Administration avec Salt

Alexandre Ancel

ancel@math.unistra.fr

Institut de Recherche Mathématique Avancée (IRMA),
Centre de Modélisation et de Simulation de Strasbourg (Cemosis),
Université de Strasbourg

15 mars 2016

Plan de la présentation

Salt : Introduction

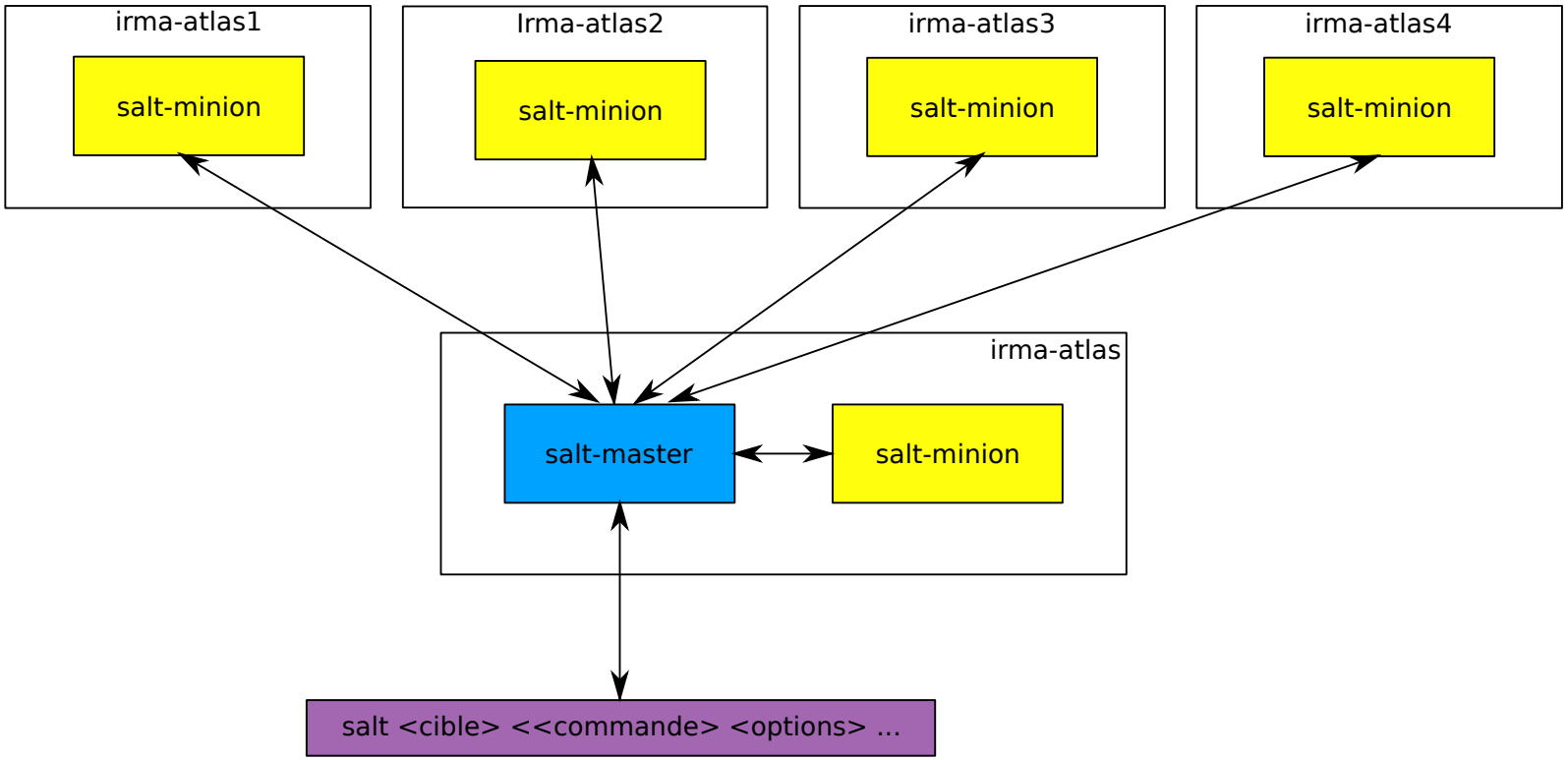
- Gestion de configuration et exécution de commandes à distance
- Open Source : https://github.com/saltstack/salt
- Portabilité : Linux, Windows et Mac OS X
- Ecrit en python
- Alternatives : cluster-ssh, Puppet, Ansible, Chef

Salt : Composants

- Deux types de machines (Services)
  master (serveur central) et minion (serveur à configurer)
- Composants :
  - Modules d'exécution :
    Fonctions disponibles pour l'exécution à distance sur les minions :
    https://docs.saltstack.com/en/latest/ref/modules/
  - Grains :
    Informations statiques à propos des minions :
    Système d'exploitation, mémoire, etc.
  - Salt SSH :
    Permet l'exécution de commandes avec SSH sur des machines sans service minion
  - Modules d'état :
    Appliquer, mettre en place ou changer une configuration
    - Représentation déclarative ou impérative de la configuration d'un système
  - Pillars :
    Variables définies par l'administrateur. Elles sont assignées à un ou plusieurs minions. Ex : Chemins, paramètres de configuration ...

Salt : Configuration initiale

- Configuration du cluster irma-atlas :



- Guides d'installation : Debian, RHEL, CentOS, Windows, OS X ... https://docs.saltstack.com/en/latest/topics/installation/index.html#platform-specific-installation-instructions

Salt : Configuration initiale

- Installer salt-master, salt-minion
- Obtenir la clé publique du master :

```
[master] sudo salt-key -F master
```

- /etc/salt/minion : master, master_finger
- Vérifier les minions qui souhaitent se connecter au master :

```
[master] sudo salt-key --list all
```

- Vérifier la correspondance des clés :

```
[minion] sudo salt-call key.finger --local
[master] sudo salt-key -f <ID_minion>
```

- Accepter les clés dans le master :

```
[master] sudo salt-key -a <ID_minion>
```

Alternative : Salt-SSH

- Approche server-only (Agentless)
  - Connexion par SSH
  - Déploiement d'une version light de Salt (repertoire temporaire)
  - Lancement des commandes
- Installer salt-ssh
- Ajouter le serveur à gérer au "Roster" : /etc/salt/roster

```
vivabrain:
  host: 130.79.210.69
  user: root
```

- Déployer la clé privée générée par salt pour l'utilisation :

```
ssh-copy-id -i /etc/salt/pki/master/ssh/salt-ssh.rsa.pub \
root@vivabrain
```

- Exécution de commandes à distance

Salt : Exécution de commandes à distance

- Commande générique : (salt-ssh pour Salt SSH)

```
sudo salt 'cible' commande options ...
```

- Débogage : -l LOGLEVEL
- Logging : /var/log/salt/master
- Ciblage :
  - Expressions régulières :

```
sudo salt -E 'minion[0-9]' ...
```

  - Liste de machines :

```
sudo salt -L 'minion1,minion2' ...
```

  - Liste par grain(s) :

```
sudo salt -G 'grains' ...
```

Salt : Ciblage : Grains

- Lister les grains disponibles :
  - Liste les valeurs courantes :

```
salt '*' grains.items
```

  - Liste les entrées disponibles (sans valeurs) :

```
sudo salt '*' grains.ls
```

- Exemples :

```
sudo salt '*' grains.item num_gpus
sudo salt -G 'os:Ubuntu' cmd.run "landscape-sysinfo"
sudo salt -G 'cpuarch:x86_64' cmd.run "arch"
sudo salt -G 'gpus:vendor:nvidia' cmd.run nvidia-smi
sudo salt -G 'manufacturer:Dell*' cmd.run "ls /etc/opt/dell"
```

Salt : Exemples de commandes utiles

- Exemples de commandes :

```
# Commandes
sudo salt '*' cmd.run 'ls -R *'

# Systèmes de paquets
sudo salt '*' pkg.install apache2
sudo salt '*' pkg.list_upgrades

# Services
sudo salt '*' service.status allinea_licensing_init

# Réseau
sudo salt 'irma-atlas1.u-strasbg.fr' network.ip_addrs
sudo salt 'irma-atlas1.u-strasbg.fr' network.get_hostname
```

- Documentation de référence :
  https://docs.saltstack.com/en/develop/ref/modules/all/index.html

Salt : Gestion des états

- Que sont les états ?
- Localisés par défaut dans /srv/salt
- Fichiers d'état écrits en YAML (par défaut, json aussi disponible) avec extension .sls
- Etat top.sls particulier

```
sudo salt '*' state.apply
```

- Exemple d'autres fichiers .sls : Exemple : slurm.sls

```
sudo salt '*' state.apply slurm
```

- Liste des états disponibles : https://docs.saltstack.com/en/develop/ref/states/all/index.html
- Ou voir les fichiers installés avec salt-common (/usr/lib/python2.7/dist-packages/salt/)

Salt : Exemple : Gestion de la configuration de slurm

- Gestion du service slurm : slurm.sls

```
slurm-llnl:
    pkg.installed:
        - name: slurm-llnl
    service.running:
        - name: slurm-llnl
        - enable: True
        - watch:
            - file: /etc/slurm-llnl/slurm.prolog
            - file: /etc/slurm-llnl/slurm.epilog
            - file: /etc/slurm-llnl/slurm.conf
```

Salt : Exemple : Gestion de la configuration de slurm

- Gestion des fichiers de configuration : slurm.sls

```
/etc/slurm-llnl/slurm.conf:
    file.managed:
        - name: /etc/slurm-llnl/slurm.conf
        - source: salt://etc/slurm-llnl/slurm.conf

/etc/slurm-llnl/slurm.prolog:
    file.managed:
        - name: /etc/slurm-llnl/slurm.prolog
        - source: salt://etc/slurm-llnl/slurm.prolog

/etc/slurm-llnl/slurm.epilog:
    file.managed:
        - name: /etc/slurm-llnl/slurm.epilog
        - source: salt://etc/slurm-llnl/slurm.epilog
```

Salt : Exemple : Gestion de la configuration de slurm

- Fichier top : top.sls

```
base:
    '*':
        - slurm

    'irma-atlas.u-strasbg.fr':
        - slurmdbd
```

Pillars

- Variables définies par l'utilisateur
- Stockée dans des fichiers YAML (par défaut) dans /srv/pillar
- Pillar top.sls particulier
- Mettre à jour avec :

```
salt "*" saltutil.refresh_pillar
```

- Interroger avec :

```
salt '*' pillar.items
```

- Utilisation d'un moteur de rendu :
  Jinja (par défaut, aussi disponibles : Mako et Wempy)

Jinja

- Template engine
- Création de fichiers templates en python (Etats, pillars ...)
- Jinja effectue ensuite un "rendu" de ces templates en substituant les valeurs. Exemple :

```
>>> from jinja2 import Template
>>> template = Template('Hello {{ name }}!')
>>> template.render(name='John Doe')
u'Hello John Doe!'
```

Jinja

- Quelques usages dans salt :
  - Conditionnelles :

```
pkgs:
    {% if grains['os_family'] == 'RedHat' %}
    apache: httpd
    {% elif grains['os_family'] == 'Debian' %}
    apache: apache2
    {% endif %}
```

  - Utilisation de variables :

```
webserver:
  pkg.installed:
    - name: {{ pillar['pkgs']['apache'] }}
```

Jinja

- Quelques usages dans salt :
  - Boucles :

```
{% set admin_user = "root" %}
{% set home_dir = "/root" %}
{% for d in ["network", "db"] %}
{{ home_dir }}/admin_scripts/{{d}}:
    file.recurse:
        - source: salt://admin_scripts/{{ d }}
        - user: {{ admin_user }}
        - dir_mode: 700
        - file_mode: 700
        - include_empty: True
{% endfor %}
```

Jinja

- Quelques usages dans salt : Template de fichiers
  - Exemple : lien
  - Fichier d'état :

```
/etc/slurm-llnl/slurm.epilog:
    file.managed:
        - name: /etc/slurm-llnl/slurm.conf
        - source: salt://etc/slurm-llnl/slurm.conf
        - template: jinja
```

  - Fichier slurm.conf :

```
{{ pillar['headers']['conf'] }}
ClusterName=atlas
...
```

Exemple : Gestion de la configuration infiniband

- /srv/pillar/ib-atlas.sls

```
network:
    ib0:
        {% if grains['id'] == 'irma-atlas.u-strasbg.fr' %}
        address: 192.168.100.10
        {% elif grains['id'] == 'irma-atlas1.u-strasbg.fr' %}
        address: 192.168.100.11
        {% elif grains['id'] == 'irma-atlas2.u-strasbg.fr' %}
        address: 192.168.100.12
        {% elif grains['id'] == 'irma-atlas3.u-strasbg.fr' %}
        address: 192.168.100.13
        {% elif grains['id'] == 'irma-atlas4.u-strasbg.fr' %}
        address: 192.168.100.14
        {% endif %}
        netmask: 255.255.255.0
```

Exemple : Gestion de la configuration infiniband

- /srv/salt/manage_network.sls

```
ib0:
  network.managed:
    - enabled: True
    - type: ib
    - proto: none
    - ipaddr: {{ pillar['network']['ib0']['address'] }}
    - netmask: {{ pillar['network']['ib0']['netmask'] }}
```

- /srv/salt/top.sls

```
base:
    '*':
        manage_network
```

Conclusion : Salt

- Exécution de commandes à distances (Module d'exécution)
- Gestion centralisée des configurations et déploiement facile (Etats)
- Accès fin aux informations des serveurs (Grains)
- Création de templates et rendu de ces templates (Pillars et Etats)
- Portable (Distributions Linux majeures, Windows et OS X) & Open Source