

Saltstack - Le sel de la pile

Matthieu Herrb



Capitoul - 17 octobre 2017

<http://homepages.laas.fr/matthieu/talks/saltstack-capitoul.pdf>



Ce document est sous licence

Creative Commons Paternité - Partage à l'Identique 3.0 non transposé.

Le texte complet de cette licence est disponible à l'adresse :

<http://creativecommons.org/licenses/by-sa/3.0/>

Agenda

- 1 Introduction
- 2 Salt au LAAS
- 3 Conclusion

Agenda

1 Introduction

2 Salt au LAAS

3 Conclusion

Introduction

Salt c'est :

- un système de gestion de configuration
 - maintient des *états* sur des *nœuds*
- un système d'exécution distribuée
 - exécute des commandes et récupère de l'information sur des *nœuds*

Marketing :

- simplicité
- passage à l'échelle de milliers de nœuds
- protocole standard sécurisé et chiffré
- performance

Salt en quelques mots

- Client/Serveur
- Description en YAML
- Templates via Jinja2
- Extensions en python

Jinja2 Minion
Python YAML Pillar
ZMQ Grains
Master

Salt - principe

- Les nœuds sont des *minions* (*sous-fifres*)
- les *grains* décrivent le minion:
 - configuration système
 - grains définis par l'admin
- L'état désiré est décrit à l'aide de *formules*
- Les formules utilisent des *modules*
 - écrits en python
 - abstraction des spécificités des systèmes
- Le *pillier* permet de fournir des paramètres aux formules et aux états.
- Les *reacteurs* définissent des actions pour réagir aux événements liés à l'exécution.

Quelques exemples

Exécution distribuée

```
master# salt '*' cmd.run 'reboot'  
master# salt '*' system.reboot
```

Gestion de configuration

```
master# salt '*' pkg.upgrade sudo  
master# salt -G applis:robotique state.apply install-ros
```

Mode pull

```
minion# salt-call state.apply test=True
```


Authentification Mutuelle des minions et du maître

- bi-clés
- approbation préalable des minions par le maître
- chiffrement des échanges

Notion d'ACL pour les non root

- permet de contrôler les droits d'exécution des états et l'accès aux ressources
- permet d'utiliser une authentification externe (LDAP)

Mécanisme d'accès à des données chiffrées (PGP)

- pour les données dans le pilier
- déchiffrées par clé privée sur le minion

Agenda

1 Introduction

2 Salt au LAAS

3 Conclusion

- présenté à Capitoul par Julien Libourel en avril 2018
- depuis, utilisé pour déploiement et maintien à jour :
 - des postes sous Ubuntu 18.04 (86)
 - des postes CentOS 7 de la salle CAO (18)
 - les nouvelles VM serveur (CentOS et Ubuntu) (33)
- et pour gérer l'arrêt de toutes les VM Linux

Approche Bottom-Up:

- décrire des états individuels d'abord,
- puis factoriser.

Postes de travail

- installation minimale
- installation de Salt
- config miroir LAAS des dépôts de paquets
- installation des paquets
- configuration LDAP + Kerberos + client NFS / automount
- paramétrage IPv6
- déploiement clés SSH sysadmin pour accès root
- configuration syslog centralisé
- configuration client OCSinventory
- install et configuration LiveNavigator pour les portables

- Un état par serveur (ciblé par le nom)
- Niveaux d'automatisation variables
- Migration incrémentale de la configuration vers Salt
- Configuration réseau statique

- un master (dans une VM CentOS)
- essentiellement mode pull (via cron sur les minions)
- reacteur pour envoi de rapports par mail
- (pour l'instant) pas vraiment d'environnement de développement séparé

Exemples

- Serveur web basique
- Déploiement d'une application

Quelques difficultés

- Existant très hétérogène
travail en parallèle pour standardiser les configurations
- Plus d'une façon de faire chaque chose
- Qualité / couverture inégales des formules / states existants
→ pas mal de développements/ adaptations maison.
- Contribuer des formules / states / modules génériques est dur
- Relativement gourmand en ressources
- Fonctionne mal avec double pile IPv4 / IPv6

Agenda

1 Introduction

2 Salt au LAAS

3 Conclusion

Conclusion

- Le besoin d'harmonisation / automatisation de la gestion des configuration est de plus en plus important.
- Saltstack répond à nos besoins.
- Nécessite plus de pratique pour en tirer tous les bénéfices.
- Investir dans le développement de formules et de modules maison.
- Ne pas vouloir en faire trop d'un coup.