuración que describe cómo configurar los recursos. Es un documento de texto con extensión ". pp" que utiliza codificación UTF-8.
- **Catálogo:** configuración compilada para cada nodo, el catálogo describe el estado esperado del nodo gestionado.
- **Módulo:** colección de manifiestos, plantillas y otros archivos utilizados para configurar una o varias características y que pueden reusarse.

Luego de la instalación de Puppet es posible crear archivos de inventario con la sintaxis YAML y para simplificar la gestión se construyen grupos haciendo uso de un archivo de inventario, similar a Ansible. Estos archivos de inventario contienen la información necesaria para que pueda establecerse una conexión con los dispositivos de red. Luego de agrupados los dispositivos se crean los manifiestos que contienen todos los cambios de configuración a ser extraídos por los dispositivos gestionados. [25]

El software de Puppet utiliza un modelo de extracción (*pull*) en el que los agentes Puppet verifican en intervalos de tiempo regulares si existen nuevas configuraciones en el maestro Puppet. Inicialmente los nodos a ser gestionados recolectan información utilizando Factor para luego enviarla desde el agente al maestro, el maestro compila los manifiestos en catálogos para especificar las configuraciones que deben tener los nodos gestionados.

Finalmente, los agentes extraen los catálogos del maestro, configuran el dispositivo y envían un reporte al maestro. La Figura 2. 4 detalla el proceso previamente descrito.

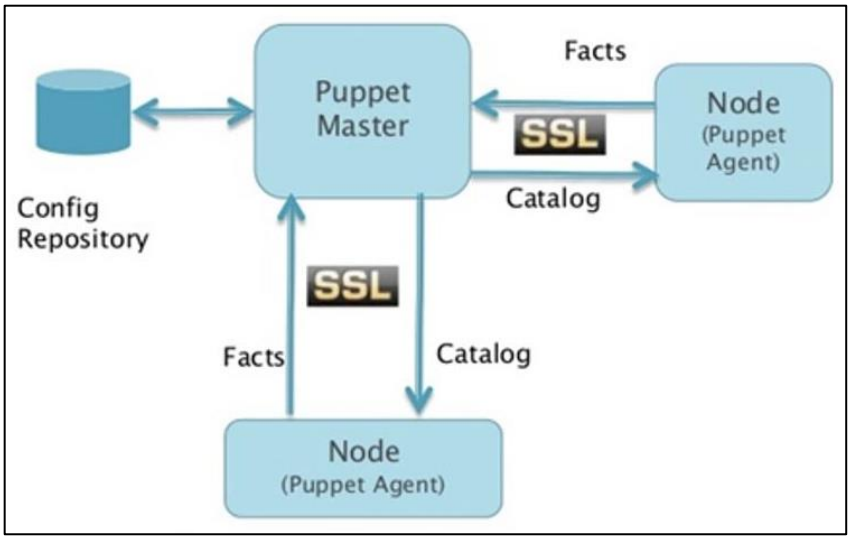

Figura 2. 4 Esquema de trabajo de Puppet. (Fuente: *[26]*)

Ventajas

Según [24] se pueden listar las siguientes ventajas de Puppet:

- Puppet personaliza la configuración y las políticas de cada nodo haciendo uso de los datos locales del nodo en cuestión, estos datos específicos son el sistema operativo, memoria, configuración de red, nombre de host, entre otros.
- Los agentes de Puppet pueden invocar etiquetas específicas del maestro, permitiendo así el filtrado de configuración y la ejecución de operaciones que correspondan a las etiquetas invocadas.
- Los agentes Puppet envían retroalimentación al maestro, reportando éxitos, fracasos y resultados convergentes para cada recurso.
- Puppet permite un enfoque descentralizado en el que cada nodo converge por separado sin depender de la culminación de ejecución en otros nodos.

- Algunas de las corporaciones que utilizan Puppet para gestionar sus redes son PayPal, Google, Reddit, Dell, Oracle y la Universidad Stanford.

Puppet en redes IoE

Para implementar Puppet en redes IoT y redes IoE se requiere un conocimiento profundo de Ruby, por lo que llegaría a ser complejo dependiendo de las capacidades del personal que implementará los scripts.

Es posible utilizar esta herramienta en redes extensas pues *Puppet device*, una versión de Puppet, permite gestionar redes tradiciones y redes IoT ya que el agente actúa como un proxy en dispositivos que no soportan la instalación de agentes como lo son los nodos de recursos limitados que conforman redes IoT. La implementación requiere configurar *Puppet device* en los proxys de los agentes para así comunicarse a través de la línea de comandos.

Chef

Es una plataforma de código abierto de automatización escrita en Ruby que transforma infraestructuras complejas en código, Chef automatiza la forma en la que las aplicaciones se configuran, despliegan y gestionan en una red sin importar su tamaño.

Transformar la infraestructura a código permite tener una infraestructura con control de versiones, redundante y comprobable. [27]

Componentes de Chef

N. Sabharwal y M. Washwa detallan los componentes de Chef en [28]:

- ***Cookbooks:*** colección de todos los componentes necesarios para realizar cambios en dispositivos, siendo el componente más importante las recetas pues estas le especifican a Chef qué recursos deseamos configurar en los hosts. Los *cookbooks* deben ser desplegados en los nodos en los que se desea realizar cambios o configuraciones. En general, Chef utiliza Ruby para escribir *cookbooks* y recetas
- **Servidor Chef:** es donde se tienen las configuraciones de datos para cada nodo registrado en el servidor. El servidor de Chef es el encargado de distribuir los *cookbooks* colocados por el administrador en el servidor hacia los nodos gestionados.
- **Estación de trabajo**: es un sistema utilizado para gestionar Chef, pueden existir múltiples estaciones de trabajo para un solo servidor de Chef. En una estación de

trabajo se desarrollan los *cookbooks* y recetas, se gestionan nodos y se sincroniza el repositorio de Chef. Los componentes principales de una estación de trabajo son:

  - ***Knife***: línea de comandos utilizada para interactuar con el servidor de Chef. Con *Knife* se pueden gestionar nodos, subir *cookbooks* y recetas y administrar roles.
  - **Repositorio local de Chef:** lugar en el que se almacena la información relacionada a los servidores y nodos Chef.

- **Nodo:** es un cliente registrado en el servidor de Chef que tiene un agente llamado cliente Chef. Existen dos tipos de nodos que Chef puede gestionar, los nodos basados en la nube (virtuales) y los nodos físicos. Los nodos tienen dos componentes principales que son:
  - **Cliente Chef:** es el agente que corre en cada nodo, el agente se encarga de comunicarse con el servidor de Chef y obtener la configuración que el nodo necesita. RabbitMQ es un intermediario de mensajes entre el cliente Chef y el servidor, generalmente maneja y envía las colas de mensajes.
  - **Ohai:** es una herramienta integrada en Chef que es utilizada para detectar ciertos atributos en un nodo en particular para luego enviar esa información al servidor de Chef cuando sea requiera, Ohai puede ser utilizado para descubrir nodos.

La Figura 2. 5 detalla los componentes principales que participan en el funcionamiento de Chef para automatizar la gestión y configuración de red.

El administrador de red utiliza la estación de trabajo y, mediante *Knife* y el repositorio Chef disponible genera los *cookbooks* y escribe las políticas que posteriormente son colocadas en el servidor Chef.

Los nodos gestionados tienen instalado un agente denominado cliente Chef que les permite comunicarse con el servidor y obtener las configuraciones a ser ejecutadas en cada uno de ellos, como se aprecia en la imagen, los nodos pueden ser físicos o virtuales.

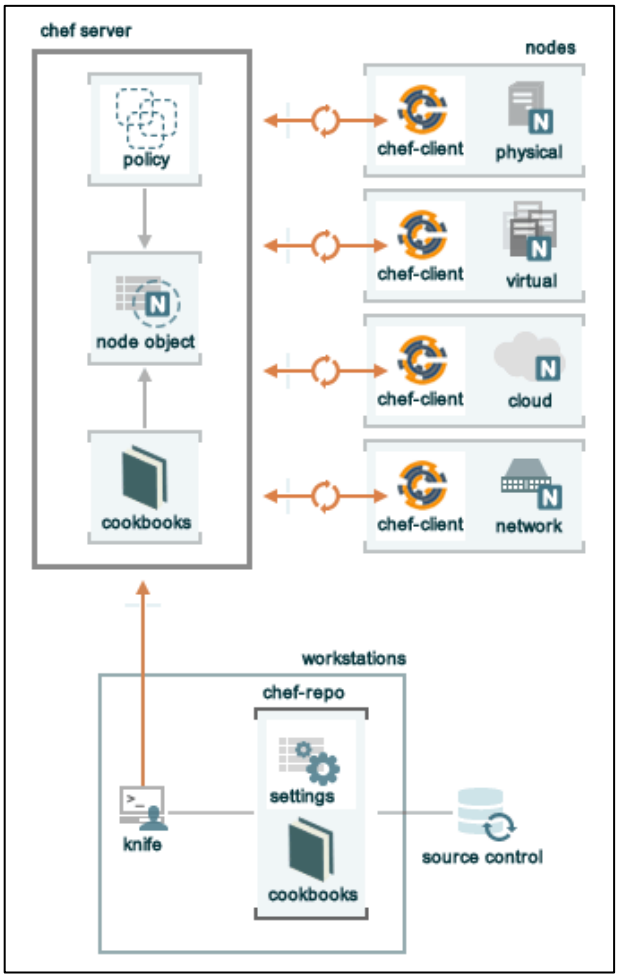

Figura 2. 5 Esquema de trabajo de Chef. (Fuente: *[29]*)

Ventajas

Algunas de las ventajas de Chef que pueden listarse al analizar [28] son:

- Permite que infraestructuras de red complejas y extensas puedan transformarse en código facilitando así su administración y modelado.
- Chef permite flexibilidad pues puede utilizarse para trabajar en modo distribuido o centralizado, puede usar también modelos *push* y *pull* o ambos.
- Puede ser desplegado en infraestructuras locales o en la nube, adaptándose fácilmente a cualquier ambiente.
- Haciendo uso de Chef puede describirse cualquier recurso de la red en código, sin importar cuanto difiera de la configuración estándar.

- Varios de las configuraciones estándar para gestión de redes están descritas en los *cookbooks* de Chef y están disponibles de forma gratuita en un sitio denominado *Supermarket* Chef, lo que hace más sencillo el empezar a utilizar Chef.

- Chef puede ser utilizado para automatizar completamente el despliegue de la red, incluyendo desarrollo interno y sistemas de usuarios finales. También puede ser utilizado para automatizar el escalamiento de infraestructura y hacer que la red sea auto reparable.

Chef en redes IoE

Chef es una herramienta que requiere un conocimiento profundo de Ruby, sin embargo, su utilización en redes IoT e IoT es posible gracias a que permite la implementación de proxys en nodos de red con recursos limitados que no permiten la instalación de agentes. Su utilización podría complicarse debido al lenguaje con el que se implementa esta herramienta, pero es la herramienta más utilizada a nivel empresarial, por lo que está muy preparada ante fallos y posibles errores del sistema.

SaltStack

Desarrollado por SaltStack, es un sistema de gestión y configuración de red escrito en Python y es también, un framework de ejecución remota similar a Chef, Puppet y Ansible.

Lo que hace resaltar a Salt ante sus competidores es que logra alcanzar objetivos a un alto nivel mediante un sistema de comunicación rápido y seguro, además es capaz de manejar pocos nodos o miles de nodos a altas velocidades. Para efectuar configuraciones en Salt se cuenta con varias opciones, pero, por defecto se utiliza YAML [30].

El proyecto SaltStack se desarrolló como un sistema de gestión y configuración basado en Python, fundado en 2011 por Thomas Hatch. Salt utiliza una arquitectura basada en un servidor central con agentes denominados minions que corren en los dispositivos a ser gestionados.

Componentes

Según [31] los principales componentes de Salt son:

- **Maestro:** dispositivo centralizado que coordina las acciones de los nodos de red.

- **Minions:** es el agente instalado en cada host o nodo de red a ser gestionado, cada minion tiene un ID único, que usualmente es el nombre de host, para facilitar su gestión.
- **Estados:** son una forma de gestionar las configuraciones a través de los minions, u estado de Salt es una forma de definir una platilla para configurar un host.
- **Zero MQ:** bus de mensajes de alta velocidad que conecta el maestro Salt con el minion Salt.
- **Módulos de ejecución:** se utilizan para ejecutar acciones, estos módulos se ejecutan generalmente como comandos de una sola vez, sus resultados son presentados como datos en formato JSON para que puedan analizarse y usarse en otros flujos de trabajo.
- **Módulos de estado:** utilizan los módulos de ejecución para hacer que un dispositivo alcance el estado deseado, permitiendo al usuario definir un estado objetivo para uno o varios dispositivos de la red. Los módulos de estado usan definiciones de estado escritas como archivos SLS (Salt State) escritos en cualquier lenguaje, siendo el lenguaje por defecto YAML.

Salt corre como un proceso en segundo plano tanto en el minion como en el maestro. Cuando el maestro se activa provee una ubicación mediante sockets en la que los minions pueden acceder para encontrar los comandos con los que deben configurarse. Cuando el minion inicia este se conecta al socket del maestro y escucha esperando por algún evento. Inicialmente existe un proceso de handshake entre el minion y el maestro en el que se confirma el ID del minion a ser configurado

La comunicación entre el maestro Salt y los minions se da mediante claves de mensajes ZeroMQ. Almacenados en el maestro dentro del servidor de archivos denominado "*file_roots*" se encuentran los módulos de estado; otros datos, como las variables, se guardan en un almacén de datos denominado Salt Pillar; esta distinción de almacenamiento permite separar la lógica (estados) de los datos (variables). Las variables pueden ser definidas también localmente en el archivo de estado, estas variables se denominan "*grains*"

Los minions recuperan los archivos de definición de estado y otros datos del Salt Pillar y del servidor de archivos sobre el bus central de mensajes [31].

En la Figura 2. 6 se puede apreciar lo descrito previamente en cuento al esquema de trabajo de Salt.

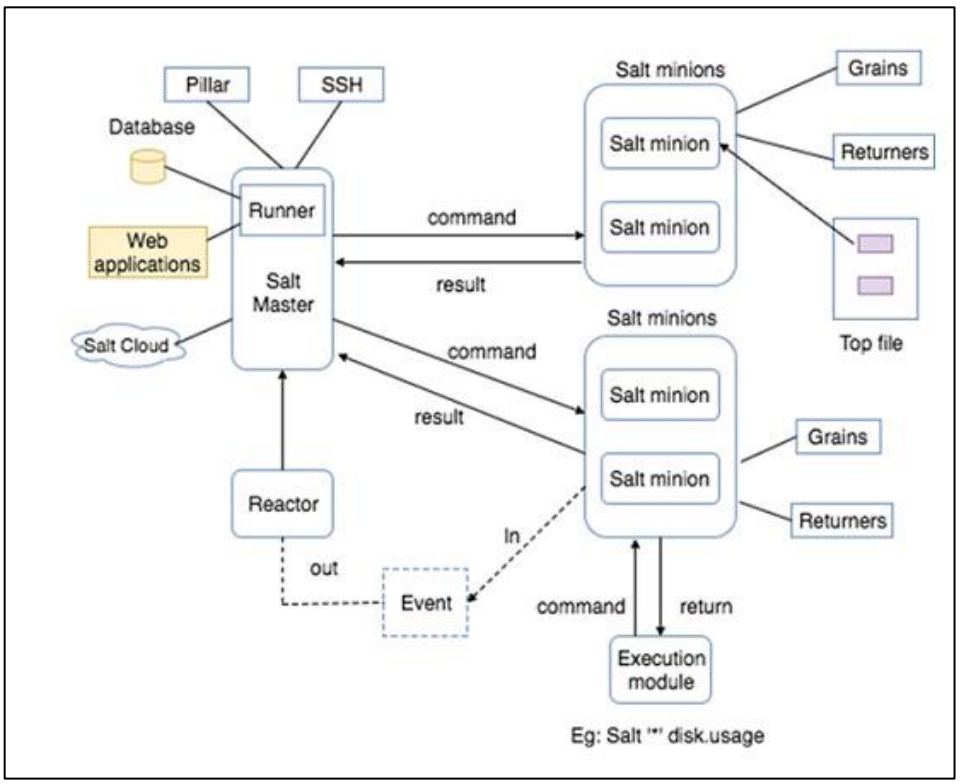

Figura 2. 6 Esquema de trabajo de Salt. (Fuente: *[26]*)

Ventajas

En [32] se destacan las siguientes ventajas de Salt:

- Toda la comunicación se Salt se realiza mediante un canal encriptado.
- Opciones de topología: los minions pueden dividirse y comunicarse con un maestro de sindicato que actuará como un proxy para el maestro, esto es útil cuando se tienen clústeres de host geográficamente dispersos permitiendo que las conexiones intra-clústeres tengan una conexión rápida a pesar de que exista latencia entre clústeres.
- Salt proporciona una extensa librería y numerosas estructuras de datos que permite que los módulos personalizados ingresen a Salt y extraigan datos o ejecuten otros módulos, lo que se denomina módulos de uso extendido. Esto permite a posterior crear módulos utilizando estados de Salt propios y lógica personalizada.

- El formato de los archivos de estado de Salt no necesariamente debe ser YAML, sino que puede ser variado siempre que se utilice un renderizador para convertir estos datos en estructuras de datos que Salt pueda manejar.
- Para los dispositivos que no soportan agentes un proceso denominado “proxy-minon” puede ser ejecutado en un servidor que luego se comunicará con el dispositivo sin agente usando sus protocolos nativos.
- Es capaz de ejecutar comandos en paralelo en sistemas remotos, acelerando el proceso de automatización.
- Permite la automatización de procesos de trabajo soportando Puppet, Chef, Git, Docker, etc. y es efectivo para una pronta escalabilidad de red y rápida respuesta a fallos.

SaltStack en redes IoE

SaltStack es una herramienta útil para aplicar en redes IoT e IoE debido a:

- La escalabilidad que provee ya que su configuración permite integrar cientos de miles de minions para los muchos nodos que conforman una red extensa.
- Asigna estados a los nodos de red y reacciona ante los eventos que ocurren dentro del sistema y la red, permitiendo solventar fallas o posibles causas de fallas a futuro. Una red IoT no solo trata de enviar y recibir datos, sino que requiere enfrentar los eventos de cada sistema que la conforma.

Ansible, Puppet, Chef y Salt comparativa

La Tabla 2. 2 resume las características generales de las 4 herramientas de automatización previamente descritas.

La fila correspondiente a lenguaje de configuración especifica el lenguaje utilizado para escribir los archivos de configuración que serán distribuidos a los nodos gestionados, para el caso de Ansible y Salt resulta más sencilla la configuración pues se utiliza YAML, que, a pesar de tener limitaciones, es de fácil comprensión y muy flexible con otros lenguajes.

Respecto a la compatibilidad con Sistemas Operativos se tiene que Ansible es la única herramienta que no es compatible con Windows, sin embargo, es la única herramienta que puede ser utilizada sin agentes lo que la hace óptima para redes IoE ya que varios de los nodos que conformen una red IoE tendrán recursos limitados y la imposibilidad de tener un

agente instalado. Aunque SaltStack es una herramienta híbrida, fue diseñada para trabajar con agentes, por lo que al utilizar el modo sin agentes trabaja más lento.

Tabla 2. 2 Características generales de las herramientas de automatización de redes.

	Ansible	**Puppet**	**Chef**	**Salt**
Lenguaje de desarrollo	Python	• C++ • Clojure • Ruby	Ruby	Python
Lenguaje de configuración	YAML	• Ruby • Puppet DSL	Ruby	YAML
Arquitectura	-	Cliente-servidor	Cliente-servidor	Cliente-servidor
Tipo de herramienta	Sin agente	Basada en agente	Basada en agente	Híbrida
Sistemas Operativos compatibles	• Linux (RHEL, CentOS, Fedora, Ubuntu, Debian) • Unix (macOS, FreeBSD, Solaris)	• Linux • Sistemas UNIX • Windows	• Unix (Solaris, AIX, macOS, FreeBSD) • Linux (Amazon, CentOS, Debian, Oracle Enterprise, Red Hat Enterprise, SUSE Server, Ubuntu LTS) • Windows.	• Linux (Red Hat, Ubuntu, Debian, CentOS, SUSE) • Unix (AIX, Solaris, macOS) • Windows
Código abierto	✓	✓	✓	✓

La Tabla 2. 3 refleja una comparación referente al esquema de trabajo de cada herramienta de automatización.

- La configuración de entrega se refiere a la forma en la que las configuraciones son entregadas a los nodos gestionados, en el caso de las herramientas basadas en agente la comunicación entre el servidor y los nodos se realiza a través del agente instalado en cada nodo:

- Para el caso particular de Puppet 4 se utiliza una herramienta intermediaria de mensajes que controla el funcionamiento del agente Puppet, esta herramienta se denomina MCollective y soporta el protocolo SSL.
- Chef hace uso del intermediario de mensajes RabbitMQ para gestionar los agentes Chef y las colas de mensajes entre el servidor y los nodos gestionados.
- ZeroMQ permite la comunicación entre el maestro y los minios en el caso de SaltStack permitiendo el intercambio de información mediante claves de mensajes.
- SSH se utiliza para acceder a los nodos gestionados que no cuentan con un agente instalado, lo que resulta beneficioso en el caso de dispositivos que no soportan la instalación de agentes. Como se muestra en la tabla todas las herramientas utilizan SSH cuando utilizan proxys y no agentes para el envío de información a los nodos gestionados.

- La fila correspondiente a método de entrega resume la forma en la que se obtienen las configuraciones de los servidores. En los métodos de entrega *push* las configuraciones son empujadas hacia los nodos mientras que en los métodos *pull* los nodos son quienes automáticamente extraen las configuraciones del servidor central. Puppet puede utilizar cualquiera de los dos métodos de entrega para configurar sus nodos.
- En lo referente a la interoperabilidad y escalabilidad, todas las herramientas son compatibles con sistemas Linux, Unix y Windows por lo que cuentan con la característica de interoperabilidad, además están diseñadas para permitir un crecimiento de red y brindad escalabilidad.
- En lo que respecta a disponibilidad:
  - Chef cuenta con un servidor de respaldo en caso de fallar el servidor principal.
  - Puppet cuenta con arquitectura multi maestro lo que significa que si el maestro falla otro servidor tomará su lugar.
  - En Ansible existe un nodo activo ejecutándose denominado instancia principal si esta llegara a fallar una instancia secundaria se activa.

- Salt puede utilizarse configurando múltiples maestros que reemplazarán al maestro principal en caso de fallos.

Tabla 2. 3 Funcionalidad de las herramientas de automatización de redes.

	Ansible	**Puppet**	**Chef**	**Salt**
Configuración de entrega	SSH	• Mcollective (SSL) • SSH (proxy)	• RabbitMQ • SSH (proxy)	• ZeroMQ • SSH
Método de entrega	Push	Push, Pull	Pull, (Push en la versión corporativa)	Pull
Instrucciones para configuración	Playbook	Manifest	Cookbook	Pillar
Disponibilidad	✓	✓	✓	✓
Escalabilidad	✓	✓	✓	✓
Interoperabilidad	✓	✓	✓	✓

La Tabla 2. 5 señala recomendaciones de uso de cada herramienta según [33]. Las ventajas y desventajas de cada herramienta se presentan resumidas en la **¡Error! No se encuentra el origen de la referencia.**, misma que no pretende destacar una herramienta sobre otra, sino que destaca los puntos a considerarse en la elección de una herramienta dependiendo de la situación y entorno en la que puede ser aplicada.

Tabla 2. 4 Recomendaciones de uso de herramientas de automatización de gestión de red.

	Ansible	**Puppet**	**Chef**	**SaltStack**
¿Qué herramienta usar?	Prioriza la simplicidad y el tiempo.	Se busca la estabilidad y madurez de la gestión de red.	Aplicable para soluciones a sistemas multinivel en entornos centrados en el desarrollo.	Enfatiza la escalabilidad y resiliencia.

Tabla 2. 5 Ventajas y Desventajas de las herramientas de automatización de gestión de red.

	Ventajas	Desventajas
Ansible	• Gestión de configuración de red. • Monitoreo de flujo de trabajo. • Despliegue automatizado de aplicaciones. • Fácil de usar. • Fácil de aprender.	• Menos eficaz que otras herramientas basadas en otros lenguajes de programación. • La lógica de implementación es con DSL por lo que se requiere una frecuente revisión de la documentación. • Requiere de registros de variables incluso en las funcionalidades más básicas.
Puppet	• Gestión y monitoreo de red. Funciona en variedad de plataformas. • Probada en ambientes con altas demandas con resultados excelentes. • Establecida comunidad de soporte. • Fuertes capacidades de reporte. • Cuenta con la interfaz web más completa	• Requiere programadores con sólidos conocimientos en Ruby. • El código base puede volverse complejo. • Se aplica con enfoques basado en modelos, lo que implica menos control que los enfoques basados en código.
Chef	• Gestión y monitoreo de red. Ofrece más de 800 módulos gratuitos. • Soporta múltiples ambientes. • Su interfaz de comandos, modo de prueba y extensa base de datos le da grandes capacidades de almacenamiento. • Altamente personalizable. • Su enfoque basado en código permite flexibilidad en las configuraciones de control	• Requiere conocimientos de Ruby. • Manejo de grandes bases de datos complica su utilización.
SaltStack	• Diseñada para operaciones a nivel de empresas. • Soporta múltiples hosts simultáneamente. • Fácil de aprender. • Se adapta a cualquier lenguaje. • Permite agrupación de clientes y generación de platillas para simplificar el control del entorno.	• La interfaz web es la menos completa pues es la más reciente. • Su uso se complica cuando el sistema operativo en el que se ejecuta no es Linux.

Lenguajes utilizados para automatización de la gestión de red

La programabilidad de red permite que las configuraciones de infraestructura y gestión de red se simplifiquen, las implementaciones de automatización se basan en métodos genéricos de programación e interfaces estándar como SSH o servicios Web REST [34]

Los lenguajes de secuencias de comandos, conocidos como lenguajes de *scripting*, no son compilados, pero son interpretados a través de un entorno adicional por lo que requiere que los comandos sean convertidos a formato de máquina línea a línea, a diferencia de los lenguajes de programación que pueden ejecutarse sin la necesidad de entornos extra.

Entre los lenguajes de scripting utilizados para automatización se tiene: YAML, Python, Ruby y *Shell scripting*. La preferencia entre éstos dependerá de la familiaridad del usuario con cada uno, sin embargo, Python se ha convertido en el preferido de los usuarios por su simplicidad y su constante desarrollo. K. Hitchcock en [33] habla brevemente de YAML, Ruby y Python de lo que se destaca lo siguiente:

YAML

Es un lenguaje de secuencia de comandos que utiliza una sintaxis simple y de fácil entendimiento, se emplea principalmente para el diseño de archivos de configuración por lo que es utilizado por dos herramientas populares de automatización como lo son Ansible y SaltStack. YAML soporta una amplia variedad de datos incluyendo listas, matrices y diccionarios y es compatible con otros lenguajes como Java, Python y Ruby.

Ruby

Ruby es un lenguaje de programación multipropósito de alto nivel, es similar a Python, pero difiere en el manejo de variables puesto que Ruby mantiene sus variables privadas de la clase y solo las expone si son accedidas por métodos de lectura o escritura. Ruby es un lenguaje orientado a objetos por lo que es influenciado fuertemente por lenguajes como C++, Java y Perl.

Cuenta con una variedad de funciones integradas lo que facilita la elaboración de scripts, además, es flexible pues no se restringe al desarrollador y se admite alteraciones de acuerdo con las necesidades de cada usuario. Generalmente se utiliza Ruby para escribir funciones de tareas de automatización, Puppet es una herramienta de automatización que implementa módulos haciendo uso de Ruby.

Python

Python cuenta con un entorno de desarrollo integrado flexible (*Integrated Development Environment*, IDE) que contribuye al desarrollo de aplicaciones. A. Ratan describe en [35] varias librerías que facilitan la automatización de red.

Netmiko y Paramiko son complementos de Python que se utilizan para el control de dispositivos aplicando SSH, son útiles para la gestión y configuración remota de dispositivos que no soportan APIs.

- Paramiko es una interfaz de Python que implementa el protocolo SSHv2 y proporciona funcionalidad de cliente y servidor, es ventajoso en la obtención de alto desempeño en conceptos criptográficos de bajo nivel.

- Netmiko es una librería de mútiples proveedores basada en Paramiko. Su característica multi proveedor permite la configuración de equipos de distintas marcas. Netmiko tiene como objetivo lograr la automatización de red, recopilación de información y configuración de equipos. Python es utilizado mayoritariamente por Ansible y SaltStack.

2.2 Entornos de trabajo basados en la nube para la gestión de redes IoT

La computación en la nube (*Cloud Computing*) ofrece acceso ubicuo y bajo demanda a la red y todos sus recursos permitiendo un rápido aprovisionamiento, configuración y monitoreo realizando un esfuerzo mínimo.

Las plataformas de Cloud son las encargadas de proveer este servicio y, debido a la naturaleza de los dispositivos que componen una red IoT e IoE han emergido plataformas de Cloud para dispositivos con recursos limitados. M. Aboubakarab, M. Kellila y P. Rouxa en [1] presentan una descripción de las plataformas para redes IoT, refiriéndose a las bibliografías en las que se explica a detalle las características de cada una.

En la Tabla 2. 6 se resumen los atributos más relevantes de algunas de las plataformas en la nube para el internet de las cosas, considerando dispositivos de bajo consumo de energía.

Tabla 2. 6 Comparativa de las plataformas en la nube para el IoT

Plataforma	Protocolo de gestión	Gestión y configuración	Monitoreo de dispositivos	Dispositivos de recursos limitados
Azure	LWM2M	✓	✓	✓
IBM IoT	LWM2M	✓	-	✓
Artik Cloud	LWM2M	✓	-	✓
Echelon	-	✓	-	-
Google Cloud	-	✓	✓	✓
Autodesk Fusion Connect	LWM2M	✓	-	-
Amazon Website Site (AWS) IoT	-	✓	✓	✓

En la Tabla 2. 6**¡Error! No se encuentra el origen de la referencia.** encontramos plataformas que son útiles para la gestión de redes IoE por sus características y aplicabilidad en dispositivos de recursos limitados, como es el caso de Azure, Google Cloud o AWS, sin embargo, la plataforma Echelon [36] no está diseñada para dispositivos de recursos limitados ni para el monitoreo de red lo que complica su aplicación en redes IoT e IoE, sin embargo, resulta importante de mencionar pues utiliza tecnologías de comunicación como WiFi y Ethernet.

Estos entornos de trabajo pueden considerarse como herramientas para la gestión de redes IoE pues, al estar equipadas y diseñadas para la gestión de redes IoT se convierten en un punto de partida siendo plataformas base útiles que facilitan la administración de redes extensas, por lo que pueden soportar el integrar procesos, datos y personas a las redes IoT y aun así facilitar y ayudar a su gestión.

2.3 Entornos de trabajo basados en aprendizaje de máquina para la gestión de redes IoT

El aprendizaje de máquina, como se ha mencionado en secciones anteriores, es muy útil para implementar una autogestión de red y perfeccionamiento continuo, es decir, las técnicas del ML permiten que la gestión de red sea automatizada facilitando así la administración y monitoreo de redes extensas. Varios entornos de trabajo han sido

propuestos para la gestión de red con ML, en su mayoría se utiliza un enfoque de clasificación de flujos de tráfico IP para la identificación efectiva de aplicaciones de red [37].

En [1] se presentan varios entornos de trabajo que utilizan técnicas de ML para gestionar redes extensas, entre los que se pueden mencionar se tiene:

- Red de sensores inalámbricos inteligentes (*Intelligent Wireless Sensor Network*, IWSN): propuesta para la gestión de rutas de datos por dispositivos IoT, esta solución se basa en una red neuronal que permite seleccionar la ruta que optimice el desempeño de la red en caso de que un nodo falle.
- Sistema centralizado de detección de intrusos (*Centralized Intrusion Detection System*, IDS): se basa en vectores de soporte y ventanas deslizantes para redes de sensores, principalmente detecta intrusos y ataques de agujero negro.
- Aprendizaje por refuerzo independiente distribuido (*Distributed Independent Reinforcement Learning*, DIRL): utilizado en la gestión de tareas y recursos en redes de sensores, permite a cada nodo organizar sus tareas de forma autónoma y autogestionar sus recursos al aprender los procesos.
- Enrutamiento de retroalimentación a múltiples sumideros (*Feedback Routing to Multiple Sinks*, FROMS): ayuda en la definición de óptimas rutas multicast usando diferentes métricas de costo, FROMS permite una rápida recuperación en caso de fallas.

La Tabla 2. 7 resume algunos de los entornos propuestos.

Tabla 2. 7 Entornos de trabajo de automatización de redes IoT con ML.

Entorno de trabajo	Requerimientos que cumple el entorno de trabajo						Técnica ML
	Escalabilidad	Tolerancia a Fallos	Energía eficiente	QoS	Seguridad	Auto configuración	
IWSN	-	✓	✓	✓	-	✓	Aprendizaje no supervisado
IDS	-	-	-	-	✓	✓	Aprendizaje supervisado
FROMS	✓	✓	✓	-	-	✓	Aprendizaje de refuerzo
DIRL	-	-	-	✓	-	✓	Aprendizaje de refuerzo

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 Resultados

Existe un amplio catálogo de herramientas para automatizar la gestión de red, las herramientas analizadas en la sección 2.1 del presente trabajo de integración curricular, son de las pocas herramientas que son compatibles y óptimas para la configuración de equipos de red y dispositivos de recursos limitados; esto, debido a que pueden en su mayoría implementarse sin la necesidad de un sistema operativo propietario y, sin la instalación de agentes, facilitando la comunicación mediante proxys.

Cada herramienta de automatización analizada es capaz de alcanzar los mismos objetivos, difiriendo únicamente en la forma en la que lo logran, para lo que se consideran los diferentes componentes que utilizan, sus arquitecturas, tipo de comunicación y los distintos esquemas de funcionamiento de cada una. La elección de una u otra herramienta dependerá del enfoque y alcance que tenga el trabajo a realizarse o incluso de la familiaridad del programador o equipo de trabajo con los lenguajes que predominan en cada herramienta, recomendaciones de la aplicación de una herramienta específica se detallan en la Tabla 2. 5.

Las tendencias de uso y preferencias entre herramientas pueden visualizarse gracias a una encuesta descrita en [22], la encuesta de desarrolladores de Python efectuada en 2018 [38] expone los resultados mostrados en la Figura 3. 1, evidenciando que el 65% de usuarios prefiere no utilizar herramientas de automatización de configuración de gestión de red, pero entre los que lo hacen, Ansible es el preferido por la popularidad y acogida de Python.

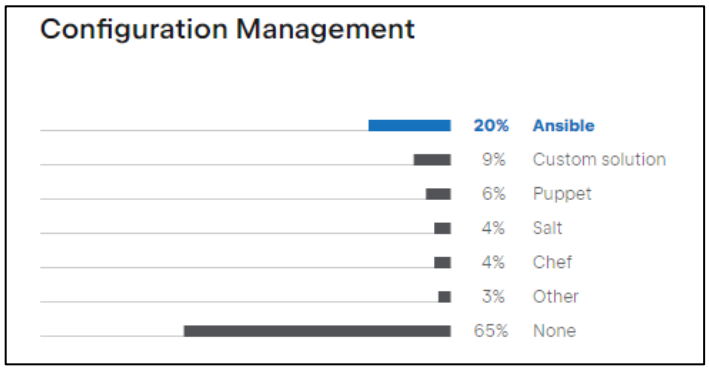

Figura 3. 1 Resultado de la encuesta de la Fundación de Software de Python para la gestión de configuración. (Fuente: *[38]*)

Dentro del tema de automatización de gestión de redes IoE también se analizaron entornos de trabajo basados en aprendizaje de máquina y entornos basados en la nube.

Según la comparativa de los entornos de trabajo basados en la nube presentada en la Tabla 2. 6 aquellas plataformas que cumplen con monitoreo, gestión y aplicación en dispositivos de recursos limitados (como sensores) son Azure, AWS y Google Cloud; de éstas únicamente Azure utiliza el protocolo de gestión LWM2M para un bajo consumo de energía, lo que resulta sumamente importante en redes IoE, pues están compuestas mayoritariamente por dispositivos de recursos limitados, que se beneficiarían de la aplicación del protocolo de bajo consumo de potencia LWM2M.

En la comparativa de entornos de trabajo basados en el aprendizaje de máquina presentada en la Tabla 2. 7 los distintos entornos de trabajo se pueden implementar con cualquiera de las técnicas de aprendizaje de máquina, como lo es el aprendizaje supervisado, no supervisado y el de refuerzo. La tabla resalta dos entornos que cumplen con la mayoría de los requerimientos de una red IoT:

- IWSN que utiliza el aprendizaje no supervisado es tolerante a fallos, maneja eficientemente la energía, ofrece QoS y autoconfiguración, sin embargo, no ofrece seguridad ni escalabilidad.
- FROMS ofrece escalabilidad, pero carece de QoS y seguridad.

Los entornos analizados continúan en desarrollo y siguen adaptándose a las necesidades y requerimientos de gestión de redes complejas. Estas plataformas se han desarrollado pensando en los dispositivos de recursos limitados que conforman comúnmente redes IoT, debido a la naturaleza de estos dispositivos resulta un reto la implementación de un entorno que ofrezca seguridad, escalabilidad y QoS.

Para continuar con el análisis de los resultados se realizará una breve distinción de las ventajas, desventajas y retos en la implementación de la automatización de gestión red en la sección 3.2 y finalmente en la sección 3.3 se detallarán brevemente escenarios de implementación de automatización de gestión y monitoreo de redes IoT que hacen uso de las herramientas de automatización previamente descritas.

3.2 Análisis de las ventajas y desventajas de la automatización de la gestión de redes IoE

Ventajas y desventajas

Tabla 3. 1 Ventajas y desventajas de la automatización de redes IoE.

Automatización de la gestión de redes IoE	
Ventajas	**Desventajas**
• **Incremento de la eficiencia de la red:** al autogestionarse, una red extensa responde rápidamente a los problemas y fallas que pueden presentarse y se reducen errores de configuración humana. • **Rápido despliegue de red y escalabilidad:** la red es capaz de autoconfigurar sus dispositivos y dar una respuesta rápida a los cambios dinámicos de red para acoplar nuevos nodos ofreciendo así un despliegue de red más rápido y más disponibilidad de red. • **Infraestructura como código:** las herramientas de automatización permiten transformar la infraestructura de red en código, haciendo posible una configuración y manejo más sencillo de la red. • **Ahorro de tiempo y dinero:** automatizar la gestión de red reduce el tiempo de configuración de los dispositivos pues ya no se lo hace de forma manual, así mismo reduce	• **Curva de aprendizaje:** para automatizar la gestión de red se requiere conocimiento de lenguajes de scripting, lenguajes de programación y el uso de adecuado de cada herramienta, plataforma o entorno de administración. • **Costo:** los verdaderos beneficios de las herramientas de automatización se obtienen al adquirir las versiones de pago, lo que permite aprender mejor sobre las capacidades de la herramienta y de esta forma se puede aplicar mejor a las necesidades de la red. Esto son embargo, representa un costo adicional. • **Automatización innecesaria:** es vital distinguir cuándo una actividad de gestión debería automatizarse y cuándo no, pues si son tareas que solo se ejecutan una vez o pocas veces, resulta más sencillo realizarlas manualmente antes que escribir un complejo código para que se realice de forma autónoma.

costos para solventar errores humanos y de mantenimiento. • **Gestión más acertada en redes complejas:** la administración de red tradicional es aplicable siempre y cuando las redes no sean complejas, caso contrario la gestión se extenderá a cientos de miles de dispositivos heterogéneos que forman parte de la red, volviéndose una tarea en extremo complicada y propensa a errores para realizarla de forma manual.	

Como se ha analizado en este trabajo de integración curricular, la automatización de gestión de red para redes complejas representa una mejora en la eficiencia del desempeño de red, reducción de errores humanos y mayo eficiencia y disponibilidad.

Los beneficios que se obtienen de la automatización son varios y contrarrestan fallas de forma más rápida que una gestión de red tradicional, sin embargo, entre las desventajas listadas en la Tabla 3. 1 sobresale el problema de la curva de aprendizaje pues la aplicación de herramientas, plataformas o entornos de automatización dependerá de la familiaridad del equipo de trabajo con los lenguajes de scripting utilizados por cada herramienta, plataforma o entorno, así como también el tiempo de aprendizaje que les tome para poder usarlas correctamente.

Retos de implementación

- **Dispositivos con características limitadas:** las redes IoE se componen de nodos de red variados que pueden ser computadores, servidores, celulares, sensores, electrodomésticos, cámaras e incluso personas. Muchos de estos dispositivos no cuentan con las características necesarias para permitirse la instalación de un sistema operativo ni mucho menos para soportar largas jornadas de trabajo pues recursos como su batería y memoria son limitadas. La automatización de gestión de red requiere de constante actividad de los dispositivos y, en ocasiones la

instalación de agentes para su gestión, por lo que es necesario implementar protocolos de gestión que prioricen el bajo consumo de potencia y manejar herramientas adecuadas que se permitan utilizar proxys con el fin de gestionar dispositivos con características limitadas.

- **Dispositivos heterogéneos:** encontrar una arquitectura, protocolo o herramienta de automatización que funcione de igual forma para cada nodo en una red IoE es prácticamente imposible pues debido a su vasta topología y variedad de tipos de dispositivos esto representa un reto pues no existe como tal una arquitectura estandarizada para la gestión de redes IoT.

- **Escalabilidad:** debido al crecimiento exponencial de redes IoT la escalabilidad representa requerimiento crítico pues la mayoría de las soluciones para la automatización de gestión en dispositivos de características limitadas no direccionan bien la escalabilidad, esto podría solventarse con el *fog computing* que es una tecnología que aprovecha la nube y los dispositivos de borde para acelerar la reconfiguración de redes IoT con dispositivos que requiere un bajo consumo de potencia.

- **Gestión de red en tiempo real:** las herramientas de automatización gestionan la red de forma asíncrona, pues el hacerlo en tiempo real implica la sobrecarga de tráfico por las constantes operaciones de gestión de red lo que representa un reto para los dispositivos de recursos limitados, sin embargo, la aplicación de gestión en tiempo real podría incurrir en una máxima disponibilidad de red.

3.3 Casos de estudio - escenarios de implementación

Actividad “F&CF availability”

En [39] se habla de una actividad de retroalimentación de monitoreo rápido y continuo de la disponibilidad de sistemas IoT (*Fast & Continuous Monitoring Feedback of System Availability*, *“F&CF availability”)* que busca monitorear de forma temprana la disponibilidad de los sistemas IoT en producción e informar continuamente sobre esto. *“F&CF availability”* pretende automatizar el monitoreo de redes IoT mediante herramientas de monitoreo y automatización, sin embargo, las soluciones deben ser implementadas manualmente por el equipo correspondiente.

Para la implementación de la actividad “F&CF Availability” se utiliza un conjunto de archivos de configuración, scripts, y archivos de código de Python que permitan un control de versiones.

La gestión del despliegue y ejecución está a cargo de los *playbooks* que Ansible ejecuta en servidores seleccionados.

Requisitos

- Sistemas IoT que provean servicios de monitoreo en áreas críticas como por ejemplo en el área de salud.
- Hace uso del metamodelo de ingeniería de procesos de software y sistemas (Software and Systems Process Engineering Metamodel, SPEM)
- Herramientas de monitoreo como Netdata, Pandora FMS, Azure, AWS.
- Herramientas de automatización como Ansible, Puppet, SaltStack, Chef.

Objetivos

- Monitoreo constante para prevenir y contrarrestar anomalías.
- Proveer una retroalimentación que influya positivamente en el mantenimiento y las actualizaciones.
- Detectar rápidamente problema y obtener información de cómo solucionarlos antes de que los clientes se vean afectados.

Características

- Configuración de infraestructura IoT reproducible y con control de versiones, denominada monitoreo como código (*monitoring as code*, MaC).
- Es fácilmente adaptable
- Está compuesta por un conjunto de actividades que son:
    - **Crear telemetría:** esta tarea se encarga de la lectura y recolección de los valores de las métricas de monitoreo.
    - **Analizar telemetría:** con esta tarea se detectan las fallas y anomalías de los valores previamente recolectados.

- **Supervisión:** esta actividad tiene como función principal el diagnóstico de problemas y la retroalimentación de los problemas y sus posibles soluciones.

- La especificación SPEM describe un conjunto de herramientas que soportan el conjunto de actividades que componen la actividad "*F&CF Avalability*".

MDE para administración de sistemas IoT e IoE

La ingeniería basada en modelos (*Model-Driven Engineering*, MDE) soportan las necesitades de la gestión de configuración de redes, en el contexto de redes IoT e IoE la administración se vuelve compleja por lo que en [40] se propone un modelo para la gestión automatizada de redes IoT utilizando Puppet y UML.

Requerimientos

- **Requerimientos de lenguaje de modelado y scripts**
  - **Lenguaje de modelado:** debe expresar todos los elementos que conforman la red y las relaciones entre ellos, debe ser flexible para acoplar nuevos dispositivos a la red.
  - **Estandarización de modelos:** el lenguaje de modelado debe seguir estándares para un mejor entendimiento e integración.
  - **Generación de scripts:** los scripts se derivan del modelo de configuración y pueden estar en distintos formatos.
- **Requerimientos de herramientas de soporte**
  - **Editor de modelo:** implementa el lenguaje de modelado para obtener la estructura de red y diagramas de configuración.
  - **Aplicación de la configuración generada:** una herramienta de automatización de gestión de red debe aplicar y distribuir los archivos de configuración generados a partir del modelo, a la red objetivo.

Funcionamiento

Se modela la topología identificando cada elemento perteneciente a la red, ya sea un dispositivo físico o lógico utilizando el lenguaje unificado de modelado (UML). Los elementos de red deben ser caracterizados y debe especificarse si se trata de un sensor, un elemento virtual o, si son elementos con funciones específicas como un switch o router

cuyas funciones difieren a las de un servidor o un computador. El modelado propuesto se encuentra detallado en [40].

Para la generación de scripts de configuración se utiliza la herramienta Puppet. Se define un mapeo entre la red y el perfil de configuración, se debe transformar el modelo a archivos de configuración. La transformación toma como entrada el modelo definido con UML y se generan los scripts de configuración en archivos de formato Puppet que serán importados por Puppet para configurar la red. Los archivos Puppet generados incluyen información sobre los nodos en la topología, módulos a instalarse y configuraciones.

3.4 Conclusiones

- La gestión de red tradicional se realiza mediante la intervención directa de actividad humana, en redes IoE la cantidad de dispositivos que conforman la red hacen que su gestión se vuelva casi imposible de realizar de forma manual, requiriendo que el administrador de red sea un experto en el tema y se permita manejar complejos y demandantes ambientes, es por esto por lo que surge la necesidad de automatizar las actividades para gestionar redes, facilitando la gestión, reduciendo errores humanos y mejorando la eficiencia de la red gestionada.
- Los protocolos de gestión comúnmente utilizados en la gestión de red tradicional como SNMP y CMIP no cumplen los requisitos de escalabilidad, vasta topología y cambios impredecibles en la gestión de redes IoT e IoE, por lo que surgen las herramientas de automatización que dan una mayor flexibilidad a la gestión de red al utilizar agentes o proxys que sigan las instrucciones especificadas en una estación de gestión principal para configurar los dispositivos de red de forma asíncrona permitiendo un trabajo en paralelo y soportando enormes cantidades de nodos gestionados de red.
- Para automatizar la gestión de red no solo existen las herramientas Ansible, Puppet, SaltStack y Chef, sino que también se han desarrollado entornos y plataformas utilizando el aprendizaje de máquina y la computación en la nube, de esta forma se captura el conocimiento de un administrador de red para enseñarle a la red a autogestionarse y tomar decisiones propias luego de seguir un proceso de entrenamiento para el aprendizaje.
- La naturaleza de las redes IoT e IoE implica componentes de red heterogéneos y con recursos limitados, como es el caso de los sensores, esto obliga a que se

potencie el desarrollo de protocolos, herramientas, plataformas y arquitecturas que consideren su aplicabilidad en estos dispositivos. Los entornos basados en la nube y plataformas que utilizan aprendizaje de máquina consideran el bajo consumo de potencia para aplicarse a estos dispositivos mediante el protocolo LWM2M.

- Las herramientas de automatización de gestión de red analizadas en el presente trabajo de integración curricular tuvieron sus inicios en diferentes períodos de tiempo por lo que cada una tiene un esquema de trabajo distinto y utiliza métodos de comunicación diferente para alcanzar el mismo objetivo, el cual es automatizar la gestión de red, cada herramienta hace uso de un lenguaje de scripting y demanda distintos grados de conocimiento por parte del administrador de red, por lo que escoger una herramienta dependerá de las condiciones del entorno de aplicación, las capacidades del equipo de trabajo, el alcance de la red y sus objetivos.

  La comparativa realizada no señala la superioridad de una herramienta respecto a otra, solo destaca las características y funcionamiento para lograr un entendimiento completo y saber elegir de acuerdo con los requerimientos que se planteen. Si bien Ansible muestra una tendencia de uso mayor, se debe a que utiliza el lenguaje Python para la configuración de dispositivos, siendo este lenguaje, el más sencillo y popular actualmente.

- Dentro de todas las herramientas analizadas se destaca que Chef, SaltStack y Puppet fueron desarrolladas para trabajar con una arquitectura cliente-servidor en la que el cliente y el servidor se comunican mediante agentes instalados en los dispositivos gestionados con el fin de obtener las configuraciones almacenadas en el servidor y gestionar así los nodos de la red, sin embargo, esto representa un inconveniente para los dispositivos gestionados que no admiten la carga de sistemas operativos ni instalación de agentes, por lo que estas herramientas cuentan con versiones que permiten el uso de proxys para la comunicación con el dispositivo gestionado.

  En el caso de Ansible es una herramienta que utiliza el protocolo SSH para comunicarse con los dispositivos gestionados y SaltStack es una herramienta híbrida desarrollada para trabajar con y sin agente.

3.5 Recomendaciones

- Para realizar un análisis comparativo adecuado es necesario listar las mismas características e información de los elementos a compararse, en este caso las herramientas de automatización de gestión de red, con el fin de tener un punto de comparación adecuado que permita evidenciar las diferencias y semejanzas de cada una permitiendo al lector informarse y conocer qué herramienta elegir para un caso de aplicación en particular.
- Las arquitecturas para automatización de gestión de red se encuentran en continuo desarrollo e investigación, como se ha analizado en los textos citados en el presente trabajo de integración curricular, la implementación de un estándar facilitaría la implementación de la automatización de redes IoT e IoE.
- Para un entendimiento profundo de los sistemas y actividades propuestas en los casos de estudio-escenarios de implementación se recomienda visitar la fuente citada pues en la misma se detalla el modelado, implementación, códigos experimentos y resultados realizados con cada caso.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] M. Aboubakarab, M. Kellila y P. Rouxa, «A review of IoT network management: Current status and perspectives,» *Journal of King Saud University - Computer and Information Sciences,* pp. 3-4, 2021, doi: 10.1016/j.jksuci.2021.03.006.

[2] CISCO, «Cisco - Networking, Cloud, and Cybersecurity Solutions: The Internet of Everything,» [En línea]. Available: https://www.cisco.com/c/dam/en_us/about/business-insights/docs/ioe-value-at-stake-public-sector-analysis-faq.pdf.

[3] D. Vaya y T. Hadpawat, «Internet of Everything (IoE): A New Era of IoT,» *ICCCE 2019. Lecture Notes in Electrical Engineering,* Vols. %1 de %2570. Springer, Singapore, pp. 1-6, 2020, doi: 10.1007/978-981-13-8715-9_1.

[4] Qualcomm, «TiECon 2014 Summary-Part 1: Qualcomm Keynote & IoT Track Overview - Technology Blog,» IEEE ComSoc Technology Blog, 2014. [En línea]. Available: https://techblog.comsoc.org/2014/05/23/tiecon-2014-summary-part-1-qualcomm-keynote-iot-track-overview/.

[5] F. Hussain, «Internet of Everything,» *Internet of Things. SpringerBriefs in Electrical and Computer Engineering,* pp. 1-11, 2017, doi: 10.1007/978-3-319-55405-1_1.

[6] CISCO, «¿Qué es la automatización de la red?,» CISCO, 2021. [En línea]. Available: https://www.cisco.com/c/es_mx/solutions/automation/network-automation.html#~%C2%BFPor%20qu%C3%A9%20Cisco?.

[7] I. Peksens y V. Zagursky, «Computer network and system remote management,» *Proceedings AUTOTESTCON 2003. IEEE Systems Readiness Technology Conference,* pp. 535-540, 2003, doi: 10.1109/AUTEST.2003.1243628.

[8] S. T. Arzo, C. Naiga, F. Granelli, R. Bassoli, M. Devetsikiotis y F. H. Fitzek, «A Theoretical Discussion and Survey of Network Automation for IoT: Challenges and Opportunity,» *IEEE Internet of Things Journal,* vol. 8, nº 15, pp. 12021-12045, 2021, doi: 10.1109/JIOT.2021.3075901.

[9] Q. Qi, W. Wang, X. Gong y X. Que, «A SDN-based network virtualization architecture with autonomie management,» *2014 IEEE Globecom Workshops (GC Wkshps),* pp. 178-182, 2014, doi: 10.1109/GLOCOMW.2014.7063427..

[10] S. Ayoubi, N. Limam, M. A. Salahuddin, N. Shahriar, R. Boutaba, F. Estrada Solano y O. M. Caicedo, «Machine Learning for Cognitive Network Management,» *IEEE Communications Magazine,* vol. 56, pp. 158-165, 2018, doi: 10.1109/MCOM.2018.1700560.

[11] R. González Duque, «Python para todos,» 2008. [En línea]. Available: https://repositorio.uci.cu/jspui/bitstream/123456789/10206/1/Python_para_todos.pdf.

[12] M. Behringer, A. Dutta, Y. Hertoghs y S. Bjarnason, «Autonomic Networking - from theory to practice,» *2014 IEEE Network Operations and Management Symposium (NOMS),* pp. 1-17, 2014, doi: 10.1109/NOMS.2014.6838294.

[13] R. Chaparadza, T. B. Meriem, B. Radier, S. Szott, M. Wodczak, A. Prakash, J. Ding, S. Soulhi y A. Mihailovic, «SDN enablers in the ETSI AFI GANA Reference Model for Autonomic Management & Control (emerging standard), and Virtualization impact,» *2013 IEEE Globecom Workshops (GC Wkshps),* pp. 818-823, 2013, doi: 10.1109/GLOCOMW.2013.6825090.

[14] IETF, «Automated Network Management.,» IETF, [En línea]. Available: https://www.ietf.org/topics/netmgmt/.

[15] R. Casellas, R. Vilalta, A. Mayoral, R. Martínez, R. Muñoz y L. M. Contreras, «Control Plane Architectures Enabling Transport Network Adaptive and Autonomic Operation,» *2017 19th International Conference on Transparent Optical Networks (ICTON),* pp. 1-4, 2017, doi: 10.1109/ICTON.2017.8025013.

[16] Y. Lee, R. Vilalta, R. Casellas, R. Martínez y R. Muñoz, «Scalable telemetry and network autonomics in ACTN SDN controller hierarchy,» *2017 19th International Conference on Transparent Optical Networks (ICTON),* pp. 1-4, 2017, doi: 10.1109/ICTON.2017.8025036.

[17] F. Morales, «Conozca 3 tipos de investigación: Descriptiva, Exploratoria y Explicativa,» vol. 11, p. 2018, 2012.

[18] R. S. Carvajal, «TÉCNICAS DE ANÁLISIS DE INFORMACIÓN,» 2016. [En línea]. Available: https://administracionpublicauba.files.wordpress.com/2016/03/tecnicas-de-anc3a1lisis-de-informacic3b3n.pdf.

[19] P. Gargano., «Network Automation,» de *31 Days Before Your CCNP and CCIE Enterprise Core Exam*, Cisco Press, 2020, pp. 599-615. ISBN: 978-0-13-696522-0.

[20] A. Ratan, «Chapter 5: Ansible for Network Automation,» de *Practical Network Automation*, Birmingham, Packt Publishing, 2017 , pp. 143-159; ISBN 978-1-78829-946-6.

[21] JavaTpoint, «Ansible Workflow,» JavaTpoint, 2011. [En línea]. Available: https://www.javatpoint.com/ansible-workflow.

[22] E. Chou, «The Python Automation Framework – Ansible Basics,» de *Mastering Python Networking*, Birmingham, Packt Publishing Ltd., 2020 , pp. 131-134; ISBN 978-1-83921-467-7.

[23] M. Taylor, «Tips on managing IoT devices at the edge with Red Hat Ansible Automation,» 19 Marzo 2021. [En línea]. Available: https://www.redhat.com/es/blog/tips-managing-iot-devices-edge-red-hat-ansible-automation.

[24] J. Rhett, «Part I. Controlling with Puppet Apply,» de *Learning Puppet 4 A Guide to Configuration Management and Automation*, USA, O'Reilly Media, Inc., 2016, pp. 3-6; ISBN: 978-1-491-90766-5.

[25] S. Wågbrant y V. Dahlén Radic, «AUTOMATED NETWORK CONFIGURATION A COMPARISON BETWEEN ANSIBLE, PUPPET, AND SALTSTACK FOR NETWORK CONFIGURATION,» 6 Junio 2022. [En línea]. Available: https://www.diva-portal.org/smash/get/diva2:1667034/FULLTEXT01.pdf.

[26] A. Safonov, «CHEF VS PUPPET VS ANSIBLE VS SALTSTACK: WHAT IS BETTER FOR 2022,» MEREHEAD, 13 Diciembre 2021. [En línea]. Available: https://merehead.com/blog/chef-vs-puppet-vs-ansible-vs-saltstack-better-2022/.

[27] M. Taylor y S. Vargo, «Configuration Management and Chef,» de *Learning Chef*, USA, O'Reilly Media, Inc.,, 2015, pp. 1-6; ISBN: 978-1-491-94493-6.

[28] N. Sabharwal y M. Wadhwa, Automation throuhg Chef Opscode, Apress, 2015; ISBN: 978-1-4302-6295-4.

[29] J. Mutai, «How To Install Chef Workstation on CentOS 8 / RHEL 8,» Computing for Geeks, 25 Julio 2020. [En línea]. Available: https://computingforgeeks.com/how-to-install-chef-workstation-on-centos-rhel/.

[30] C. Sebenik y T. Hatch, Salt Essentials, USA: O'Reilly Media, Inc.,, 2015; 978-1-491-90063-5.

[31] A. Mallett, Salt Open, Peterborough: Apress, 2021; /doi.org/10.1007/978-1-4842-7237-4.

[32] V. Boon, «Ansible & Salt,» 24 Julio 2018. [En línea]. Available: https://www.slideshare.net/mynog/ansible-salt-vincent-boon.

[33] K. Hitchcock, «Chapter 5: Automation,» de *Linux System Administration for the 2020s*, Apress, 2022, doi: https://doi.org/10.1007/978-1-4842-7984-7_1, pp. 110-119.

[34] P. MIHĂILĂ, T. BĂLAN, R. CURPEN y F. SANDU, «Network Automation and Abstraction using Python Programming Methods,» de *MACRo 2017 - 6th International Conference on Recent Achievements in Mechatronics, Automation, Computer Science and Robotics*, Brasov, 2017, doi: 10.1515/macro-2017-0011.

[35] A. Ratan, «Chapter 2: Python for Network Engineers,» de *Practical Network Automation*, Birmingham, Packt Publishing, 2017, ISBN 978-1-78829-946-6, pp. 35-40.

[36] ECHELON, «Enns, R., Bjorklund, M., Schoenwaelder, J., Bierman, A., 2011. Network configuration protocol (netconf),» Izot platform, 2018. [En línea]. Available: https://www.echelon.com/izot-platform..

[37] S. Sohail, «Automation of Network Management with Multidisciplinary Concepts,» *Int. J. Comp. Tech. Appl,* vol. 1, nº 1, pp. 71-77, 2014.

[38] Python Software Foundation; JetBrains, «Python Developers Survey 2018 Results,» Python, JetBrains, 2018. [En línea]. Available: https://www.jetbrains.com/research/python-developers-survey-2018/.

[39] M. L. Peña, J. Díaz, J. Pérez y H. Humanes, «"DevOps for IoT Systems: Fast and Continuous Monitoring Feedback of System Availability,» *IEEE Internet of Things Journal,* vol. 7, nº 10, pp. 10695-10707, Oct. 2020, doi: 10.1109/JIOT.2020.3012763..

[40] M. Centurión, M. Kotvinsky, D. Calegari y A. Delgado, «A Model-Driven Approach for System Administration,» 2019. [En línea]. Available: http://ceur-ws.org/Vol-2442/paper2.pdf.

[41] L. Carvajal, Metodología de la Investgación Científica. Curso general y aplicado, 28 ed., Santiago de Cali: U.S.C., 2006, p. 139.