FORMATION ANSIBLE

(2 jours)

Pour

MODE D'EMPLOI PRÉSENTATION

ESPACE = slide suivante

FLECHE DIRECTIONNELLE = se déplacer dans les slides

ECHAP = mode plan miniatures

ajout ?print-pdf dans l'URL = version imprimable/exportable en PDF

Source : https://github.com/hakimel/reveal.js/

Démo : https://revealjs.com/#/

THÉORIE

DÉFINITION

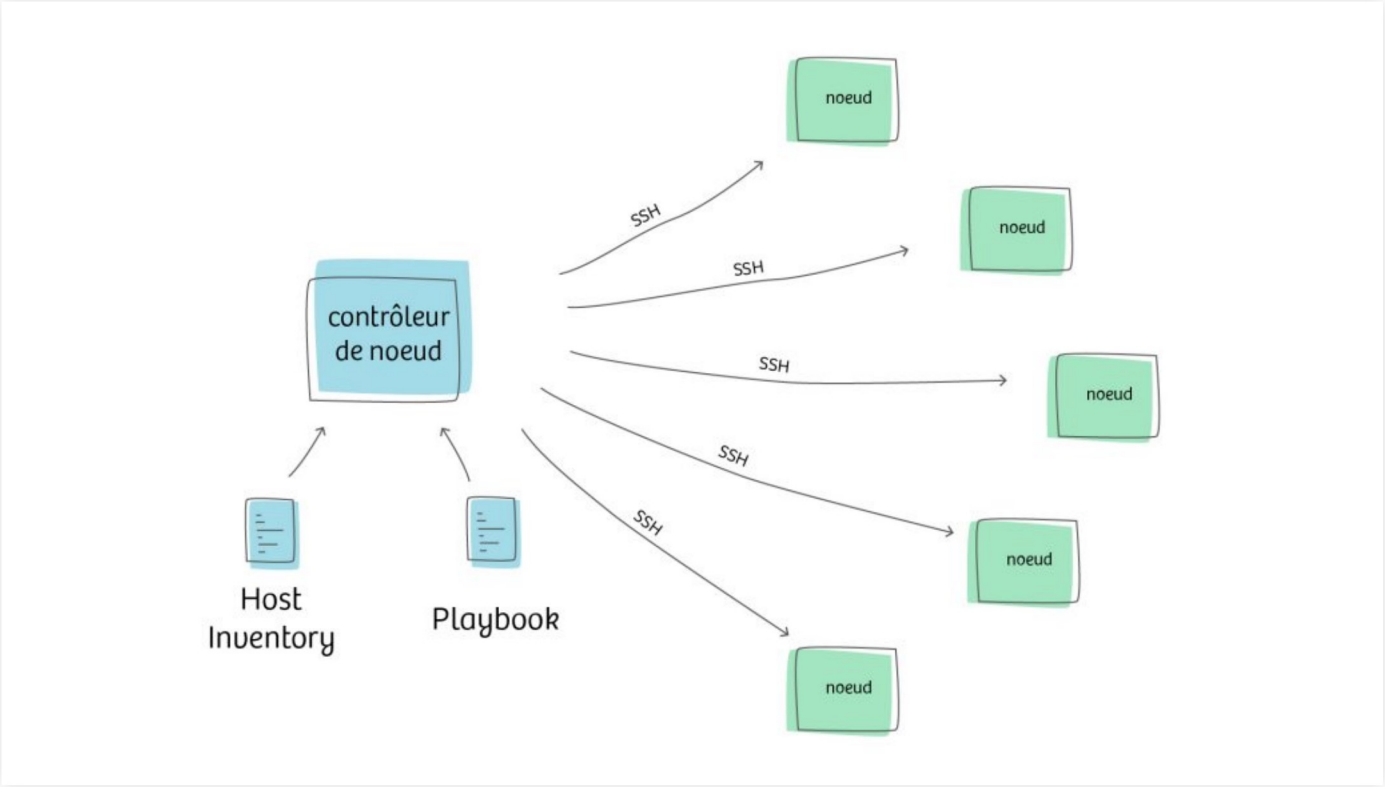
Plateforme logicielle libre pour la configuration et la gestion d'ordinateurs.

- Déploiement de logiciels multi-noeuds
- Exécution de tâches ad-hoc (pas d'intermédiaire)
- Gestion de configuration (jinja2)
- Agentless (SSH)
- Retour en json
- Ecriture en yaml
- Licence GPL (et propriétaire RedHat en partie)
- Projet mature et maintenu depuis 2012 (rachat en 2015 par RedHat)

A QUOI ÇA SERT ?

- Infrastructure as code
- Déploiement aisé sur tous types d'environnement (on premise, cloud public ou privé, local)
- Compatibilité avec tous matériels gérant SSH (serveurs, routeurs, switch, etc...)
- Intégration facile dans la CI/CD
- Possibilité avec des UI de faire du "devops à la demande self service" (tower & awx)
- Industrialiser les infrastructures et déploiements
- Eviter les erreurs humaines

CA MARCHE COMMENT ?



VOCABULAIRE

INVENTAIRE

Liste de machines à controler par ansible, avec possibilité de groupe et variables propres à des machines ou groupes de machines

Ils peuvent être alimenter via des scripts les rendant dynamique (exemple inventaire dynamique hyperviseur proxmox) ou encore coupler à terraform ou autres outils de type provisionning de cloud public

Exemple

```
[client1]
ubuntu ansible_host=172.16.123.129 ansible_connection=ssh ansible_ssh_user=root a
[client2]
debian ansible_host=172.16.123.132 ansible_connection=ssh ansible_ssh_user=root a

[production:children]
client1
client2

[client1:vars]
project_name=myblog
host_name=myblog.localdomain.fr
host_alias=["www.myblog.fr", "myblog.fr"]
cms=wordpress
mysql_host=localhost
mysql_login=client1
mysql_password=toor
cms_password=admin

[client2:vars]
```

ROLES

Se compose de tâches, permettant différentes actions (installation, configuration, déploiement de templates,...) sur les noeuds managés via ansible

Bonnes pratiques les rôles doivent être idempotents (pouvoir être rejouer x fois et donner le même résultat), indépendants, testables, réutilisables

Exemple roles/common/task.yml

```
---
# tasks file for common

- name: Update cache apt repo
  apt: update_cache=yes cache_valid_time=3600

- name: Upgrade packages
  apt: upgrade=safe

- name: Install common packages
  apt:
    pkg:
      - git
      - htop
      - curl
      - unzip
      - iotop
      - iftop
      - sudo
      - screen
```

PLAYBOOK

Il va permettre d'appeler différents **rôles** à appliquer à certaines machines ou totalité de l'inventaire

Exemple main.yml

```
---
- name: "This playbook will setup {{ cms }} and LAMP Stack required and will be accessible
  hosts: all
  # vars_files:
  #   - vars/variables.yml

  roles:
    - role: wordpress
      when: cms == "wordpress"
    - role: prestashop
      tasks:
        - debug:
            msg: "Prestashop BO Credentials are \n login : {{ prestashop_email }}\n password
      when: cms == "prestashop"
```

LES VARIABLES

Elle vont pouvoir être défini à différents endroits (inventaire, playbook, via saisie utilisateur, par défaut avec un role, etc...)

Elles vont surtout permettre de rendre nos rôles et playbook plus ou moins prêt à l'emploi.

Les bonnes pratiques veulent qu'un playbook soit idempotent au même titre que les rôles qui le compose, si une tâche copie un fichier,

on peut choisir dans la tâche de ne le copier que s'il a été modifié ou n'est pas déjà présent, rendant ainsi une seconde exécution "idempotente" elle ne fera aucuns changements si ce n'est pas nécessaire.

MODULES

Ils sont nombreux à être natifs à ansible afin d'intéragir avec différents systèmes et applicatifs, on retrouve par exemple le module "apt" pour gérer les installations de paquet sur debian.

Mais aussi des fois des modules plus génériques tel que "package" au lieu "apt" qui sera compatible avec potentiellement plus de système mais permettra moins d'options

```
- name: install ntpdate with package module
  package:
    name: ntpdate
    state: present

- name: install ntpdate with apt module (only debian/ubuntu)
  apt:
    name: ntpdate
    state: present
```

Il en existe en plus des "officiels et natifs" (voir github) des maintenu par la communauté et sont totalement open-source, écrit en python (donc ce qui ne sait pas encore parlé avec ansible mais sait parlé en SSH peut être interfacé avec ansible !)

DOCUMENTATION OFFICIELLE

https://docs.ansible.com/ansible/latest/

DOCUMENTATION DES MODULES

La documentation officielle d'ansible est très bien fournie, pleines d'exemples et cas concrets d'utilisation

https://docs.ansible.com/ansible/2.9/modules/list_of_all_modules.html

https://docs.ansible.com/ansible/2.9/modules/modules_by_category.html

https://docs.ansible.com/ansible/2.9/modules/apt_module.html#apt-module

ARBORESCENCE D'UN PROJET TYPE

Exemple

```
├── defaults
│   └── main.yml
├── group_vars
│   ├── all.yml
│   ├── dev.yml
│   ├── preprod.yml
│   └── production.yml
├── inventory
├── main.yml
├── roles
│   ├── apache
│   │   └── tasks
│   │       └── main.yml
│   ├── common
│   │   ├── tasks
│   │   │   └── main.yml
│   │   └── templates
│   │       ├── etc
│   │       │   └── bash.bashrc
│   │       └── public_keys
│   │           ├── eg_deploy_id25519.pub
│   │           └── eg_jtoutain_id25519.pub
```

BONNES PRATIQUES

- Rester le plus simple possible
- Nommer les tâches
- Découper de façon logique de façon à ne pas avoir de "doublon" de code ou tâches
- Versionner vos IaC (Infrastructure as Code) sur des dépôts de code type git
- Tester vos rôles avec molecule (et vagrant ou docker)
- Plus d'infos ici

OUTILS ANNEXES

ANSIBLE GALAXY

https://galaxy.ansible.com/

https://medium.com/@v.brissat/ansible-molecule-devops-part-1-b49ee97fe404

https://medium.com/@v.brissat/ansible-molecule-devops-part-2-1ad1d9f4b3fe

https://medium.com/@v.brissat/ansible-molecule-devops-part-3-60ddbe147e97

VAGRANT & VIRTUALBOX & DOCKER

https://www.vagrantup.com/docs

https://www.vagrantup.com

https://www.packer.io/docs/builders/virtualbox

https://www.packer.io/docs/builders/vagrant

https://www.packer.io/docs/builders/docker

PACKER

https://www.packer.io/docs/provisioners/ansible

https://www.packer.io/docs/provisioners/ansible-local

https://www.packer.io/docs/builders/vmware

MOLECULE

https://molecule.readthedocs.io/en/latest/

https://medium.com/@v.brissat/ansible-molecule-devops-part-3-60ddbe147e97

AWX

https://github.com/ansible/awx

PRATIQUE

TP 1 : DÉPLOYER ET CONFIGURER UNE INFRASTRUCTURE POUR UN PROJET WEB

PRÉPARATIFS

- Machine virtuelle debian 10.x server x3 (prévoir un template avec clé ssh intégré à cloner)
- Une machine avec ansible installé (prérequis : python) pour gérer des noeuds debian

INSTALLER ANSIBLE SUR VOTRE MACHINE CLIENTE (VM OU PHYSIQUE)

Microsoft lover's

https://docs.ansible.com/ansible/latest/user_guide/windows_faq.html#can-ansible-run-on-windows

Pour les linuxiens

```
pip install ansible
```

CAHIER DES CHARGES

Sur 3 machines debian 10 sans interface graphique vous installer

- sur la première nommé "infra1" (minimum 4Go de RAM) :
  - Docker
  - Gitlab
  - Bonus : AWX
- sur la seconde nommé "web1" :
  - un serveur web (au choix apache/nginx)
  - PHP 7.x
  - 2 sites web dont un CMS au choix (wordpress, prestashop, autres ?)
- sur la troisème nommé "db1":
  - mariadb (base de donnée pour le CMS au minimum)
  - utilisateur et base de donnée créée via ansible

CAHIER DES CHARGES

Le travail d'écriture des playbooks pourra se décomposer de la sorte (à titre indicatif et non exhaustif il faudra sûrement compléter la liste) :

```
* Infrastructure
    * Rôles appliqué à la machine "infra" : docker, gitlab, (awx ?)

* Web
    * Rôles : serveur web (apache/nginx), php, CMS et site web, autres modules requis pour le CMS (c
* Base de donnée
    * Rôles : mariadb
```

Note : Il peut être intéressant d'utiliser un rôle "common", déployer sur votre machine contenant des actions communes à tous (mise à jour dépot apt, mise à jour paquets, installation utilitaires type open-vm-tools)

Autres points à respecter :

- Le CMS devra avoir un plugin ou thème qui se déploie automatiquement depuis un dépôt sur votre gitlab.
- Le code yaml des playbooks et rôles ansible écrits doit être "propre" afin d'être réutilisable et idempotent (pas d'action effectué en double si déjà faites une première fois)
- Le code ansible devra envoyé sur git (github.com ou gitlab.com ou bitbucket & co...) et être partager avec l'intervenant avant la fin du TP