# SALTSTACK

Capitoul – 12 Avril 2018

- Julien Libourel -

# Administration système

> Avant
- Script shell
- Crontab
- Git

> Problématique
- Retour - monitoring
- Multiplication de VM (Spécificités)
- Configurations qui diffèrent

# Système de gestion de configurations

> Simplifier et automatiser le travail d'administration
- Centraliser - Tracer - Déployer

> Qualités
- Homogénéité
- Fiabilité
- Rapidité
- « Langage universel »

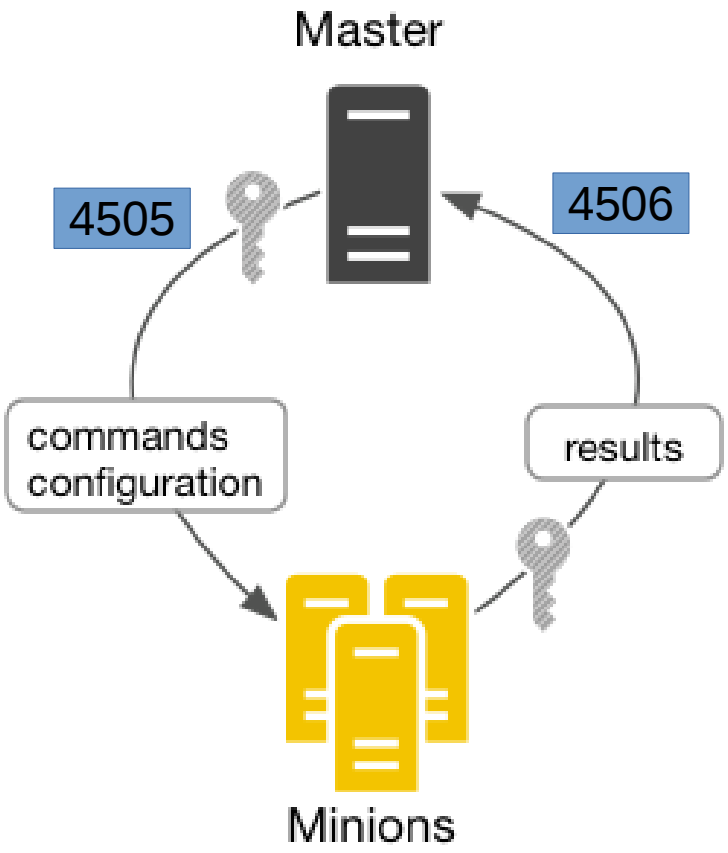
> Test
- Puppet, Salt

# Pourquoi Salt ?

> Framework d'exécution à distance
  - Gestion de configuration + shell distribué
> Apprentissage
  - Tutoriels
  - Langage (structurel)
  - Qualité de la documentation
  - Modules (+ 300)
> Salt OSS (salt open source)
  - Apache 2.0
  - Version actuelle 2018.3.0 (Oxygen)

# Architecture de Salt

- Authentification
- Communication
  - ZeroMQ
  - Msgpack
- Chiffrement
  - 256 bit (AES)



# Installation – Mise en route

## > Installation

- Ajout dépôt + apt

```
apt-get install salt-master # sur le Master (serveur)

apt-get install salt-minion # sur le Minion (client)
```

## > Configuration

- Minion
  - Positionner le nom du master dans /etc/salt/minion
  - Relancer le minion
- Master
  - Accepter la clé

```
salt-key -a minion1   # on accepte le clé de la machine monion1
```

# Exécuter une commande

```
salt '*' pkg.install cowsay
```

ciblage — module.fonction — argument

## > Exemples

```
salt '*' sys.doc pkg
salt '*' sys.doc pkg.install

salt '*' test.ping
salt '*' disk.usage
salt '*' cmd.run `uname -a`
salt '*' network.interfaces
```

Mode batch

```
salt '*' -b 10 test.ping
salt -G 'os:RedHat' --batch-size 25% apache.signal restart
```

# Ciblage

> Déterminer quels systèmes appliquent la commande.

> Exemples

```
salt '*' test.ping
salt 'minion1' disk.usage
salt 'minion*' disk.usage
salt -G 'os:Ubuntu' test.ping
salt -E 'minion[0-9]' test.ping
salt -L 'minion1,minion2' test.ping
salt -C 'G@os:Ubuntu and minion* or S@192.168.50.*' test.ping
```

# State

> Le fichier d'état (extension .sls)

> YAML (simple et lisible) + Jinja2 ...

> Structure de fichier cohérente /svr/salt/

/srv/salt/nettools.sls

```
install_network_packages:
  pkg.installed:
    - pkgs:
      - rsync
      - lftp
      - curl
```

id

module.fonction

arguments

> Application

```
salt 'minion1' state.apply nettools

salt 'minion1' pkg.install pkgs='["rsync", "lftp", "curl"]'
```

```
[root@saltmaster]# salt "minion1" state.apply nettools
minion1:
----------
          ID: install_network_packages
    Function: pkg.installed
      Result: True
     Comment: The following packages were installed/updated: lftp
              The following packages were already installed: rsync, curl
     Started: 10:03:26.414566
    Duration: 17515.017 ms
     Changes:
              ----------
              lftp:
                  ----------
                  new:
                      4.0.9-14.el6
                  old:
Summary for minion1
------------
Succeeded: 1 (changed=1)
Failed:    0
------------
Total states run:     1
Total run time:  17.515 s
[root@saltmaster]# salt "minion1" state.apply nettools
minion1:
  Name: install_network_packages - Function: pkg.installed - Result: Clean Started: - 10:05:41.750433
Duration: 1176.884 ms
Summary for minion1
------------
Succeeded: 1
Failed:    0
------------
Total states run:     1
Total run time:   1.177 s
```

Application du state nettools.sls

# Ex. State

> Configuration sssd

- /srv/salt/sssd/init.sls
  - Jinja2
  - include / extend
  - Modules (virtuels)
  - Imperatif / Declaratif
    - require
  - watch
  - failhard
  - file

```
{% set source = salt['grains.filter_by']({
  'fixe': {'target': 'salt://sssd/sssd.conf'},
  'cao': {'target': 'salt://sssd/sssd_cao.conf'},
  },
grain='laas',
default='fixe'
)%}

Include:
   - sssd.nsswitch

sssd-dependencies:
  pkg.installed:
    - name: sssd-ldap

sssd:
  pkg:
    - installed
  service:
    - running
    - enable: true
    - watch:
       - file: /etc/sssd/sssd.conf
  require:
    - pkg: sssd-dependencies

/etc/sssd/sssd.conf:
  file.managed:
    - source: {{ source.target }}
    - user: root
    - group: root
    - mode: 600
    - require:
       - pkg: sssd
```

/srv/salt/sssd/nsswitch.sls

```
/etc/nsswitch.conf:
  file.managed:
    - user: root
    - source: salt://sssd/nsswitch.conf
    - mode: 644
```

# top.sls

> Le fichier top.sls est utilisé pour appliquer plusieurs states aux minions.

Top File

```
base:
  '*':
    - vim
    - scripts
    - users
  '*web*':
    - apache
    - python
    - django
  '*db*':
    - mysql
```

/srv/salt/top.sls

states files

```
vim:
  pkg.installed

/etc/vimrc:
  file.managed:
    - source: salt://vimrc
    - mode: 644
    - user: root
    - group: root
```

/srv/salt/vim.sls

```
apache:
  pkg.installed:
    - name: {{ apache.server }}
  service.running:
    - name: {{ apache.service }}
    - enable: True
```

/srv/salt/apache.sls

> Application

```
salt '*' state.highstate
```

# Données

## > Grains

- Informations statiques sur les minions.

```
[root@master ~]# salt "minion1" grains.ls
Minion1 :
    - num_cpus
    - num_gpus
    - os
    - os_family
    - host
    - hwaddr_interfaces
    - kernel
    - mem_total
    - num_cpus
    - num_gpus
    - pythonversion
    - saltversion
...
```

Exemple dans une exécution de commande

```
salt -G 'os:Ubuntu' test.ping
```

Exemple dans un state (Jinja2)

```
{% if grains['os'] == 'Ubuntu' %}
/etc/default/nfs-common:
  file.managed:
    - user: root
    - source: salt://states/autofs/nfs-common
    - mode: 644
{% endif %}
```

## > Pillars

- Variables stockées sur le master et affectées à un ou plusieurs minions.

Exemple de pillar

```
{% if grains['os_family'] == 'RedHat' %}
editor: vim-enhanced
{% elif grains['os_family'] == 'Debian' %}
editor: vim
{% endif %}
```

Utilisation dans un state

```
vim installed:
  pkg.installed:
    - name: {{ pillar['editor'] }}
```

# Autres choses intéressantes

## > Formulas

Collection de States et de Pillars qui configurent une application ou un composant système. Ils sont mis à disposition par la communauté et facilement adaptables.

## > Returners (minion)

Les résultats des commandes envoyées aux minions peuvent être redirigées vers des bases externes.

## > Runners (master)

Modules qui s'exécutent sur le Salt master pour effectuer des tâches de support.

## > Reactor (master)

Système qui permet de réagir à la réception d'événements.

## > Beacons (minion)

permettent de générer des événements lorsque qu'une activité système à lieu au niveau fichier, charge, service, shell (authentification), usage du disque et du réseau.

# Aller Plus loin

> salt-ssh

Pour exécuter des commandes Salt via ssh sans installer de minion.

> salt-call

Pour exécuter des commandes Salt sur le minion.

SALT ENTREPRISE

Syndic

https://docs.saltstack.com

Windows

Os X

Salt-virt

Salt-cloud

# Questions ?