Informática
XVI CONVENCIÓN Y FERIA INTERNACIONAL 2016
16TH INTERNATIONAL CONVENTION AND FAIR
Conectando sociedades
Connecting societies
La Habana, Cuba
Del 14 al 18 de marzo
Havana, Cuba
March 14th to 18th

COMPARACIÓN DE HERRAMIENTAS DE GESTIÓN DE LA CONFIGURACIÓN

COMPARISON OF CONFIGURATION MANAGEMENT TOOLS

Enrique Carbonell Muela[1], *Ana María García Pérez*[2]

1 DATYS, Cuba, enrique.carbonel@datys.cu, Máximo Gómez #65 Norte e/Tirso Marín y Calderón, Sancti Spíritus, Sancti Spíritus, Cuba, CP 10600

2 DESOFT, Cuba, anamaria.garcia@vcl.desoft.cu

RESUMEN: *Las herramientas de gestión de la configuración (CMT) en esencia permiten* la *automatización de la instalación, configuración y actualización de software en un sistema. La variedad de estas herramientas disponibles en el mercado hace difícil la selección de la adecuada, acorde a las necesidades de los clientes cuando no se cuenta con expertos en el área para consultar. El propósito de este trabajo es seleccionar las características para establecer un marco de comparación y aplicarlo a cuatro de las CMT más populares. Para la selección se tuvieron en cuenta algunos criterios de comparación de estudios previos que de conjunto con los aportados en este trabajo dan lugar a un sencillo marco de comparación para las CMT. Las herramientas seleccionadas por su popularidad fueron: CFEngine, Puppet, Ansible y Salt. El estudio aporta una tabla resumen donde se evidencian las características seleccionadas y su evaluación en cada una de las herramientas.*

Palabras Claves: herramientas de gestión de la configuración; *marco de comparación*; CFEngine; Puppet; Ansible; Salt

ABSTRACT: *The configuration management tools (CMT) essentially allow automation of installation, configuration and update of system's software. The variety of these tools available in the market makes it difficult the right selection according to the needs of clients when you do not have consultant experts. The purpose of this work is select the features to build a comparative framework and apply it to four of the most popular CMT. In order to select some CMT we take into account some benchmarks of previous studies and join with others proposed here to obtain a simple comparative framework. The tools selected by popularity went: CFEngine, Puppet, Ansible and Salt. This work provides a summary table where the selected features are evaluated in each of analyzed tools.*

KeyWords: configuration management tools; comparison framework; CFEngine; Puppet; Ansible; Salt

1. INTRODUCCIÓN

Cada día las personas se relacionan con aplicaciones de software, tanto en el ámbito personal como son las redes sociales para comunicarnos con amistades, así como en el ámbito profesional con servicios para la gestión de los clientes de una organización. Estas aplicaciones pueden estar alojadas en host físicos, entornos virtualizados o incluso en la nube y a su vez estar interconectadas para crear sistemas distribuidos de gran escala. Para que funcione, el software necesita ser puesto en marcha. Esto significa instalar, configurar y mantener los recursos necesarios para su explotación en producción.

Mientras estas acciones se ejecuten manualmente, siempre se corre el riesgo de generar complejidad innecesaria en el sistema, haciendo el mantenimiento difícil y costoso. Siendo agravado cuando se necesita escalar los servicios o hay que mantener diferentes ecosistemas para el desarrollo y pruebas de software. Desafortunadamente las aplicaciones de software suelen fallar y el costo del tiempo fuera de servicio aumenta con cada segundo. Estos errores frecuentemente son causados por problemas de configuración y cuando se manejan sistemas distribuidos donde existen dependencias entre los servicios, la probabilidad de error aumenta drásticamente.

La configuración de las aplicaciones, está formada por diversos parámetros que deben ser consistentes en toda la configuración: desde los parámetros de la aplicación final de usuarios hasta los de configuración de red de la infraestructura. Asimismo debe garantizarse que luego de cada actualización de las configuraciones, estos parámetros queden consistentes en aras de evitar futuras fallas en el funcionamiento de las aplicaciones.

Las herramientas de gestión de la configuración (CMT) como Puppet y Chef fueron diseñadas para resolver estas tareas de forma consistente y confiable, haciendo el software fácil de configurar e instalar, preparándolo para su modificación o expansión.

Estas herramientas simplifican las tareas de gestión de grandes y complejos despliegues manteniendo los sistemas actualizados, reduciendo los costos de operación. Su objetivo fundamental es la automatización de la instalación, configuración, actualización de software en un entorno dado. Adicionalmente, pueden mantener un seguimiento del estado y brindar una descripción del software instalado.

Las CMT emplean las especificaciones de entrada para modelar el estado deseado de la configuración, incluyendo la interdependencia entre los parámetros de configuración.

Algunos trabajos se han realizado en el área de las pruebas de las especificaciones de entrada para garantizar la consistencia de la configuración [1]-[4].

En la actualidad existen en el mercado varios sistemas reconocidos que se pueden utilizar, sin embargo como toda aplicación de software, la adopción de uno específico implica una inversión grande de tiempo y dinero. Por lo que hacer una elección bien fundamentada sobre la base de criterios objetivos es lo mejor para asegurarse que la empresa va a adquirir la herramienta justa para su entorno.

Pero, ¿cuáles pudieran ser estos criterios objetivos en ausencia de expertos en el tema?

El objetivo de este trabajo consiste en seleccionar esos criterios estableciendo un marco de comparación y aplicarlo a cuatro de las CMT más referenciados y populares [5]-[9]: CFEngine, Puppet, Ansible y Salt.

El marco propuesto es un ajuste y actualización de varios estudios previos sobre el tema.

El resultado del trabajo es útil para los profesionales en la selección rápida de una CMT y dando a conocer las características actualizadas de cada una de ellas, sobre todo en el contexto cubano con una limitada adopción de este tipo de herramientas, aun cuando constituyen una mejora sobre la continua explotación del scripting ad-hoc que se ha empleado en las últimas dos décadas.

2. CONTENIDO

En los últimos tiempos con la popularidad de las redes sociales, blog y demás plataformas de comunicación, se suele realizar una búsqueda rápida de los pros y contras para evaluar un producto de software. Sin embargo estos resultados suelen estar influenciados por los requerimientos y experiencias individuales de los redactores quienes, en determinados momentos pueden llegar a contradecirse entre sí. Por lo que no es recomendable tomar decisiones definitivas basadas en estos resultados parciales.

Se hace necesario establecer las características a considerar y cómo evaluarlas. En el ámbito de los proyectos de código abierto podemos encontrarnos con varios trabajos que proponen y analizan metodologías de evaluación para estos tipos de

proyectos [10][11] Pero como ningún método es perfecto, frecuentemente sus criterios están siendo revisados y actualizados.

Para la selección de los criterios específicos del campo de aplicación tratado, se tuvieron en cuenta en primera instancia estudios comparativos previos [12], trabajos que introducen y caracterizan diferentes alternativas de CMS [8][13], presentaciones en eventos, documentaciones oficiales de los productos, criterios de la comunidad (blog, IRC, redes sociales, etc.), entre otras fuentes disponibles de los últimos cinco años. De igual forma se tuvo en cuenta que en Cuba no se cuenta con grandes compañías que empleen estos sistemas, por lo que el reducido nivel de expertos en el área llevó a descartar cualquier criterio previamente empleado que dependa de la validación por el criterio de expertos.

2.1 Criterios de comparación

Propiedades de la especificación

Algunos trabajos basados en la opinión de la comunidad plantean como elementos esenciales a tener en consideración el lenguaje para usar el software [14] y si este es declarativo o imperativo [15]. Adicionalmente se tiene en cuenta si la herramienta brinda una interfaz de usuario para la gestión de los recursos.

Un elemento a considerar es si la herramienta provee de un mecanismo de modularización que permita la reutilización del código escrito por los desarrolladores y que este pueda ser empleado en futuras tareas (por ejemplo: las recetas de Chef o los roles de Ansible) y si cuenta con un repositorio de estos elementos para socializar el conocimiento (como Galaxy en Ansible).

Propiedades del despliegue

Delaet, Joosen y Vanbrabant [15] se refieren a la escalabilidad como la habilidad de la CMT de adaptarse a los cambios en el crecimiento de la infraestructura. La herramienta debe ser capaz de proveer la especificación de configuración necesaria para estos grandes y complejos entornos.

Aunque la mayoría de las herramientas analizadas en el estudio comparten elementos de la arquitectura de despliegues típica Figura 1, consideramos necesario incluir este elemento en la comparación, analizando: la arquitectura de los agentes de traducción y la tecnología empleada para distribuir las especificaciones (push o pull).

Es común encontrar variedad de plataformas en las infraestructuras actuales: Windows/Unix/Mac OS X servers; de igual forma podemos hallar la necesidad de gestionar diversos dispositivos: servidores, teléfonos inteligentes, equipamiento de red, etc. Es por ello que considerar las plataformas y dispositivos soportados aportan elementos para evaluar cuán flexible es la CMT y cuántos sistemas diferentes se pueden gestionar con el software [14][15].

Propiedades de gestión

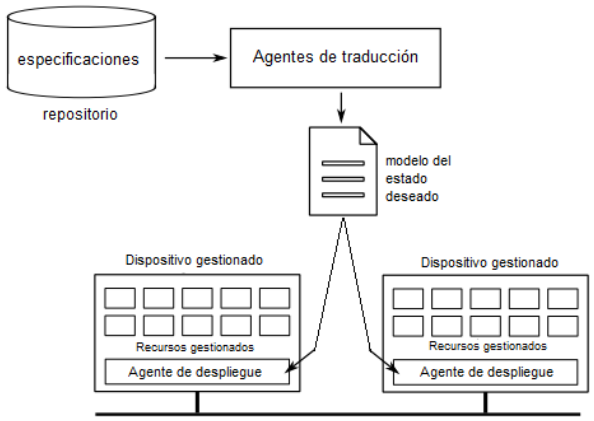

Figura 1 Arquitectura general de los CMT.

De los elementos para valorar la usabilidad solamente se tienen en cuenta: si la herramienta brinda soporte para probar las especificaciones introducidas y si la herramienta provee algún sistema o si cuenta con algún mecanismo para monitorear el estado de la infraestructura actual.

La integración con el entorno ilustra la habilidad de la herramienta de permitir a la especificación creada utilizar información proveniente del entorno de trabajo, por ejemplo adquirir los usuarios a partir de un servidor LDAP.

Durante el empleo de estas herramientas suelen presentarse conflictos en las definiciones como: conflictos específicos de aplicación (ejemplo: cuando se declaran varios servicios con el mismo puerto en un host), conflictos de modalidad que son contradicciones en la especificación de la configuración.

El versionado de las especificaciones es un elemento deseable en los sistemas que trabajan la infraestructura como código, mediante el empleo de sistemas de control de versiones (CVS). Estos posibilitan el restablecimiento de los sistemas ante fallos, así como el seguimiento de la evolución de las especificaciones.

Las especificaciones para la configuración de las infraestructuras usualmente son desarrolladas por diferentes personas, por lo que es necesario tener un control de acceso a los recursos que cada cual puede configurar. Por esto se analizó la presencia de mecanismos de autenticación y autorización.

Soporte

Una característica del soporte es la documentación

disponible y dentro de esta, la documentación de instalación; evaluando los prerrequisitos o cantidad de elementos necesario a instalar para que funcione, así como las formas de instalación pueden dar una idea de cuán rápido pudiera ser la puesta en marcha del nuevo sistema.

Un elemento deseable para viabilizar la evaluación de un producto de software es que se provean recursos para el mismo de forma tal que se minimicen los esfuerzos de instalación y configuración que en etapas de exploración pudiesen incurrir en gasto de tiempo por la falta de familiarización con los productos, para ello es frecuente encontrar demos online, recursos de Vagrant, recursos para un despliegue limitado en un proveedor de servicios en la nube o máquinas virtuales con los componentes instalados para probar localmente.

Constar con una versión comercial del producto usualmente brinda mayor confianza a la hora seleccionar una alternativa, pues generalmente estas vienen asociadas con un alto compromiso de soporte técnico por parte de los creadores y funcionalidades que no podemos encontrarnos en las versiones de la comunidad. Adicionamos entonces al análisis la presencia de versiones comerciales del producto y sus funcionalidades adicionales en comparación con la de la comunidad.

La estabilidad de un proyecto depende de muchos factores difíciles de medir cuando estamos iniciándonos en un producto; sin embargo podemos investigar sobre su política y frecuencia de liberación para tener una idea al respecto.

Comunidad

Un elemento importante que buscamos en productos de software es la comunidad que tiene un proyecto. En presencia de una comunidad grande es fácil resolver problemas, encontrar las mejores prácticas, casos de uso exitoso de las herramientas, etc. Por lo que se adicionó la evaluación de la comunidad teniendo en consideración: la cantidad de usuarios contribuyentes del proyecto, la cantidad de proyectos relacionados y los canales de comunicación (blog, redes sociales, salas de chat, foros) donde usualmente se publica información al respecto.

Actualización

Para medir la actividad de actualización se tuvo en cuenta: el tiempo transcurrido desde la última actualización del producto y la cantidad de issues resueltos en el primer semestre del año.

Popularidad

Usualmente cuando evaluamos un producto de software buscamos la cantidad de usuarios y cuáles son las compañías de referencia que emplean el producto. Estos elementos muestran su generalización en las diferentes organizaciones y si sirve para grandes escenarios de despliegue [14][16]. No obstante es difícil medir exactamente el número de usuarios y despliegues, por lo que se emplean otros criterios aproximados como el número de usuarios en la lista de distribución y las organizaciones que emplean el producto. Adicionalmente se analizó la presencia en las redes sociales teniendo en cuenta su actividad.

Licencias

Aunque todos los CMT que se evalúan en este trabajo tienen una distribución de código abierto, es interesante conocer bajo qué licencia específica se distribuyen: MIT, GPL, LGPL, etc., criterio que ha sido empleado por Torberntsson y Rydin [14].

Precio

El precio constituye un factor determinante en todo producto a adquirir que usualmente varía en dependencia de la cantidad de nodos a gestionar y el soporte contratado, por lo que se tuvo en cuenta el precio anual más bajo para 100 instancias con la versión Empresarial, similar al trabajo de Torberntsson y Rydin [14].

Lenguaje de desarrollo

El lenguaje de desarrollo de un software puede afectar elementos como la comunidad y el desempeño. Es común considerar que un software desarrollado en un lenguaje muy popular tenga mayor cantidad de contribuidores y seguidores. De igual forma sucede con un software desarrollado en C, debe tener mejor rendimiento que otro desarrollado en un lenguaje a más alto nivel. Por estas razones se agregó al estudio el lenguaje con que es desarrollada la CMT.

A modo de resumen la Figura 1 muestra las características que fueron seleccionadas para conformar el marco de comparación.

Requerimientos de hardware

Para tener una idea de las prestaciones básicas necesarias en la explotación de las CMT se incluyeron los requerimientos de hardware mínimos para el despliegue más básico de la distribución empresarial.

2.2 Evaluaciones

2.2.1 CFEngine

CFEngine es una CMT que permite gestionar de forma segura las infraestructuras de IT. Los escenarios de usos son disimiles entre los que podemos encontrar: verificación del contenido y la integridad de archivos, despliegue de aplicaciones y actualizaciones de sistemas.

Su Design Center (DC) es ofrecido como el repositorio público donde los miembros de su comunidad pueden compartir código y patrones de diseño. En la versión Empresarial se cuenta con el Design Center UI que permite a los ingenieros de infraestructura autorizados configurar, desplegar y monitorear plantillas de políticas. El DC se encuentra público en Github pero según las estadísticas del repositorio no cuenta con mucha actividad y en los últimos 6 meses ha estado prácticamente inerte.

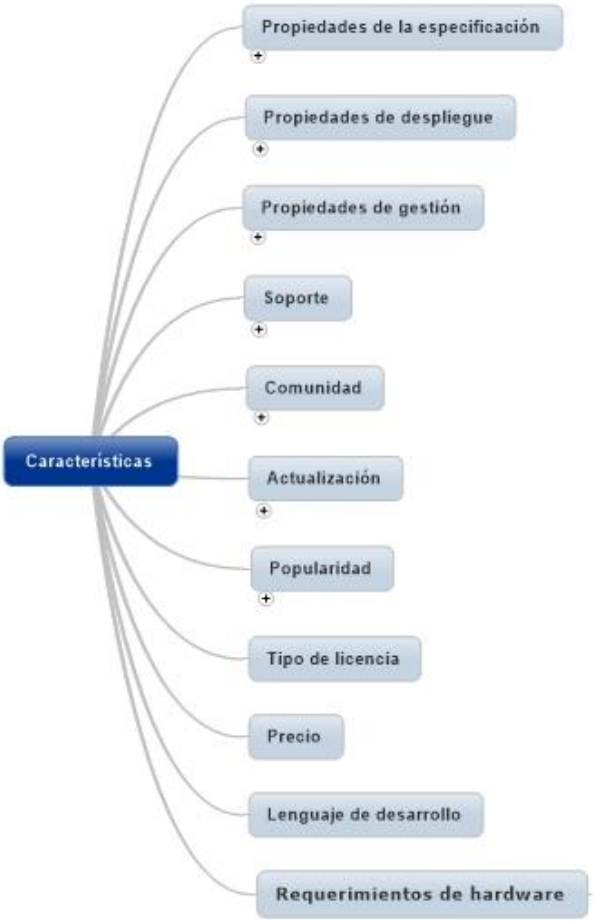

Figura 1: Criterios aplicados para el análisis de CMT.

De forma general si está preparado para grandes despliegues incluso manteniendo la alta disponibilidad de servicio mediante un pull de servidores de políticas (configuraciones).

La arquitectura empleada es distribuida con el énfasis en los agentes que se ejecutan en cada host gestionado. El servidor central solo se emplea para coordinar la distribución de las políticas.

Los clientes de CFEngine pueden trabajar como clientes o como una estructura multinodo. Cuando se opera como una estructura multinodo, algunos clientes pueden ser configurados como servidor de políticas y otros como simples clientes. Los clientes contactan al servidor de políticas que son los responsables de la distribución de las políticas. Una política no es más que un conjunto de promesas que a su vez son empleadas para describir el estado deseado de un sistema.

Cuando el servidor de políticas ha publicado una promesa los clientes pueden actualizarse automáticamente. Cada nodo en una red de CFEngine son actores independientes y trabajan de forma oportunista que significa que incluso si un servidor de políticas desaparece, se mantendrá en funcionamiento haciendo lo que puede con la información que tiene almacenada.

No está documentado el uso de herramientas complementarias para la ejecución de pruebas de las especificaciones, solo aparece una referencia sobre la necesidad de hacer test de las políticas antes de liberarlas en producción. No obstante como parte del mecanismo de distribución de los estados deseados, los agentes locales siempre ejecutan una comprobación de sintaxis antes de aplicar los cambios [17].
Soporta la gestión de conflictos detectando conflictos de modalidad, como una configuración inestable que no converge [12][18].

CFEngine cuenta dos versiones del producto: la Empresarial recomendada para producción y la de la Comunidad que fue la pionera en 1993.

Valoramos la actividad del proyecto cfengine/core analizando los 12K commits, 210 releases y la cantidad de issues resueltos cuya mayor actividad se enmarca entre 2013 y finales del 2014, por lo que se valoró como activo.

Su presencia en las redes sociales se mantiene activa, con actividad en: Google+ con casi 100 seguidores y más de 13K visitas, junto a su canal de YouTube que desde 2010 ha compartido más de 100 videos para sus 300 suscriptores. En el caso de LinkedIn se puede encontrar junto a su sitio oficial varios grupos que garantizan la constante actualización de temáticas relacionadas con sus funcionalidades y mejores prácticas. De igual forma la actividad se mantiene en el caso de Twitter con valores que superan los 2K tweet y 1K seguidores. Por estas razones de clasifico como activo.

2.2.2 Puppet

Puppet Labs es uno de los líderes en la automatización de IT desde sus inicios en 2005, facilitando la automatización y gestión de los host y sus softwares tanto en máquinas físicas como virtuales en servidores dedicados o integrándose con servicios en la nube. Puppet aparece como proyecto de código abierto y a partir de este se distribuye Puppet Enterprise (PE). PE es una extensión de Puppet que incluye soporte profesional, un stack de Puppet listo para

producción, una consola web para analizar reportes y controlar las infraestructuras, funcionalidades de orquestación y herramientas para provisionar en la nube.

El ecosistema de Puppet incluye decenas de proyectos de código abierto que se emplean tanto en la versión de la Comunidad como en la Empresarial. Dentro de estos podemos encontrar Geppeto que es un IDE para facilitar el desarrollo en Puppet. Adicionalmente provee una interfaz: Forge para crear proyectos desde módulos preexistentes y facilitar su socialización en la comunidad.

En la versión PE se incluye una interfaz web llamada Console que permite analizar los elementos de la infraestructura de forma centralizada, ofreciendo una forma sencilla para desplegar configuraciones y aplicaciones a los servidores y dispositivos de red.
Adicionalmente Puppet cuenta con un repositorio de módulos escritos por la comunidad llamado Forge. Al publicarse un módulo, este puede ser revisado y clasificación según su popularidad, originalidad entre otros elementos aportados por la comunidad. Como resultado el modulo se etiqueta estableciendo su estabilidad y confiabilidad para el empleo en producción.

Puppet está diseñado para funcionar en una arquitectura cliente/servidor o de forma independiente. En la primera todos los ficheros de configuración con las "recetas" escritas en el lenguaje declarativo son almacenados en el nodo servidor denominado "Puppet Master". Este nodo central se encarga de muchas de las tareas como analizar las recetas y compilarlas para generar catálogos. Estos catálogos son conjuntos de ficheros XML que serán recogidos por los clientes o agentes. Los clientes solo se encargan de implementar la funcionalidad requerida al comparar los catálogos almacenados en su cache local con los del servidor y finalmente reporta al servidor los cambios realizados. El Master también puede enviar notificaciones a los clientes en caso de que los ficheros de configuración hayan cambiado para hacer que el cliente obtenga los nuevos catálogos. En la arquitectura independiente los nodos administrados contienen la copia completa de su información de configuración y compilan su propio catalogo para luego en demanda o por un proceso planificado aplicarlos.

Trabaja con gran variedad de plataformas, anunciando para próximas entregas la ampliación del soporte de los agentes en Mac OS, AIX, Solaris, etc.

Se emplea para gestionar servidores pero es capaz de trabajar sobre cualquier sistema que soporte la instalación de los agentes. En las recientes versiones ya se soporta el sistema operativo Cisco NX-OS como plataforma para correr los agentes en los switches de red. Inclusive algunos socios como Huawei ya han incluido nativamente el soporte de los agentes en sus dispositivos de red.
Puppet fue de los pioneros en facilitar el trabajo de la Infraestructura como Código (IaC) y tiene experiencia aplicando test sobre las infraestructuras [7]. Por ejemplo para que un módulo de Forge esté listo para producción es requerido su verificación en PE asegurando el estilo de la sintaxis con Puppet-lint, pasar los test unitarios con rspec-puppet y realizar test de aceptación con Beaker framework. Considerando estos elementos junto a la facilidad para trabajar varios entornos podemos decir que tiene soporte para test de especificaciones.

Para la integración con un CVS hay algunas referencias donde abordan como emplear Git para trabajar con Puppet, incluso el IDE Geppeto permite trabajar con los módulos de la comunidad e integrarlos con Git para su provisionamiento en PE.

Para valorar actividad del proyecto puppetlabs/Puppet se contó con los 20K commits, 301 releases y la cantidad de issues resueltos que se ha mantenido bastante alta desde finales del 2011, por lo que es un proyecto muy activo.
Mantiene cuentas en las redes sociales como Google+ con más de 2K seguidores y 60K visitas, junto a su canal de YouTube creado en el 2011 con +500 videos compartidos para más de 7K suscriptores. En el caso de LinkedIn se puede encontrar junto a su sitio oficial varios grupos en los que se comparten experiencias de desarrollo y administración. Esta actividad se mantiene en Twitter con valores que superan los 8K tweet y 53.6K seguidores. Por estas razones se clasificó como muy activo.

2.2.3 Ansible

Ansible es una de las más recientes herramientas de automatización de IT, dentro de sus uso podemos encontrar la automatización del aprovisionamiento en la nube, la gestión de la configuración, el despliegue de aplicaciones y la orquestación de servicios y sus interdependencias.

Ansible Tower es su distribución comercial recomendada para el contexto empresarial. Es una interfaz web que permite centralizar y controlar las infraestructuras incluyendo: un dashboard, control de acceso basado en roles, planificación de trabajos y la gestión visual de los inventarios; simplificando la creación de nuevos entornos, permitiendo a desarrolladores establecer fácilmente sus entornos para QA y Test amend de la intervención del personal de Ops. Incluso el propio personal comercial podría establecer en demanda entornos para demos.

Cuenta con roles y playbook. Los roles son la agrupación de un conjunto de tareas, variables, plantillas para lograr un fin específico y los playbook son los que permiten representar el conjunto de pasos o roles a seguir/aplicar en la infraestructura gestionada con Ansible.

Los roles en Ansible son construidos en aras de incluir ficheros y combinarlos de forma clara, reusable que nos permitan abstraernos y enfocarnos en la idea general y solo profundizar en los detalles de implementación cuando sea necesario.

Los roles pueden ser compartidos a la comunidad mediante Galaxy que es el sitio web para compartir roles o herramientas de línea de comandos creadas para ayudar a trabajar con los roles.

Ansible trabaja sin contar con un agente instalado en cada host, evitando la necesidad de mantener actualizado estos agentes y pudiendo resolver el impedimento de gestionar host donde no existen los agentes instalados. Ansible es descentralizado delegando el control de acceso a las credenciales del OS a administrar o integrándose a sistemas de gestión centralizada de usuarios.

Su versatilidad le permite su aplicación en gran variedad de dispositivos, incluso dentro de sus socios hay proveedores de dispositivos de red como CISCO y Juniper Networks.

Ansible está diseñado para ser fail-fast, sin embargo se plantea que existen diferentes formas de plantear test con su sintaxis. Aplicando estos mecanismos de pruebas en las diferentes etapas de desarrollo local, QA y producción podemos reducir los riesgos en los entornos reales de producción. Existen diferentes formas de trabajo para los test, pero la esencia es siempre ejecutar los mismos playbook en desarrollo, QA y producción.

Usualmente se liberan nuevas versiones cada dos meses. La aplicación principal se desarrolla conservadoramente valorando la simplicidad en el diseño del lenguaje y la configuración. Sin embargo, el desarrollo de la comunidad alrededor de nuevos módulos y plugins ocurre rápidamente, añadiendo 20 o más módulos en cada entrega.

Cuenta con un servicio de consultoría y entrenamiento, mediante el que especialistas pueden revisar los contenidos y configuraciones de Ansible y Tower instalados haciendo recomendaciones para futuros proyectos; ayudando en el proceso de migración desde otros CMT o soluciones locales. También cuenta con un sitio destinado al soporte en el que se organizan las respuestas a preguntas frecuentes.

Tower se ofrece en diferentes ediciones: Básica, Empresarial y Premium diferenciándose respecto al soporte en el tiempo de atención, las vías de comunicación y el tiempo de respuesta según el tipo de problema reportado.

La actividad del proyecto ansible/ansible se evidencia con los 16K commits, 81 releases y la cantidad de issues resueltos que se ha mantenido alta desde principios del 2013, por lo que se valora como muy activo.

Mantiene cuentas en las redes sociales como Google+ con más de 625 seguidores y más de 42K vistas, junto a su canal de YouTube con muy pocos videos compartidos para alrededor de 600 suscriptores. En el caso de LinkedIn se puede encontrar junto a su sitio oficial varios grupos en diferentes localidades. Pero donde se evidencia una gran actividad es en Twitter con cifras superiores a los 6K tweet y 12K seguidores. Por estas razones de clasificó como activo.

2.2.4 Salt

SaltStack es una de las herramientas para desplegar y orquestar infraestructuras de nube, optimizar la configuración de sistemas y controlar una vasta cantidad de entornos como: VMware, OpenStack, Google Compute Engine, AWS, Docker, etc.

SaltStack es comúnmente empleado para desplegar, gestionar y automatizar infraestructuras y aplicaciones para big data, IoT, storages dinámicos, redes definidas por software, servidores de seguridad, etc.

Ha sido acreedora de varios premios en los últimos años en competencias y eventos relacionados con la gestión de virtualización, entornos de nube y big data, siendo un referente en las mejores prácticas de DevOps y herramienta de gestión de la nube.

En la próxima versión empresarial han anunciado la incorporación de una GUI siguiendo los pasos de su competencia.

Para la modularización cuenta con módulos y fórmulas que permiten representar la configuración, despliegue y orquestación de los host. Adicionalmente mantiene público en github una colección de fórmulas y además cuenta con un gestor de fórmulas Salt Package Manager (SPM) que permite construir y desplegarlas de forma sencilla y análoga a los sistemas de empaquetado como Yum o RPM.

Una de sus ventajas es que soporta ambos mecanismos de distribución (push y pull) según sean las necesidades del cliente. Cuenta con diferentes componentes desde los que podemos ejecutar las acciones remotamente sobre los nodos administrados o distribuir los cambios mediante aplicaciones instaladas previamente en los nodos a gestionar.

Salt emplea dos enfoques de prueba: pruebas de integración y pruebas unitarias, para las que provee bibliotecas en aras de minimizar las pruebas manuales.
Salt permite trabajar por entornos de trabajo quedando agrupados sus archivos por estos entornos. Esta estructura de archivos puede integrarse con diferentes CVS para que los *minions* (agentes instalados en los host) los empleen como código fuente.
Cuenta con una detallada guía para probar las funcionalidades básicas de las herramientas de su arquitectura, incluyendo para ello un proyecto en github con el Vagrantfile necesario para probar el despliegue de un master y 2 *minions*.
La licencia de la versión Empresarial se basa en la cantidad de nodos y la suscripción del soporte en dependencia del entorno y la infraestructura. Brindando siempre un soporte exclusivo y priorizado con expertos. Adicionalmente cuenta con servicios de consulta y entrenamiento que pueden ser incluidos en el contrato de soporte e incluso un servicio de certificación.
Los requerimientos de hardware no aparecen oficialmente, no obstante se menciona que requiere pocos recursos y deben ser afinados en dependencia de la cantidad de nodos y tareas concurrentes que se quiera realizar.
La actividad del proyecto saltstack/salt se evidencia con más de 50K commits, 101 releases y la cantidad de issues resueltos desde los inicios que ha ido en ascenso, por lo que lo estimamos como muy activo.
Mantiene cuentas en las redes sociales como Google+ con más de 600 seguidores y más de 115K visitas, junto a su canal de YouTube con 82 videos compartidos para alrededor de 115K suscriptores. En el caso de LinkedIn se puede encontrar junto a su sitio oficial algunos grupos relacionados. La actividad en Twitter se mantiene activa con cifra superiores a los 2K tweet y 6K seguidores. Por estas razones se clasificó como muy activo.

Tabla I: Comparación de las CMT seleccionadas

			CFEngine	Puppet	Ansible	Salt
Propiedades de la especificación						
Tipo de lenguaje			declarativo	declarativo, imperativo	declarativo	declarativo
Interfaz de usuario			CLI, Web	CLI, Web	CLI, Web	CLI
Mecanismo de modularización			si/Design Center	si/Forge	si/Galaxy	si
Propiedades del despliegue						
Escalabilidad			medios-grandes	medios-grandes	pequeños-medios	pequeños-medios-grandes
Arquitectura de despliegue		Agentes de traducción	Gestión distribuida	Gestión centralizada	Gestión centralizada	Gestión centralizada
		Mecanismo de distribución	pull	pull	push	push, pull
Flexibilidad	Plataformas	Servidores	CentOS, Debian, RHEL, Ubuntu	Red Hat Enterprise Linux, CentOS, Oracle Linux, Scientific Linux, SUSE Linux Enterprise Server, Ubuntu	Plataformas con Python 2.6 o 2.7 (no Windows).	Plataformas Unix con Python 2.6 o superior e inferior a 3.0
		Clientes	AIX, CentOS, Debian, UP-UX, RHEL, SLES, Solaris, Ubuntu, Windows	Red Hat Enterprise Linux, CentOS, Oracle Linux, Scientific Linux, SUSE Linux Enterprise Server, Ubuntu, Debian, Windows	Plataformas con Python 2.4 o superior e inferiores a 3.0	plataformas Unix o Windows
	Dispositivos		host	host, dispositivos de red	host, dispositivos de red	host, dispositivos de red
Propiedades de gestión						
Usabilidad	Soporte para test de especificación		Manual, comprobación de sintaxis	Puppet-lint, rspec-puppet, Beaker framework	Manual, comprobación de sintaxis	Manual, componentes para test de integración y unitario
	Monitoreo de la infraestructura		si	si	si	si
Integración con el entorno			Características de los host administrados durante la ejecución			
Resolución de conflictos			Conflictos de modalidad	Conflictos de modalidad	NE	NE
Soporte para CVS			si/Repositorio externo			
Control de acceso			Autenticación integrable con recursos externos y Autorización			

		CFEngine	Puppet	Ansible	Salt
Soporte					
Documentación disponible		documentación, referencias, tutoriales, presentaciones, videos			
Documentación de instalación		web	web	web + pdf(para Tower)	Web + pdf
Formas de instalación		repo, script, manual o Vagrant	repo, script, manual	repo, manual, script wizard para Tower	repo, manual, script
Elementos para instalación		Instalador del servidor de políticas y el de los clientes.	Puppet Master, PuppetDB y PostgreSQL, Console, Agents	python-simplejson, Ansible Core, Ansible Tower, libselinux-python(si está habilitado SELinux)	msgpack-python, YAML, Jinja2, MarkupSafe, apache-libcloud, Requests, ZeroMQ o RAET
Recursos para evaluar producto		Hasta 25 nodos versión Empresarial con Vagrant	máquina virtual de prueba hasta 10 nodos, cursos online	Tower con Vagrant y recursos para AWS	Vagrant para prueba de un master con 2 nodos
Frecuencia/ Política de liberación		2 x año/LTS	NE	cada 2 meses	NE
Funcionalidades adicionales de la versión comercial		GUI con dashboards para monitoreo de estado de los host, alertas, inventario y planificación de tareas.			
Comunidad					
Contribuidores		73	398	1156	1248
Proyectos relacionados		<10	>2116	<15	<40
Actividad del proyecto		activo	muy activo	muy activo	muy activo
Canales de comunicación		Redes sociales, IRC, Newsletter, RSS, Youtube	Redes sociales, IRC, Newsletter, RSS, Youtube, grupos de google, Puppet Ask, Reddit,	Redes sociales, IRC, Newsletter, RSS, Youtube, grupos de google	Redes sociales, IRC, Newsletter, RSS, grupos de google, Youtube, Reddit
Actualización					
Tiempo desde la última entrega		<6 meses	<6 meses	<6 meses	<6 meses
Issues resueltos 1er semestre		211 de 238	729 de 763	926 de 1290	1441 de 2730
Año de creación		1993	2005	2012	2011
Popularidad					
Usuarios/Temas de la lista de distribución		712/6070	8181/14527	5234/7070	3234/5252
Empresas de referencia	Clientes	Intel, Linkedin, SAMSUNG,Chevron, Comcast, DIRECTV	SONY, redhat, Twitter, SUN Oracle, Motorola, WMWare, github, Intel, at&t, NESTLE	Evernote, Universidad de Hardvard, Verizon, Twitter	LinkedIn, NASA JPL, Departamento de Defensa de los E.U, Universidades de Columbia y Hardvard, Rakspace, Salesforce
	Socios	IBM, Cisco, Amazon, Microsoft, HP, VMWare	VMWare, redhat, RackSpace, mas de 50	redhat,VMWare, Cisco, HP	NE
Presencia en redes sociales		Activo	Muy Activo	Activo	Muy activo
Tipo de Licencia		GPL v3,0	Apache License v2.0	GPL v3,0	Apache License v2.0
Precio		Negociar con proveedor	$10500	$5000	Negociar con proveedor
Lenguaje de desarrollo		C 95%	Ruby 97%	Python 89,4%	Python 97.6%
Requerimientos de hardware	Servidor	12núcleos, 2GRAM, 100MB Disco x agente	4 núcleos, 16GB RAM, 142G Disco	4GB RAM, 20G Disco	NE
	Cliente	256MB RAM, 100MB Disco o 1G para Windows	NE	NE	NE

NE: no especificado oficialmente.

3. CONCLUSIONES

Este trabajo permitió conformar un marco de comparación basado en un conjunto de características aplicables a las CMT, para de forma sencilla y rápida poder conformar una descripción general que caracterice este tipo de productos en ausencia de expertos en el tema. El marco establece una comparación entre cuatro de las CMT más populares permitiendo a un usuario no experto tomar en consideración los aspectos planteados en el mismo para seleccionar la herramienta que sea la más adecuada a las necesidades de empresas e instituciones de pequeña, media y gran escala.

4. REFERENCIAS BIBLIOGRÁFICAS

1. W. Hummer, F. Rosenberg, F. Oliveira, and T. Eilam: "Testing idempotence for infrastructure as code," en Middleware 2013, Springer, 2013, pp. 368–388.

2. S. Nelson-Smith: "Test Driven Infrastructure with Chef", Segunda Edición, O'Reilly Media Inc., 2014.

3. M. Burgess: "Testable system administration," Communications of the ACM, Vol. 54, No. 3, pp. 44 - 49, 2011.

4. B. Siebert, M. van Eekelen, y J. Visser: "Evaluating the testing quality of software defined infrastructures", Thesis, Radboud University Nijmegen, 2014.

5. J. Humble y D. Farley: "*Continuous delivery: reliable software releases through build, test, and deployment automation"*, 2010.

6. "Sysadmin & Security Salary and Skill Report", 2015.

7. M. Hüttermann: "*DevOps for developers: integrate development and operations, the agile way"*, 2012.

8. C. Delettre: "Plateforme ouverte, évolutive, sécurisée et orientée utilisateur pour l'e-commerce," Universidad Nice Sophia Antipolis. Escuela Doctoral de Ciencias y Tecnologías de la Información y la Comunicación (STIC), 2014.

9. D. Spinellis: "Don't Install Software by Hand", IEE Computer Society, 2012.

10. D. A. Wheeler, "How to evaluate open source software/free software (OSS/FS) programs", [Online]: http://www.dwheeler.com/oss_fs_eval.html, 2011

11. K. Stol y M. Ali Babar: "A Comparison Framework for Open Source Software Evaluation Methods", Libro: "Open Source Software: New Horizons", pp. 389-394, 2010

12. J. Rahman: "Investigating Configuration Management Tools Usage in Large Infrastructure", Tesis de maestría en la Universidad de OSLO, 2012

13. M. Caballer Fernández: "Gestión de infraestructuras virtuales configuradas dinámicamente", Tesis doctoral en la Universidad Politécnica de Valencia, 2014.

14. K. Torberntsson y Y. Rydin: "A Study of Configuration Management Systems: Solutions for Deployment and Configuration of Software in a Cloud Environment", 2014.

15. T. Delaet, W. Joosen, y B. Vanbrabant: "A survey of system configuration tools", 2010.

16. S. Pandey: "Investigating Community, Reliability and Usability of CFEngine, Chef and Puppet", Tesis en la Universidad de Oslo, 2012.

17. A. Chandrasekar y G. Gibson: "A Comparative Study of Baremetal Provisioning Frameworks", 2014.

18. B. Vanbrabant: "A framework for integrated configuration management of distributed systems", Tesis doctoral en Universidad de KU Leuven, 2014.

5. SÍNTESIS CURRICULARES DE LOS AUTORES

Enrique Carbonell Muela. Licenciado en Ciencias de la Computación, graduado con honores en la Universidad Central de Las Villas (UCLV, 2010). Se incorporó a trabajar en DATYS en septiembre del 2010 donde ha trabajado en diferentes roles: desarrollador de software, líder de un equipo de desarrollo y actualmente se desempeña como DevOps Engineer en el equipo de infraestructura y operaciones. Es miembro de la Unión de Informáticos de Cuba y estudiante de la Maestría en Ciencias de La Computación de la UCLV en su 11na edición.

Ana María García Pérez. Licenciada en Cibernética-Matemática de la Universidad Central de Las Villas (UCLV, 1982). Profesora Titular. Doctora en Ciencias Técnicas del Instituto Superior Politécnico José Antonio Echevarría (CUJAE, 1997). Miembro del Comité Académico de la Maestría en Ciencia de la Computación de la UCLV y del Comité Académico de la Maestría en Gestión de Proyectos de la Universidad de las Ciencias Informáticas (UCI). Jefe de la Carrera Ciencia de la Computación de la UCLV (1995–2009). Ha dictado pre y postgrado como profesora invitada en varias universidades latinoamericanas, entre ellas: Universidad de Antioquia de Medellín, Colombia, UPC de Lima, Perú, Universidad de Guadalajara, México, Centro Universitario de la Ciénega de Ocotlán, Jalisco, México, Universidad Católica de Santa María de Arequipa, Perú y la Universidad de Managua, Nicaragua. Jefe del Grupo Científico de Programación e Ingeniería de Software del Centro de Estudios de Informática de la UCLV (2000 – 2008). Miembro del Comité de Editores de la Revista Cubana de Ciencias Informáticas (RCCI). Revisora de la revista ACIMED, Revista Cubana de Investigaciones en Ciencias de la Salud. Pertenece al CTN 18 de la ONN de Cuba para las normas cubanas del software siendo coautora de 4 normas ya registradas. Actualmente es Sub Directora Comercial de la División Territorial DESOFT Villa Clara.