Déploiement et configuration avec Ansible

Simon Delamare
Laboratoire de l'Informatique du Parallélisme

Journée ARAMIS, 13 avril 2017

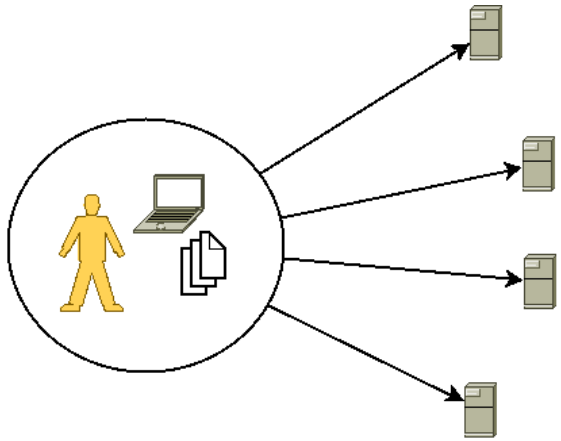
Introduction

Qu'est-ce qu'Ansible

- Automatisation des tâches d'admin
  - Sur un parc de machines
  - Déployer des applications, configurer, gérer les services
- Développement :
  - `https://github.com/ansible/ansible`
  - Écrit en python (les utilisateurs manipulent surtout du YAML)
  - GPLv3
  - Première version il y a 5 ans, version 2.2.1 il y a 3 mois

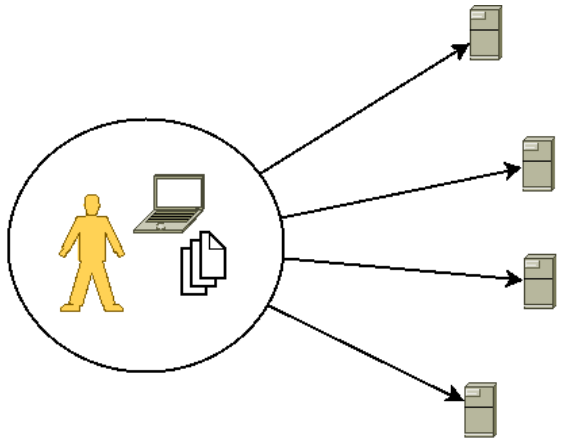
Principes sur les outils de gestion de configuration

- “Recettes” de configuration centralisées, appliquées sur les machines (= noeuds)
  - Pas de config “à la main”



Principes sur les outils de gestion de configuration

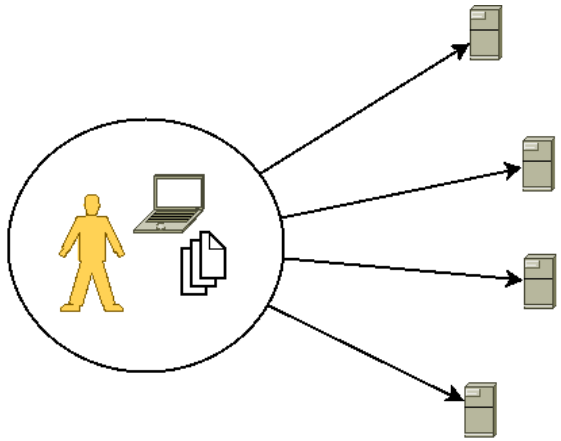
- “Recettes” de configuration centralisées, appliquées sur les machines (= noeuds)
  - Pas de config “à la main”



- Fournit des modules de haut niveau pour manipulations courantes (vs. script shell) : Installation de paquets, gestion de services

Principes sur les outils de gestion de configuration

- “Recettes” de configuration centralisées, appliquées sur les machines (= noeuds)
  - Pas de config “à la main”



- Fournit des modules de haut niveau pour manipulations courantes (vs. script shell) : Installation de paquets, gestion de services
- On décrit l’état souhaité plutôt que les étapes pour y arriver (vs. script shell)
  - Idempotence : une recette aboutit au même résultat qu’on l’applique une ou plusieurs fois
  - Les recettes doivent être écrites de cette façon pour que l’état soit stable

Rapport avec le devops ?

- Centralisation des configurations
  - *Infrastructure as a code*: L'état d'une l'infra décrite par des recettes
  - Travail en équipe (utilisation d'un VCS)
  - Partage du savoir : Les devs connaissent l'env de prod, les ops connaissent les modification à appliquer à la prod. avant le déploiement d'une appli.
- Automatisation
  - Moins de tâches répétitives, manuelles $\rightarrow$ Raccourcir les cycles de dev./publication/déploiement/utilisation
  - Il devient ~~très~~ plus simple d'instancier une plateforme de test

Les autres outils

- Parmi les plus connus :
  - CFEngine
  - Chef
  - Puppet
  - Salt
  - Ansible
- Avantages souvent cités pour Ansible :
  - Simplicité :
    - Recette en YAML
    - Rien à installer sur les noeuds
    - Mode push (par défaut): pas de serveur dédié
  - Complet : Description du parc machine, commandes ad-hoc, ...
- Inconvénients :
  - Recettes : plus une succession de tâches à appliquer qu'une vision "haut niveau" du résultat souhaité
  - Risque de mauvaises pratiques, erreurs non détectées
  - Peut être plus lent (tâches appliquées individuellement au lieu d'être analysées globalement)

Bases d'Ansible

Bases : Installation

- Installation, sur le poste de travail de l'utilisateur (le contrôleur) :

```
pip install ansible
```

Bases : Installation

- Installation, sur le poste de travail de l'utilisateur (le contrôleur) :

```
pip install ansible
```

- Installation, sur les noeuds : *ssh*, *python*

Bases : Installation

- Installation, sur le poste de travail de l'utilisateur (le contrôleur) :

```
pip install ansible
```

- Installation, sur les noeuds : *ssh*, *python*
- Pas d'agent, tout passe par SSH.

Bases : Inventaire

- Fichier avec la liste des noeuds gérés par Ansible

```
web.example.org
db.example.org
```

Bases : Inventaire

- Fichier avec la liste des noeuds gérés par Ansible
- Organisés en groupes

```
[webservers]
web1.example.org
web2.example.org

[dbservers]
db1.example.org
db2.example.org
```

Bases : Inventaire

- Fichier avec la liste des noeuds gérés par Ansible
- Organisés en groupes
- Associés à des variables

```
[webservers]
web1.example.org
web2.example.org

[dbservers]
db1.example.org  master=true
db2.example.org

[webservers:vars]
http_port=80
```

Bases : Modules

Unité de base pour la réalisation d'une tâche sur un noeud.

- Abstraction par Ansible
- Exemples : apt, yum, service, shell, copy, template, file, lineinfile, cron, user...
- Liste des modules (et leurs arguments):
  - `http://docs.ansible.com/ansible/modules_by_category.html`

```
ansible all -m apt -a pkg=vim
ansible webservers -m copy -a src=~/index.html dest=/var/www/index.html
ansible db1.example.org -m shell -a "ps aux"
```

Démo 1

Bases : Playbooks

- Fichier décrivant l'application d'une opération
  - tâches à appliquer sur certains noeuds

```
- hosts: webservers
  vars:
    http_port: 80
  tasks:
  - name: Install Apache
    apt: pkg=apache2 state=present
  - name: Install default index.html
    copy: src=index.html dest=/var/www/index.html
```

Bases : Playbooks

- Fichier décrivant l'application d'une opération
  - tâches à appliquer sur certains noeuds
  - *handlers*: tâches à appliquer en cas de changement

```
- hosts: webservers
  vars:
    http_port: 80
  tasks:
  - name: Install Apache
    apt: pkg=apache2 state=present
  - name: Copy default index.html
    copy: src=index.html dest=/var/www/index.html
  - name: Setup default configuration
    template: src=apache.conf.j2 dest=/etc/apache2/site-enabled/000-default.conf
    notify: Reload apache
  handlers:
  - name: Reload apache
    service: name=apache2 state=reloaded
```

Bases : Playbooks

- Fichier décrivant l'application d'une opération
  - tâches à appliquer sur certains noeuds
  - *handlers*: tâches à appliquer en cas de changement

```
- hosts: webservers
  vars:
    http_port: 80
  tasks:
  - name: Install Apache
    apt: pkg=apache2 state=present
  - name: Copy default index.html
    copy: src=index.html dest=/var/www/index.html
  - name: Setup default configuration
    template: src=apache.conf.j2 dest=/etc/apache2/site-enabled/000-default.conf
    notify: Reload apache
  handlers:
  - name: Reload apache
    service: name=apache2 state=reloaded
```

- *template*: génère (et copie) un fichier "paramétrisé":
  - Utilise jinja2
  - conditions, boucles, etc.

Bases : Playbooks

- *template*: génère (et copie) un fichier “paramétrisé”:
  - Utilise jinja2
  - conditions, boucles, etc.

```
<VirtualHost *: {{ http_port }} >
  ServerAdmin webmaster@localhost
  DocumentRoot /var/www/
  <Directory "/var/www/">
    AllowOverride All
  </Directory>
</VirtualHost>
```

Bases : Rôles

- Organisation fonctionnelle des playbooks
  - Playbook trop gros
  - Réutilisation

```
roles/
    apache-server/
        tasks/
            main.yml
        handlers/
            main.yml
        files/
            index.html
        template/
            apache.conf.j2
```

Bases : Rôles

- Organisation fonctionnelle des playbooks
  - Playbook trop gros
  - Réutilisation

```
roles/
    apache-server/
        tasks/
            main.yml
        handlers/
            main.yml
        files/
            index.html
        template/
            apache.conf.j2
```

Ansible Galaxy (`https://galaxy.ansible.com/`) : Dépot de rôles prêts à l'emploi

Bases : Rôles

- organisation fonctionnelle des playbooks
- appelés depuis un playbook de plus haut niveau

```
- hosts: webservers
  roles:
  - apache-server
  - example-website
```

Bases : Rôles

- organisation fonctionnelle des playbooks
- appelés depuis un playbook de plus haut niveau

```
- hosts: webservers
  roles:
  - common-admin-tools
  - common-monitoring-tools
  - apache-server
  - example-website

- hosts: dbservers
  roles:
  - common-admin-tools
  - common-monitoring-tools
  - postgres-server
```

Bases : Rôles

- organisation fonctionnelle des playbooks
- appelés depuis un playbook de plus haut niveau

```
- hosts: all
  roles:
  - common-admin-tools
  - common-monitoring-tools

- hosts: webservers
  - apache-server
  - example-website

- hosts: dbservers
  roles:
  - postgres-server
```

Bases : Variables

- Noeud associé à des variables
- Peuvent être définies, dans l'inventaire, le playbook, les rôles, les *facts*
  - facts : variables générées dynamiquement avec des informations sur la machine :
    - *ansible_distribution_release*
    - *ansible_default_ipv4.address*
    - *ansible_memtotal_mb*
    - ...

Bases : Variables

- Noeud associé à des variables
- Peuvent-être définies, dans l'inventaire, le playbook, les rôles, les *facts*
- Utilisées dans les playbooks :

```
- name: Installation du paquet foo
  apt: name=foo={{ foo_version }}

- name: Demarrage du service redundant
  service: name=redundant state=started
  when: "not backupserver"
```

Bases : Variables

- Noeud associé à des variables
- Peuvent-être définies, dans l'inventaire, le playbook, les rôles, les *facts*
- Utilisées dans les playbooks :

```
- name: Installation du paquet foo
  apt: name=foo={{ foo_version }}

- name: Demarrage du service redundant
  service: name=redundant state=started
  when: "not backupserver"
```

- Ou dans les templates

```
#backport.list.j2
deb http://ftp.fr.debian.org/debian {{ ansible_distribution_release }}-backports main
```

Bases : Variables

- Noeud associé à des variables
- Peuvent-être définies, dans l'inventaire, le playbook, les rôles, les *facts*
- Utilisées dans les playbooks :

```
- name: Installation du paquet foo
  apt: name=foo={{ foo_version }}

- name: Demarrage du service redundant
  service: name=redundant state=started
  when: "not backupserver"
```

- Ou dans les templates

```
#backport.list.j2
deb http://ftp.fr.debian.org/debian {{ ansible_distribution_release }}-backports main
```

- Il est possible d'accéder aux groupes et aux variables des autres noeuds

Bases : Variables

- Il est possible d'accéder aux groupes et aux variables des autres noeuds :

```
#dhcpd.conf.j2

group {
{% for host in groups['dhcpclients']|sort %}
        host {{ host }} {
            hardware ethernet {{ hostvars[host]['macaddr'] }};
            fixed-address {{ host }}.{{ domain }};
        }
{% endfor %}
}
```

Démo 3

Exemple au LIP

Gestion de l'infrastructure du LIP

- Environ 120 noeuds gérés
- 20 groupes, 46 rôles
- Recettes dans dépôt Git : collaborer, suivre les évolutions, revenir en arrière
- Y compris les secrets (mot de passe, etc.), chiffrés
- Génération automatique de pages d'information aux utilisateurs
- Gestion dynamique de l'inventaire (module maison)
- Intégration à Vagrant

Démo 4

Conclusion

Conclusion

- Ansible : Gestion des configurations, déploiement d'application, automatisation, gestion du parc de machines
  - S'inscrit pleinement dans une démarche devops
- Un survol des fonctionnalités
  - Vault, gestion de machines Windows et d'équipement réseau, les UI, le mode pull, etc.
  - Mais on peut déjà faire beaucoup
- À essayer !