Déploiement et gestion d’un parc machine avec Cloudinit et Ansible

François Legrand - LPNHE

Le LPNHE

- 200 personnes
- 300 comptes actifs
- 200 postes de travail (160 linux)
- 200 portables (130 linux)
- 50 serveurs (linux)
- Annuaire LDAP
- Stockage centralisé (Netapp + ceph)

Contexte historique

- Serveurs
  - Debian et CentOS
  - Installation « manuelle » + gestion complète par Ansible
- Postes de travail
  - Ubuntu
  - Installation et configuration initiale automatique via pxe+kickstart/kickseed
  - Gestion par Ansible (mode pull) pour les évolutions/maj etc...
- Portables
  - Ubuntu
  - Installation usine ou automatique ou manuelle
  - Gestion « manuelle »

Pourquoi évoluer ?

- Kickstart ancien et bientôt plus maintenu[1] + complexe + besoin de tout définir (pas vraiment possible de laisser les choix par défaut de l'installer)
- Toutes les confs étaient dans KS → lourd et peu modulaire
- Objectifs :
  - Séparer l'installation du système des configurations labo
  - Avoir plusieurs scénarios d'installation (postes, portables, bancs de test, etc.) → modules
  - Pouvoir configurer une machine déjà installée
  - Réutiliser les rôles et l'expertise ansible

[1] https://ubuntu.com/server/docs/install/autoinstall

Choix

- Cloud-init (autoinstall chez canonical) pour l’installation de base
  - Nouveau standard
  - Supporte la majorité des distributions
  - Simple
- Ansible pour la configuration
  - Expérience au labo
  - Utilisable pour toutes les distributions
  - Facile à utiliser/apprendre et déployer

Cloud-init

- Nouveau standard de configuration des machines
- À l’origine pour le cloud
- Fichier de conf (YAML) qui sera utilisé au premier démarrage pour configurer la machine
- Implémenté par toutes les distributions « majeures » (avec quelques variantes)
- Utilisable pour une instance cloud mais aussi pour une install de machine physique

cloud-init

Cloud-init

- Avantages :
  - Syntaxe simple (YAML)
  - Standard (à peu près)
  - Peu de paramétrages « obligatoires » (i.e. utilise les choix par défaut de l'installer si rien n'est spécifié)
  - Possibilité de spécifier des sections « interactives » (par ex choix du clavier)
- Inconvénients :
  - Nouvelle syntaxe, nouvel outil → Faut tout refaire
  - Documentation pas toujours complète et variations d'implémentation entre les distributions (certaines options non dispo sur certaines distribs)

Cloud-init : config PXE

Implémenté seulement
sur la version « serveur »

```
LABEL Install Ubuntu 20.04 workstation
  kernel
http://xxx.xxx.xxx.xxx/DISTRIBS/ubuntu/20.04/AMD64/vmlinuz
  append ramdisk_size=1500000 ip=dhcp
url=http://xxx.xxx.xxx.xxx/DISTRIBS/ubuntu/20.04/AMD64/ubuntu-
20.04.3-live-server-amd64.iso autoinstall ds=nocloud-
net;s=http://xxx.xxx.xxx.xxx/DISTRIBS/ubuntu/20.04/AMD64/
CLOUDININT/
cloud-config-url=http://xxx.xxx.xxx.xxx/DISTRIBS/ubuntu/20.04/AM
D64/CLOUDININT/workstation root=/dev/ram0
  initrd
http://xxx.xxx.xxx.xxx/DISTRIBS/ubuntu/20.04/AMD64/initrd
```

Fichier de configuration

Cloud-init : fichier de conf

```
#cloud-config
autoinstall:
  version: 1
  interactive-sections:
    - keyboard
  apt:
    geoip: true
    preserve_sources_list: false
    primary:
    - arches: [amd64, i386]
      uri: http://www-ftp.lip6.fr/pub/linux/distributions/Ubuntu/archive
    - arches: [default]
      uri: http://ports.ubuntu.com/ubuntu-ports
```

Possibilité de définir des sections interactives !

Cloud-init : fichier de conf

identity: {hostname: localhost, password: '6i1GofddsASDeflkd$MAcrOnTeTEDeFiOnwHp6hlNpDIbeXCnRSForEverlBFQmWxw5jjh20xOTNySHRqVH02/7iEMoRtAUxcOnsKKtR/', realname: toto, username: toto }

keyboard: {layout: us, variant: ''}

locale: fr_FR.UTF-8

timezone: Europe/Paris

Ubuntu 20.04 veut un user initial déclaré avec « identity »

Cloud-init : fichier de conf

Ubuntu 22.04 accepte aussi user-data !

```
user-data:
  users:
    - name: toto
      gecos: TOTO
      sudo: ALL=(ALL) NOPASSWD:ALL
      primary_group: root
      groups: [users, admin, sudo, adm]
      password: $6$uRM7eMeSChausSEttEsSoNtRoSEslwjtNlOVBGRAMoNSliPAUssITkU/….
      homedir: /home-local/toto
      shell: /bin/bash
      lock_passwd: false
```

Cloud-init : fichier de conf

```
packages:
  - ansible
  - apt-transport-https
  - git
  - openssh-client
  - python
  - python3
  - python-dev
  - ssh
  - sudo
  - vim
  - wget
snaps:
  - name: zoom-client
updates: security
package_update: true
```

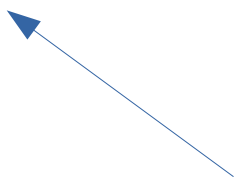
Installation d'un minimum de packages. Tout est ensuite fait avec ansible

Tout ce qui n'a pas été spécifié sera pris aux valeurs par défaut de l'installeur (e.g. partitionnement des disques, conf réseau,...)

Cloud-init : fichier de conf

late-commands:

- curtin in-target --target=/target -- lvextend -l +100%FREE /dev/mapper/ubuntu--vg-ubuntu--lv
- curtin in-target --target=/target -- resize2fs /dev/mapper/ubuntu--vg-ubuntu--lv
- **curtin in-target --target=/target -- ansible-pull -d /tmp/ansible-pull –purge -i /tmp/ansible-pull/hosts -U git://my.git.server/ansible/ install-workstation.yml**



Toute la configuration « labo » se fait avec ansible

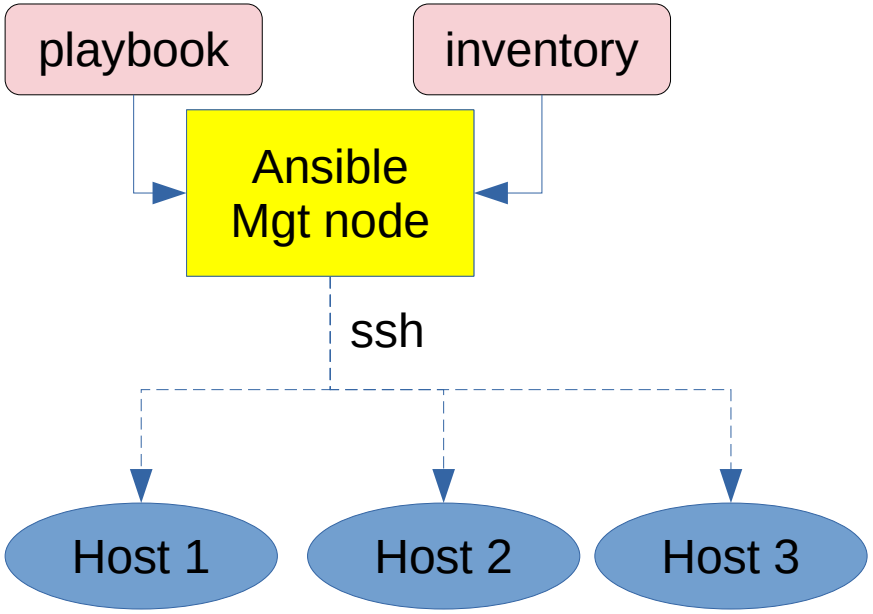
Ansible

- Créé ~2012 par Michael DeHaan (ex Cobbler, Puppet,…)
- Simplifier l'automatisation (proche des commandes des sysadmins)
- Syntaxe simple (yaml)
- Administration distante via ssh → pas de client à installer, juste une clé ssh
- Courbe d'apprentissage rapide
- Tâches doivent être « idempotentes » (i.e. même résultat si on les exécutes plusieurs fois)

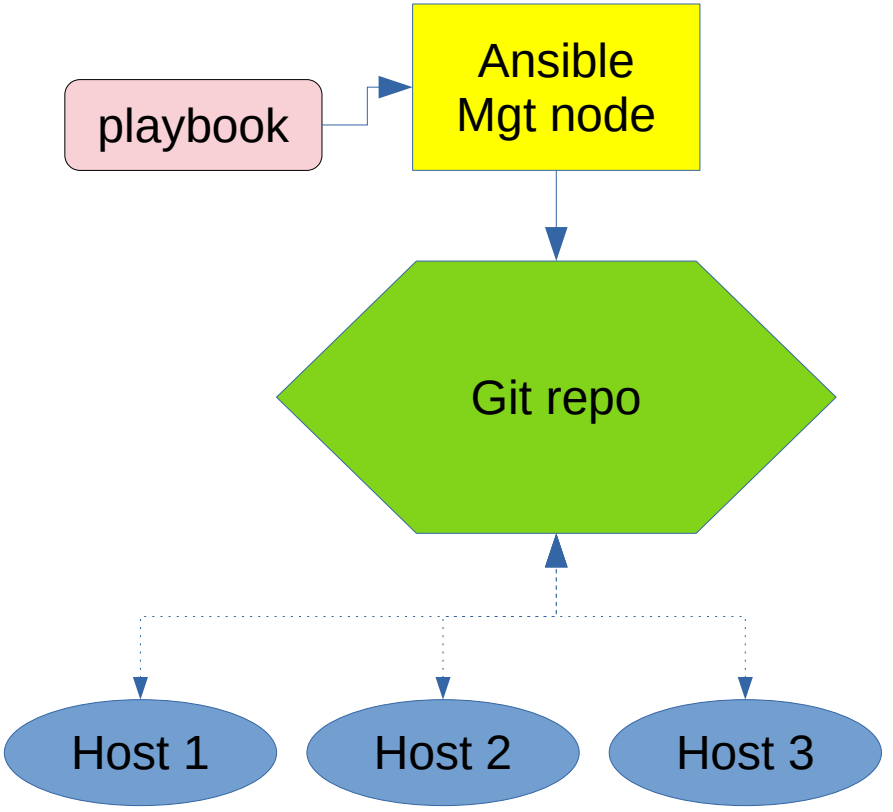
Ansible

- 2 types d'utilisation
  - Ligne de commande. ex : `ansible messerveurs -m reboot`
  - Playbook : fichier « recette » pour exécuter des commandes sur une machine. ex : `ansible-playbook configure_ldap.yml`
- 2 modes de fonctionnement
  - Mode push : serveur → machine cible
  - Mode pull : machine cible ← playbook (git) ⇒ exécution locale (nécessite ansible sur la machine cible)

ANSIBLE PUSH



ANSIBLE PULL



Ansible:inventory

```
localhost
monserveur.in2p3.fr

[docker_servers]
docker01.in2p3.fr
docker02.in2p3.fr

[ldap_servers]
ldap1.in2p3.fr

[critical_servers:children]
ldap_servers
storage_servers
```

Ansible:playbook

```
- hosts: localhost:monserveur
  become: yes

tasks:
- name: Install fusioninventory-agent for Debian-like distros
  apt:
    name: "fusioninventory-agent"
    state: present
    update_cache: yes
  ignore_errors: yes

- name: setup configuration server
  lineinfile:
    create: yes
    path: /etc/fusioninventory/agent.cfg
    line: "server = https://fi-server.in2p3.fr/glpi/plugins/fusioninventory/"
    regexp: '^(#\s*)?server\s*=\s*'
```

Ansible : rôles

- Possibilité de définir des « tasks » et de les appeler dans un playbook → roles

```
- hosts: linux_servers:!autreserveur.in2p3.fr
  become: yes

  roles:
    - fusioninventory
    - rkhunter
```

Ansible : mode pull

- Tirer les playbooks d’un git et les exécuter localement
  - `ansible-pull -d /tmp/ansible-pull –purge -i /tmp/ansible-pull/hosts -U git://my.git.server/ansible/ install-workstation.yml`
- Fichier « hosts »
  - `localhost ansible_connection=local`
- Par défaut exécute le playbook « local.yml »
- Adapté à l’installation et à la maintenance des workstations

Ansible : install-workstations.yml

```
---
- hosts: localhost
  vars:
    fusion_inventory_tag: "workstation"
  roles:
# general roles valids for all LPNHE machines (laptop, workstation,...)
    - lpnhe_user
    - packages_repositories
    - packages_base
    - sshkeys
    - passwords
    - lightdm
    - fusioninventory
    - rkhunter
# roles valids for workstations (not for laptop)
    - cvmfs_client
    - ansible_pull
    - ldap_client
    - homedir
    - cups_printers
```

Ansible : maintenance mode

- Cron qui lance ansible-pull à chaque boot et 1 fois par jour → execute local.yml qui contient éventuellement des maj de configs, déploiements, etc...

```
  - name: add cron at boot for ansible
    cron:
      cron_file: ansible-pull-mode
      state: present
      user: root
      special_time: reboot
      name: ansible_reboot
      job: ansible-pull -d /tmp/ansible-pull --purge -i
/tmp/ansible-pull/hosts -U git://my.git.server/ansible/ >> /var/log/ansible-
pull.log 2>&1
```

Ansible : local.yml

```
---
- hosts: localhost
  roles:
    - sshkeys
    - passwords
    - fusioninventory
    - sudoers
    - packages_base
  tasks:
  - { include: change_mount_path.yml }
```

Conclusion

- Il est relativement simple et rapide de gérer un parc avec Cloudinit et Ansible :
  - Installation avec Cloudinit
  - Configuration avec Ansible
  - Maintenance avec Ansible
- La modularité d’ansible permet de créer différents scénarios (workstations, laptop, bancs de test,...)

cloud-init

BACKUP

Fichier PXE

```
LABEL Ubuntu cloud init 20.04 workstation
    kernel http://xxx.xxx.xxx.xxx/DISTRIBS/ubuntu/20.04/AMD64/vmlinuz
    append ramdisk_size=1500000 ip=dhcp  url=http://xxx.xxx.xxx.xxx/DISTRIBS/ubuntu/20.04/AMD64/ubuntu-20.04.3-live-server-amd64.iso
autoinstall ds=nocloud-net;s=http://xxx.xxx.xxx.xxx/DISTRIBS/ubuntu/20.04/AMD64/CLOUDININT/
cloud-config-url=http://xxx.xxx.xxx.xxx/DISTRIBS/ubuntu/20.04/AMD64/CLOUDININT/workstation root=/dev/ram0
    initrd http://xxx.xxx.xxx.xxx/DISTRIBS/ubuntu/20.04/AMD64/initrd

LABEL Ubuntu cloud init 20.04 banc de test
    kernel http://xxx.xxx.xxx.xxx/DISTRIBS/ubuntu/20.04/AMD64/vmlinuz
    append ramdisk_size=1500000 ip=dhcp  url=http://xxx.xxx.xxx.xxx/DISTRIBS/ubuntu/20.04/AMD64/ubuntu-20.04.3-live-server-amd64.iso
autoinstall ds=nocloud-net;s=http://xxx.xxx.xxx.xxx/DISTRIBS/ubuntu/20.04/AMD64/CLOUDININT/
cloud-config-url=http://xxx.xxx.xxx.xxx/DISTRIBS/ubuntu/20.04/AMD64/CLOUDININT/banc root=/dev/ram0
    initrd http://xxx.xxx.xxx.xxx/DISTRIBS/ubuntu/20.04/AMD64/initrd

LABEL Ubuntu cloud init 20.04 laptop
    kernel http://xxx.xxx.xxx.xxx/DISTRIBS/ubuntu/20.04/AMD64/20.04.4/hwe-vmlinuz
    append  ramdisk_size=1500000 ip=dhcp  url=http://xxx.xxx.xxx.xxx/DISTRIBS/ubuntu/20.04/AMD64/20.04.4/ubuntu-20.04.4-live-server-
amd64.iso autoinstall ds=nocloud-net;s=http://xxx.xxx.xxx.xxx/DISTRIBS/ubuntu/20.04/AMD64/CLOUDININT/
cloud-config-url=http://xxx.xxx.xxx.xxx/DISTRIBS/ubuntu/20.04/AMD64/CLOUDININT/laptop root=/dev/ram0
    initrd http://xxx.xxx.xxx.xxx/DISTRIBS/ubuntu/20.04/AMD64/20.04.4/hwe-initrd-new
```

Fichier Cloudinit Workstation

```
#cloud-config
merge_how:
  - name: list
    settings: [append]
  - name: dict
    settings: [no_replace, recurse_list]
autoinstall:
  version: 1
  interactive-sections:
    - keyboard
  apt:
    geoip: true
    preserve_sources_list: false
    primary:
    - arches: [amd64, i386]
      uri: http://www-ftp.lip6.fr/pub/linux/distributions/Ubuntu/archive
      search:
        - http://www-ftp.lip6.fr/pub/linux/distributions/Ubuntu/archive
        - http://fr.archive.ubuntu.com/ubuntu
    - arches: [default]
      uri: http://ports.ubuntu.com/ubuntu-ports
  user-data:
    users:
      - name: toto
        gecos: TOTO
        sudo: ALL=(ALL) NOPASSWD:ALL
        primary_group: root
        groups: [users, admin, sudo, adm]
        password: $6$uRM7eJgwOrj.sm2c$d2rFaiTChiErMacRonuFPEOxTkU/NBoRneAUssiPR./
        homedir: /home-local/toto
        shell: /bin/bash
        lock_passwd: false
```

Fichier Cloudinit Workstation (suite)

```
keyboard: {layout: us, variant: ''}
  locale: fr_FR.UTF-8
  timezone: Europe/Paris
  ssh:
    allow-pw: true
    authorized-keys: []
    install-server: true
  packages:
    - ansible
    - apt-transport-https
    - git
    - openssh-client
    - python3
    - ssh
    - sudo
    - vim
    - wget
  updates: security
  package_update: true
  late-commands:
    - curtin in-target --target=/target -- lvextend -l +100%FREE /dev/mapper/ubuntu--vg-ubuntu--lv
    - curtin in-target --target=/target -- resize2fs /dev/mapper/ubuntu--vg-ubuntu--lv
    - curtin in-target --target=/target -- ansible-pull -d /tmp/ansible-pull --purge -i /tmp/ansible-pull/hosts -U git://gitserver.in2p3.fr/ansible/ install-
workstation.yml >
> /target/var/log/installer/ansible.log
    - wget http://installserver.in2p3.fr/set_localboot.php
    - wget http://installserver.in2p3.fr/add_to_ansible.php?computertype=ubuntu-amd64-ws
```

Exemple de rôle pour cvmfs

```
 - name: install lsb-release
   apt:
     name: lsb-release
     update_cache: yes
     state: present
   when: ansible_os_family == 'Debian'
   ignore_errors: yes

 - name: add cvmfs repo
   yum:
     name: https://ecsft.cern.ch/dist/cvmfs/cvmfs-release/cvmfs-release-latest.noarch.rpm
     state: present
   when: ansible_os_family == 'RedHat'
   ignore_errors: yes

 - name: add cvmfs repo for debian
   apt:
     deb: https://ecsft.cern.ch/dist/cvmfs/cvmfs-release/cvmfs-release-latest_all.deb
     update_cache: yes
   when: ansible_os_family == 'Debian'
   ignore_errors: yes

 - name: install cvmfs-release for debian
   apt:
     name: cvmfs-release
     state: present
     update_cache: yes
   when: ansible_os_family == 'Debian'
   ignore_errors: yes

- name: install cvmfs
   package:
     name:
       - cvmfs
       - cvmfs-config-default
       - bc
     state: present
   ignore_errors: yes

 - name: initial setup
   shell: cvmfs_config setup
   ignore_errors: yes

   - name: available size
   shell: df -lP  /var/ |tail -n 1 |awk '{print $4 "/1000"} '|bc
   register: df
   ignore_errors: yes

 - name: cache size
   shell: cvmfs_talk cache size |grep "cache size"|tail -n 1|awk '{ print substr($5,0,length($5) - 2)}'
   register: cached
   ignore_errors: yes

 - set_fact:
     mycache: "{{  ((df.stdout |int + cached.stdout |int)/3) |int }}"

 - name: push default.local
   template:
     src: default.local.j2
     dest: /etc/cvmfs/default.local
     owner: root
     group: root
     mode: 0644
   ignore_errors: yes

 - name: push hess config
   copy:
     src: sw.hess-experiment.eu.conf
     dest: /etc/cvmfs/config.d/sw.hess-experiment.eu.conf
     owner: root
     group: root
     mode: 0644
   ignore_errors: yes

 - name: create /etc/cvmfs/keys/sw.hess-experiment.eu
   file:
     path: /etc/cvmfs/keys/sw.hess-experiment.eu
     state: directory
```

Exemple de rôle pour cvmfs

```
- name: push hess key
  copy:
    src: sw.hess-experiment.eu.pub
    dest: /etc/cvmfs/keys/sw.hess-experiment.eu/sw.hess-experiment.eu.pub
    owner: root
    group: root
    mode: 0644
  ignore_errors: yes

- name: restart autofs
  service:
    name: autofs
    state: restarted
  ignore_errors: yes

- name: check mount
  command: cvmfs_config probe
  ignore_errors: yes
```