Conceptos
Herramientas
Puppet
Chef
SaltStack
Ansible

# CDA. Automatización en centros de datos

Conceptos básicos
Puppet
Chef
SaltStack

Ansible

**Centros de datos**
3º Grado en Ingeniería Informática
ESEI

15 de noviembre de 2018

Conceptos

Herramientas

Puppet

Chef

SaltStack

Ansible

# Contenido

1. Conceptos

2. Herramientas
   - Puppet
   - Chef
   - SaltStack
   - Ansible

# Automatización de CD

Necesidad de herramientas para la gestión automática (o semiautomática) de las infraestructuras y servicios de los Centros de Datos

- Afrontar crecimiento del número de componentes que necesitan ser administrados
  - favorecido por auge de soluciones de virtualización
- Reutilización y replicación de soluciones a tareas repetitivas
  - automatiza arranque, parada y replicación de los sistemas
- Simplificación de las tareas de administración
  - información de configuración centralizada
  - tareas identificadas y documentadas
- Favorece buenas prácticas en la administración de sistemas
  - modularidad y reutilización de configuraciones (uso de sistemas de control de versiones)
  - mejora y despliegue continuo, posibilidad de testing
  - reducción de errores (y simplificación de su gestión)
  - simplificación de administración remota

# Conceptos relacionados

- **Orquestación de sistemas** automatización de la creación, despliegue y coordinación de la infraestructura y componentes de un sistema
  - vinculado al despliegue de forma coordinada de conjuntos de instancias bajo virtualización, contenerización y Cloud Computing
- **Aprovisionado de sistemas** conjunto de procesos necesarios para preparar un equipamiento o infraestructura para que pueda prestar el servicio esperado
  - tareas de instalación de los componentes/servicios necesarios, su configuración y su puesta en marcha
- **Infraestructure As Code (IaC)** plantea el aprovisionado y administración de los elementos de un Centro de Datos mediante el empleo de ”descripciones” textuales entendibles (tanto por administrador como por un software) y procesables de forma automática, en lugar de mediante interacción directa con los equipos finales
- **DevOps** unificación del proceso de desarrollo de software (Dev) y la operación/administración del software (Ops)
  - busca ligar tareas de desarrollo y administración de sistemas para agilizar despliegue de aplicaciones
  - vinculado con metodologías ágiles, hace uso extensivo de técnicas y tecnologías IaC

# Aproximaciones

## Herramientas declarativas *vs.* procedurales

**Declarativas.** describen cúal es la configuración deseada para la infraestructura/componente

- ofrecen un lenguaje para describir el estado en que debe quedar un componente determinado y las herramientas ejecutan las acciones necesarias (¿qué?)

**Procedurales.** definen qué tareas realizar para conseguir la configuración deseada

- ofrecen un lenguaje para describir cómo cambiar la configuración mediante una secuencia ordenada y explícita de tareas (¿cómo?)

## Métodos push *vs.* métodos pull

Configuración reside en un componente central (servidor, maestro, etc)

**Configuración *pull*:** desde el equipo a ser administrado se pide la configuración al servidor central

- requiere componentes específicos (agentes) en el lado del equipo administrado que realicen las correspondientes tareas

**Configuración *push*:** servidor central envía la configuración a cada equipo final

- uso de agentes en equipo administrado es opcional
- soluciones *agentless* no emplean elementos especiales en los equipos administrados

# Puppet I

Herramienta de automatización declarativa

- Describe estado deseado mediante un DSL(*Domain Specific Language*) basado en Ruby
  - Usa *manifests* que describen ”recursos” y su estado de forma independiente de la plataforma
  - Permite herencia
- Sigue esquema *pull* en una arquitectura cliente-servidor con agentes instalados en los equipos administrados
- Clientes consultan servidor periódicamente y aplican cambios de configuración
- Tareas son idempotentes, se ejecutan sólo si el estado de un nodo no encaja con la configuración
- `https://puppet.com/`

## Elementos

- **Puppet Master** mantiene configuración centralizada, recibe consultas e informes de estado de agentes, proporciona comandos a agentes
- **Puppet Agents** consultan Puppet Master, ejecutan comandos del maestro si se requiere, informan de resultados a Master
- **Puppet forge** coleccion de modulos
- **Puppet DB** informacion de cada nodo de la infraestrucutra

# Puppet II

## Ejemplo

```
package { 'openssh-server':
 ensure => installed,
}

file { '/etc/ssh/sshd_config':
 source  => 'puppet:///modules/sshd/sshd_config',
 owner   => 'root',
 group   => 'root',
 mode    => '0640',
 notify  => Service['sshd'], # sshd will restart whenever you edit this file.
 require => Package['openssh-server'],
 }

service { 'sshd':
 ensure     => running,
 enable     => true,
 hasstatus  => true,
 hasrestart => true,
}
```

Fuente: https://jjasghar.github.io/blog/2015/12/20/chef-puppet-ansible-salt-rosetta-stone/

# Chef I

Herramienta de automatización mixta (declarativa y [parcialmente] procedural)

- Describe estado deseado mediante un DSL(*Domain Specific Language*) basado en Ruby
  - Usa *receipes* (recetas), que contienen recursos que deben ser configurados en el estado declarado en ellas
- Sigue esquema *pull* en una arquitectura cliente-servidor con agentes instalados en los equipos administrados
  - También disponible `chef-solo` para configuración local en modo aislado
- `https://www.chef.io/`

## Elementos

- **Chef Workstation** controla el despliegue de las configuraciones desde el Chef Server a los nodos
  - usuario crea *cookbooks* (grupos de recetas) para enviar al Chef Server
- **Chef Server** gestiona configuraciones a aplicar a los nodos
  - centraliza estado, cookbooks y configuraciones de los Workstations, controla los clientes
- **Chef Agents** agentes en los nodos administrados que piden (*pull*) configuracion al servidor

# Chef II

- contactan con servidor para comprobar cambios de configuración (*run lists*), ejecutan las tareas para converger al estado requerido e informan del resultado al servidor

# Chef III

## Ejemplo

```
package 'openssh-server' do
  action :install
end

template '/etc/ssh/sshd_config' do
  source 'sshd_config.erb'
  owner 'root'
  group 'root'
  mode '0640'
  notifies :reload, 'service[ssh]'
end

service 'ssh' do
  action [:enable, :start]
  supports :status => true, :restart => true
end
```

Fuente: https://jjasghar.github.io/blog/2015/12/20/chef-puppet-ansible-salt-rosetta-stone/

# SaltStack I

Herramienta de automatización declarativa y de ejecución remota de comandos

- Esquema mixto (push-pull) basado en agentes
  - Push: envío de comandos a minions
  - Pull: minions solicitan info. de configuración
  - Mantiene nodos remotos en estado definidos
  - Describe estados en ficheros YAML
- `https://www.saltstack.com/`

## Elementos

- **Salt master** centraliza configuración y controla minions
  - *master daemon* ejecuta tareas para el master (autentica minions, comunica con minions, soporte CLI)
  - *salt client* (en misma máqina que Master) interacción con usuario, envía comandos a master
- **Minions** reciben comandos del master, ejecuta tareas y envía resultados a master, consultan cambios a master
- **Salt modules** conjunto de funciones que pueden ejecutarse desde *salt client*

# SaltStack II

## Ejemplo

```
openssh-server:

  pkg.installed:
    - name: openssh-server

  service.running:
    - name: sshd
    - enable: True
    - require:
      - pkg: openssh-server

  file.managed:
    - name: /etc/ssh/sshd_config
    - source: salt://ssh/sshd_config
    - user: root
    - group: root
    - mode: 640
    - watch_in:
      - service: openssh-server
```

Fuente: https://jjasghar.github.io/blog/2015/12/20/chef-puppet-ansible-salt-rosetta-stone/

# Ansible I

Sistema de ejecución remota para orquestar la ejecución de comandos y consultar datos en tareas de orquestacion y gestion de la configuracion

- Esquema procedural
- Modelo *push* sin agente (sólo requiere ssh y python en nodos)
- `https://www.ansible.com/`

## Elementos

- Playbooks: documents YAML con tareas a realizar
- Roles: colección de playbooks y variables
- Inventario: listado (agrupado) de nodos
- Módulos: scripts Python responsables de ejecutar las tareas

# Ansible II

## Ejemplo

```
- name: install the latest version of openssh-server
  package: name=openssh-server state=present

- template: src=/mytemplates/sshd_confige.j2
     dest=/etc/ssh/sshd_config
     owner=root
     group=root
     mode=0644

- service: name=ssh state=started
```

Fuente: https://jjasghar.github.io/blog/2015/12/20/chef-puppet-ansible-salt-rosetta-stone/