**Puppet**

Es una configuración en el manejo de la solución que permite definir el estado de la infraestructura de la tecnología informática, y permite llegar a un estado deseado automáticamente. Puppet automatiza cada paso del proceso de entrega del producto, desde proporcionar máquinas virtuales y físicas para el funcionamiento y reporte. Desde fases anteriores a la codificación hasta las pruebas, el reléase y as actualizaciones.

* Tiempo libre para trabajar en proyectos que conllevan más valor en el negocio.
* Asegura consistencia, rentabilidad y estabilidad.
* Facilita la colaboración cercana entre el administrador del sistema y los desarrolladores, permitiendo más eficiencia en la entrega de código limpio que significa más valor al negocio

**Open Source Puppet**

Es flexible, una solución personalizable disponible bajo licencia de Apache 2.0, diseñado para ayudar en la automatización de la administración del Sistema en tareas repetitivas regularmente.

**La forma Como trabaja:**

* **Definir**

Primero se define el estado deseado de la computadora, almacenamiento y los dispositivos de red utilizando una simple y ponderosa declaración de configuración del lenguaje.

* **Simular**

Después uno puede similar el cambio de configuración antes de que se lleven a la vida real, así que no se sabe exactamente que pasara si el cambio es hecho.

* **Reforzar**

Puppet refuerza el estado deseado de la máquina, y automáticamente resuelve la configuración que no funciona.

* **Reporte**

Muchos reportes están disponibles en Puppet. Se puede obtener detalles en la configuración exacta del Sistema, incluyendo los cambios que se hicieron.

**Salt**

Es una plataforma basada en Python, es opensource nos sirve para el manejo de configuración y ejecución remota de una aplicación. Soportando la infraestructura como código, acercándose al despliegue y el manejo en la nube, compite principalmente con Puppet, chef y Ansible.

Salt se originó de la necesidad de tener una colección de datos rápido y ejecución del Sistema de administración el autor de Salt, Thomas S Hatch, había creado numerosas soluciones para muchas compañías para resolver problemas que se encuentren.

**Diseño**

Salt ha sido diseñado para ser altamente modular y fácilmente extensible. La meta del diseño es hace Salt fácilmente moldeable para diversas aplicaciones

El diseño modular de Salt crea módulos de Python que manejan ciertos aspectos de la disponibilidad de Salt. Estos módulos permiten interacción entre las necesidades del desarrollador y el administrador del sistema

**Ansible**

Es una plataforma de software libre para configurar y administrar computadoras. Combina instalación multi-nodo, ejecuciones de tareas ad hoc y administración de configuraciones. Adicionalmente, Ansible es categorizado como una herramienta de orquestación.Maneja [nodos](http://es.wikipedia.org/wiki/Nodo_%28inform%C3%A1tica%29) a través de [SSH](http://es.wikipedia.org/wiki/SSH) y no requiere ningún software remoto adicional (excepto Python 2.4 o posterior para instalarlo. Dispone de módulos que trabajan sobre [JSON](http://es.wikipedia.org/wiki/JSON) y la salida estándar puede ser escrita en cualquier lenguaje. Nativamente utiliza [YAML](http://es.wikipedia.org/wiki/YAML) para describir configuraciones reusables de los sistemas.

En contraste con otros programas de control de configuración como [Chef](http://es.wikipedia.org/w/index.php?title=Chef_%28software%29&action=edit&redlink=1) y [Puppet](http://es.wikipedia.org/wiki/Puppet_%28software%29), Ansible usa una [arquitectura sin agentes](http://en.wikipedia.org/wiki/Agentless_data_collection)Con la arquitectura basada en [agentes](http://es.wikipedia.org/wiki/Modelo_basado_en_agente), los nodos deben instalar localmente un proceso de comunicaciones con la máquina de control. Con la arquitectura sin agentes los nodos no necesitan instalar ni ejecutar en segundo plano ningún proceso que se comunique con la máquina de control. Este tipo de arquitectura reduce la sobrecarga de la red y previene el uso de estrategias de control más agresivas por parte del servidor (como puede ser la realización de [polling](http://es.wikipedia.org/wiki/Polling), con sus constantes operaciones de consulta).

El diseño de Ansible incluye:

* **Mínimo por naturaleza.** Los sistemas de administración no deben imponer dependencias adicionales.
* **Consistente.**
* **Seguro.** Ansible no instala agentes vulnerables en los nodos. Solamente se requiere OpenSSH que es considerado crítico y altamente testeado.
* **Alta confiabilidad.** El modelo de idempotencia es aplicado para las instalaciones y configuraciones, para prevenir efectos secundarios en la ejecución repetitiva de scripts.
* **Suave curva de aprendizaje.** Los *playbooks* usan un lenguaje descriptivo simple, basado en [YAML](http://es.wikipedia.org/wiki/YAML).

**GigaSpaces**

GigaSpaces es una compañía de tecnología israelí, que desarrolla middleware, establecida en el 2000 por Nati Shalom, con sede central en New York, y oficinas en Europa y Asia. Se ha centrado en aplicaciones para entornos Java y .Net, adoptando conceptos de Jini y JavaSpaces.

Inicialmente la compañía se centró en un software llamado XAP, que provee soluciones para Extreme Transaction Processing (más de 10000 transacciones concurrentes o 500 transacciones por segundo). En febrero de 2012, la compañía presentó Cloudify.

La compañía propone una nueva arquitectura: Plantea que el modelo de separación de capas clásico propuesto en arquitecturas empresariales de presentación-negocio-datos no es válido si quieres implementar soluciones de alto rendimiento y altamente escalables. Ellos plantean otra arquitectura en la que cada nodo es autocontenido y por lo tanto escalar es sencillo ya que simplemente hay que añadir más nodos. En otras palabras, se resume en computación distribuida y Grid de datos de nivel empresarial.

**Cloudify**

Es la solución PaaS para gestión de plataformas de Cloud (IaaS) basada en tecnologías open source. Cloudify ajusta el uso de los recursos bajo demanda por medio de una aplicación middleware de balanceo de carga que funciona bajo el principio de orquestación; detrás de la aplicación y desarrollo está GigaSpaces, una empresa de origen israelí con sede en Nueva York.

La aplicación ha sido definida por sus autores como PaaS de segunda generación. Hasta la versión 2.1 el propio stack estaba basado mayoritariamente con tecnologías Java. Desde la versión 2.2 hacia adelante se ha hecho un giro radical hacia el uso de orquestación con tecnología Chef (OpsCode), también conocido como orquestación por recetas.

En cuanto a su licenciamiento este middleware se encuentra bajo los términos de Apache Software License (ASL), aunque no tan libre como una GPLv2 o v3, garantiza de algún modo la supervivencia del proyecto mediante algún fork en el caso de que la empresa que lo desarrolla decidiese hacer un giro inesperado en su estrategia.

**Plutora**

Plutora es una aplicación empresarial desarrollada y dirigida para IT Release y manejo de entornos de pruebas. Es considerada una solución SaaS (Software As A Service).

La arquitectura de Plutora permite ver los detalles de la organización. Esto permite alinear el trabajo para cumplir objetivos estratégicos y optimiza proyectos y entregas. Permite escalar fácilmente para cumplir las demandas.

Mejora el acceso a información relevante acerca de la entrega y del ambiente de pruebas para tomar acciones inmediatas. Permite archivar información significativa. Entre los beneficios que ofrece es reducir fallos en el release, disminuir los costos de los cambios e incrementar la efectividad.

El marcado objetivo de Plutora son las instituciones que poseen departamentos de IT mayores a 100 personas. Esta característica lo hace relevante en todas las industrias y en todas las ubicaciones.