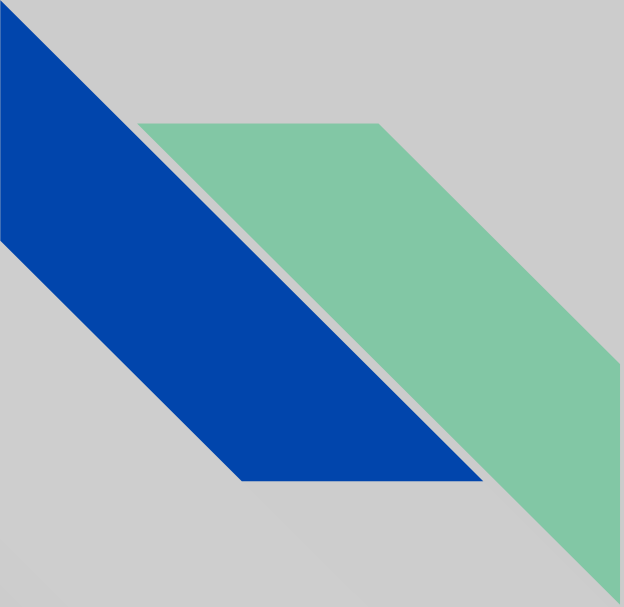


ANSIBLE

Ansible : L’outils d’orchestration agentless

Sommaire

I. Le problème
II. Le besoin
III. 2 approches, 2 technologies
IV. Ansible
V. Le principe d’idempotence
VI. Les composants
VII. Le fonctionnement
VIII. La structure des dossiers
IX. (+ Démonstration)
X. Conclusion

I Le Problème

- Intervention manuelle et individuelle sur chaque serveur
- Développement rapide du cloud (+ de serveurs)
- Applicatif de plus en plus complexe
- émergence du développement agile
- Le CI/CD et le DevOps

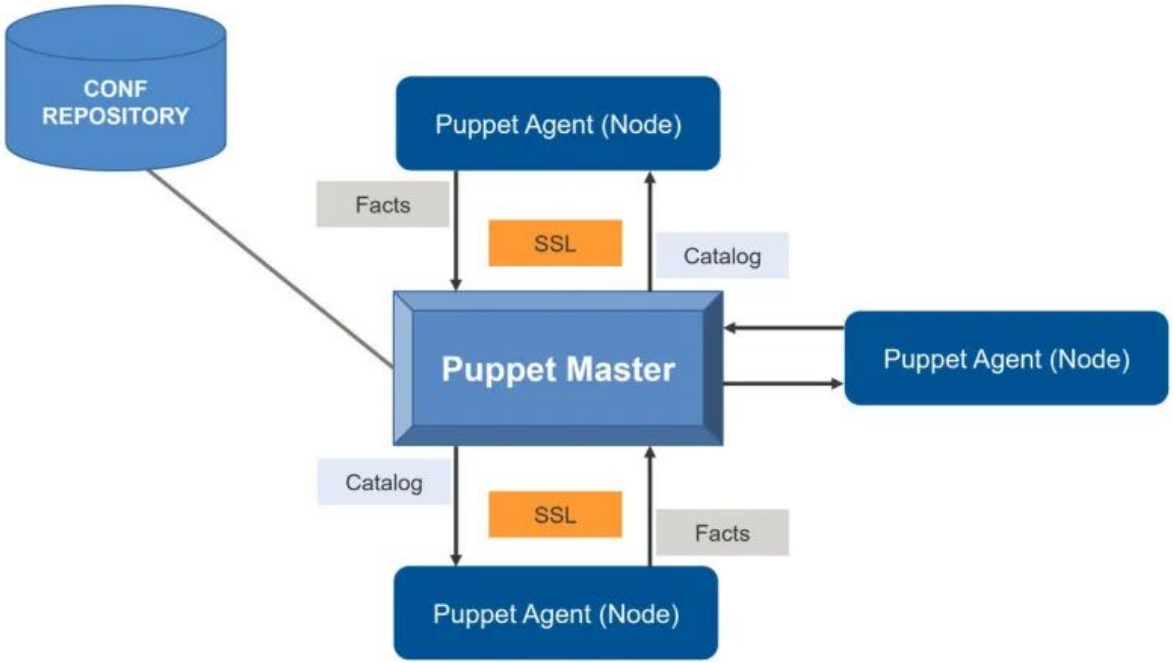
II Le besoin

- Des outils de provisionnement
- Des outils de gestion de configuration
- Des outils d’orchestration

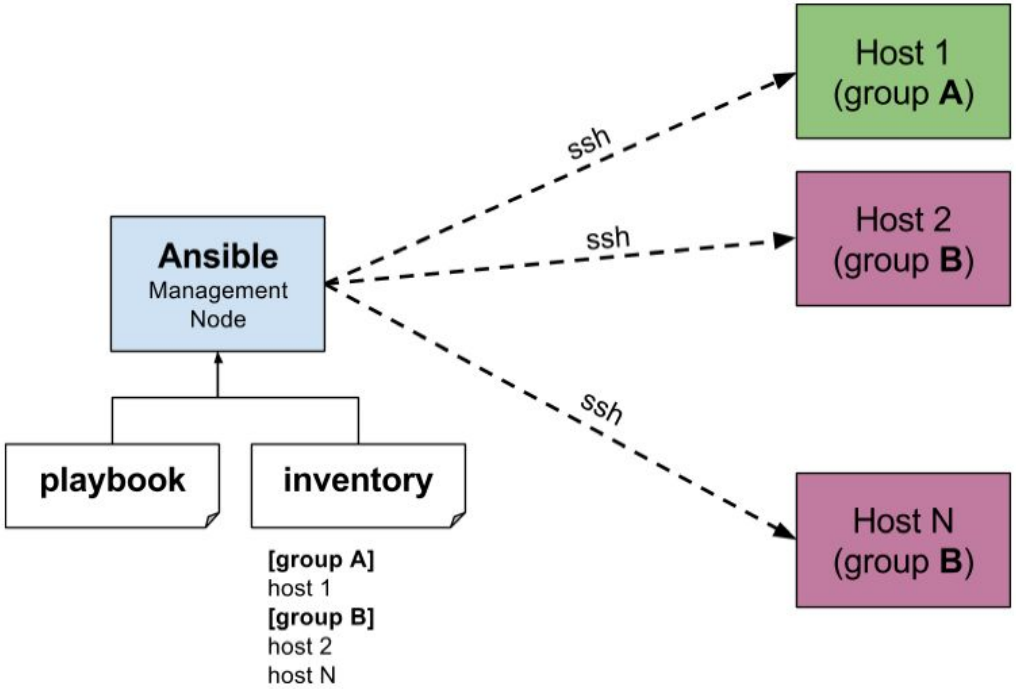
III 2 approches, 2 technologies

- Pull (agent) : puppet
- Push (agentless) : Ansible

III 2 approches, 2 technologies



III 2 approches, 2 technologies

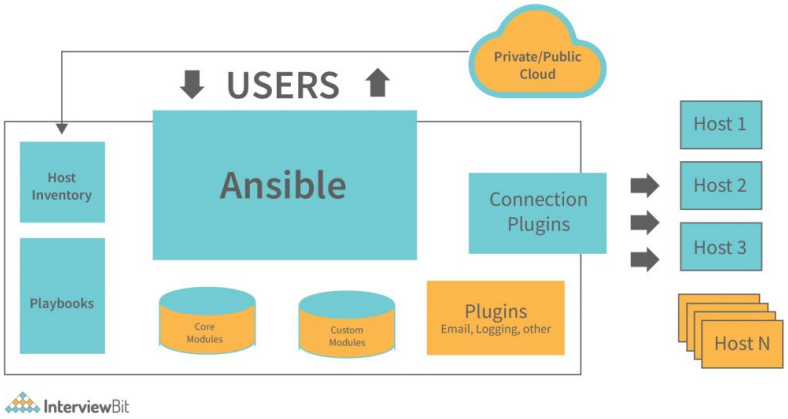


IV Ansible

- Créé en 2012, il y a 10 ans
- version 2.14 (11/22)
- Développé par Red Hat
- Écrit en python
- IaC : Provisionnement, (gestion de conf), orchestration

Avantages : Gratuit (OpenSource), Simple, Puissant, AgentLess, Cohérent, Sécurisé, 3 en 1

V Les composants



- Inventory
- Variable
- Module
- Task
- Role
- Playbook
- Handlers
- Plugin
- Collection

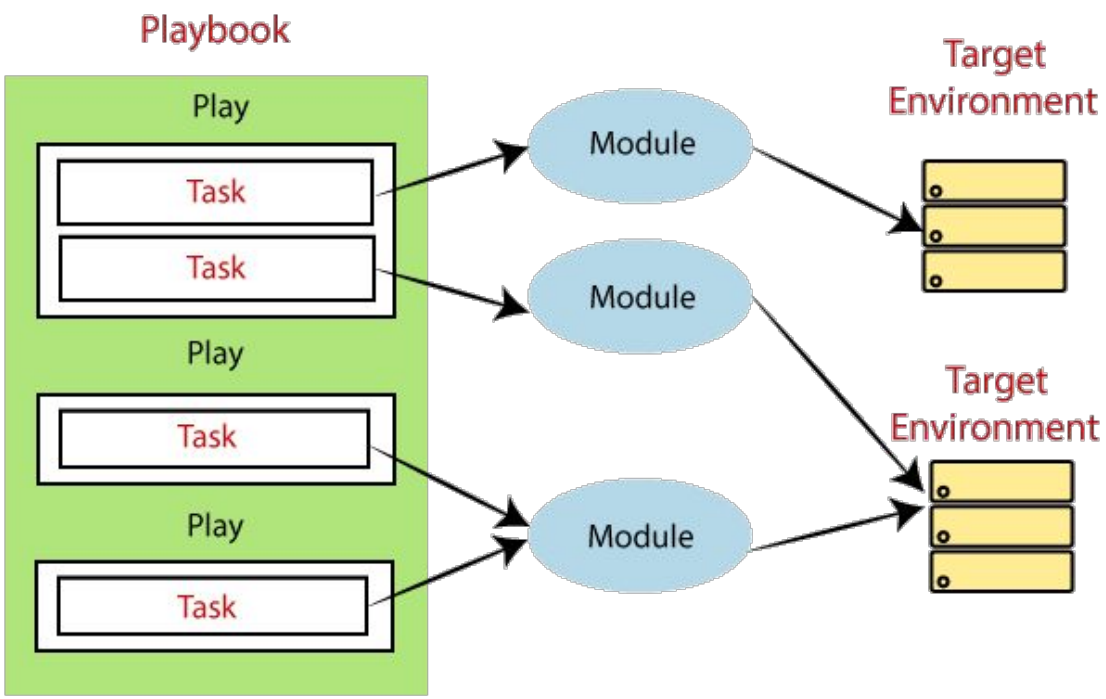
VI La structure des dossiers

```
production                # inventory file for production servers
staging                   # inventory file for staging environment

group_vars/
   group1.yml             # here we assign variables to particular groups
   group2.yml
host_vars/
   hostname1.yml          # here we assign variables to particular systems
   hostname2.yml

library/                  # if any custom modules, put them here (optional)
module_utils/             # if any custom module_utils to support modules, put them here (optional)
filter_plugins/           # if any custom filter plugins, put them here (optional)

site.yml                  # master playbook
webservers.yml            # playbook for webserver tier
dbservers.yml             # playbook for dbserver tier

roles/
    common/               # this hierarchy represents a "role"
        tasks/            #
            main.yml      #  <-- tasks file can include smaller files if warranted
        handlers/         #
            main.yml      #  <-- handlers file
        templates/        #  <-- files for use with the template resource
            ntp.conf.j2   #  <------- templates end in .j2
        files/            #
            bar.txt       #  <-- files for use with the copy resource
            foo.sh        #  <-- script files for use with the script resource
        vars/             #
            main.yml      #  <-- variables associated with this role
        defaults/         #
            main.yml      #  <-- default lower priority variables for this role
        meta/             #
            main.yml      #  <-- role dependencies
        library/          # roles can also include custom modules
        module_utils/     # roles can also include custom module_utils
        lookup_plugins/   # or other types of plugins, like lookup in this case

    webtier/              # same kind of structure as "common" was above, done for the webtier role
    monitoring/           # ""
    fooapp/               # ""
```

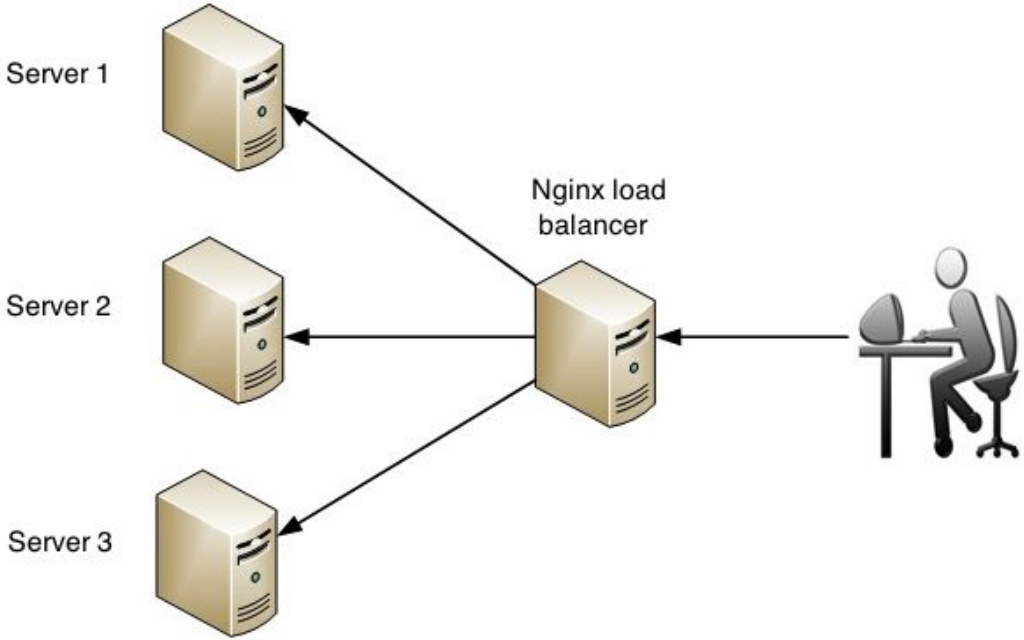
VII Le principe d’idempotence

“En mathématiques et en informatique, l'**idempotence** signifie qu'une opération a le même effet qu'on l'applique une ou plusieurs fois. “ Source : Wikipedia

VIII Le fonctionnement



IX Démonstration



X Conclusion

- Indispensable aujourd’hui
- Adapté au structure “moyenne”