Retour d'expérience comparatif Ansible & Puppet

Benoit Métrot (*benoit.metrot@univ-poitiers.fr)*

UPR 3346 - Institut PPrime (Poitiers)

Rencontre Mathrice - 2 Octobre 2019

Plan

Progression

Qu'est-ce qu'un gestionnaire de configuration ?

Definition

C'est un outil qui décrit la configuration système d'une machine. Il s'assure que chaque machine administrée est conforme à ce qui est mentionné dans la description. Il garantit le suivi et le maintien de cette configuration dans le temps.

Pourquoi utiliser un gestionnaire de configuration ?

- **Reproductibilité** Réinstaller une machine à l'identique.
- **Industrialisation** Déployer sur un grand nombre de machine.
- **Fiabilité** Toutes les machines d'une même série sont configurées de la même façon. Elles reçoivent toutes les mêmes ajustements.
- **Documentation** Décrit de façon exhaustive toutes les opérations nécessaire à sa configuration.

mathrice

Ansible : Contexte d'utilisation

- Départements d'enseignement du campus Poitiers Futuroscope (SP2MI)
  - 5 Unités de Formations (UF)
  - Infrastructure mutualisée autour d'un domaine ActiveDirectory
  - 2500 utilisateurs
  - 400 machines dont 230 en double amorçage
  - Quelques serveurs sous GNU/Linux (stockage, dhcp, web)

→ Installation automatique Ubuntu 18.04 et configuration par Ansible

mathrice

Puppet : Contexte d'utilisation

- Participation à l'administration de la PLM
  - Ensemble de plusieurs dizaines de machines virtuelles
  - Réparties sur plusieurs sites géographiques
  - Une équipe d'une douzaine de personnes

- Laboratoire de Mathématiques et Applications (Poitiers)
  - 20 serveurs physiques
  - 20 machines virtuelles
  - 60 postes fixes
  - 95% de systèmes GNU/Linux

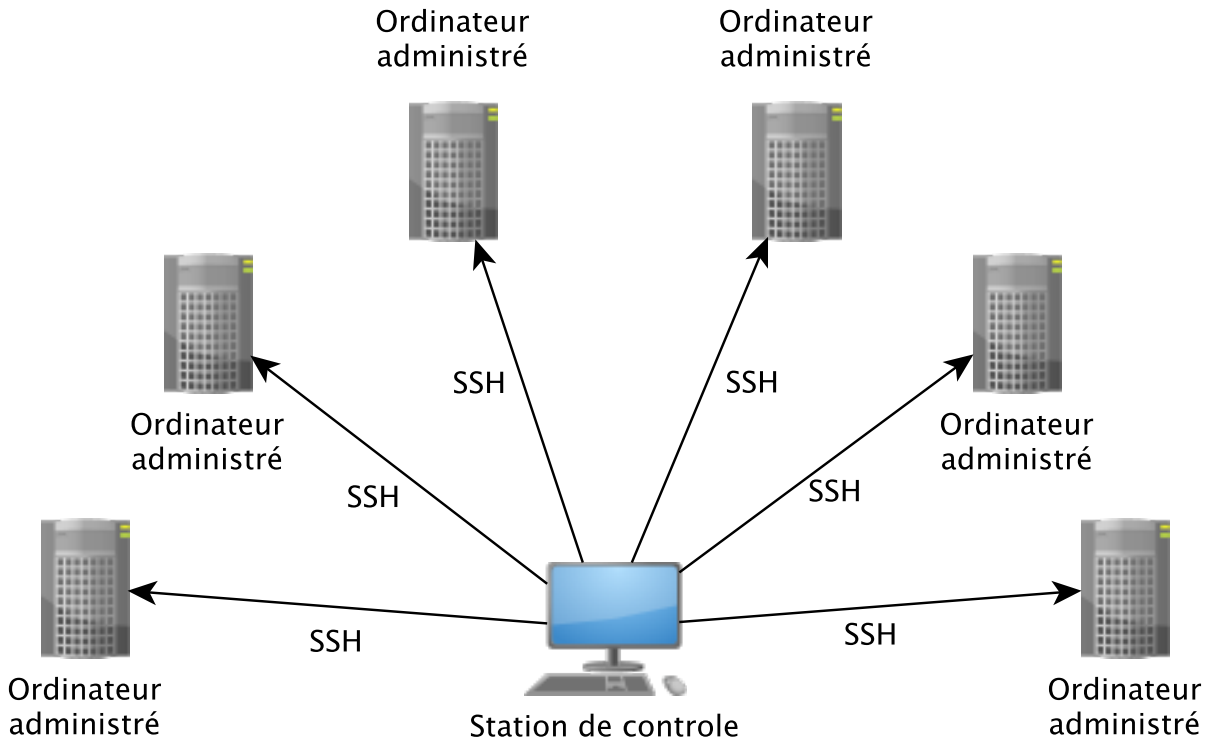
mathrice

D’autres outils existent...

- Cfengine
- Sprinkle
- Chef
- ...

Progression

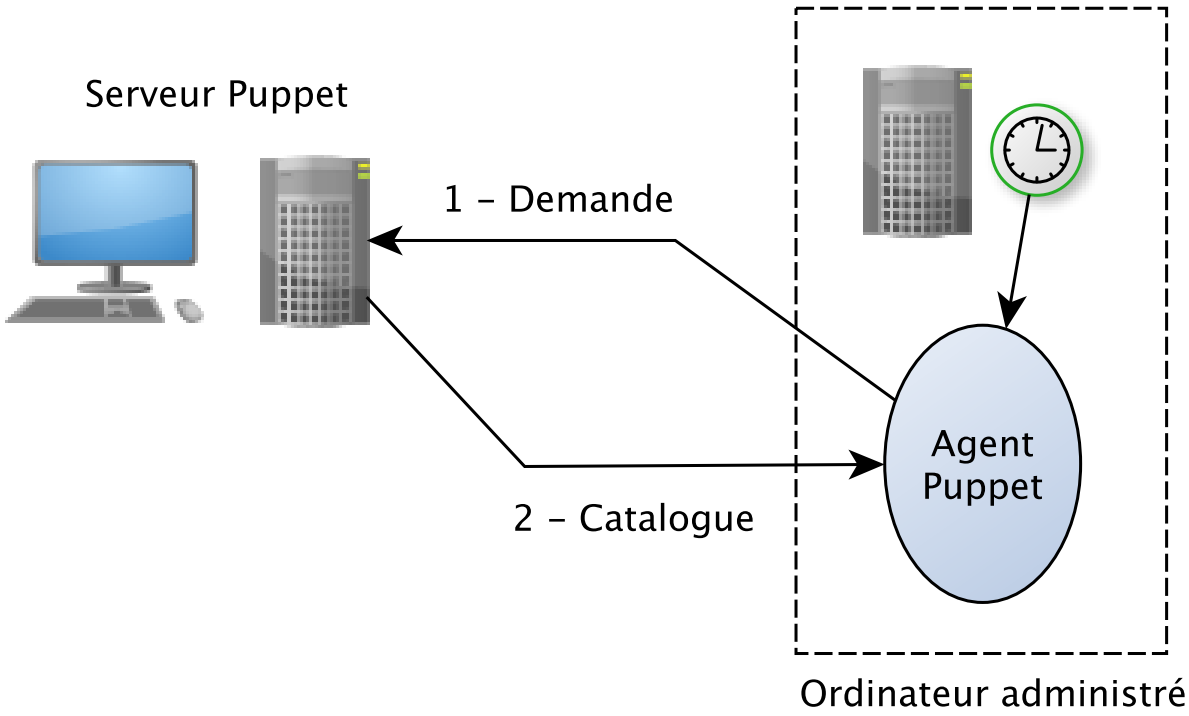
Ansible : Architecture



Ansible : Fonctionnalités

- Initiative de la station de controle (*Control Node*)
- Traitement par lot de plusieurs ordinateurs administrés (*Managed Node*)
- Utilisation du protocole SSH
- Mécanismes d'authentification supportés
  - Couple de clés (passphrase recommandé)
  - Mot de passe
  - Support de sudo
- Modèle sans agent sur l'ordinateur administré

Puppet : Architecture



Puppet : Fonctionnement

- Initiative de l'ordinateur administré (*Puppet Node*)
- Déclenchement manuel ou automatique (horloge)
- Authentification par infrastructure de clés sur le serveur Puppet (*Puppet Master*)
- Protocole de communication spécifique à Puppet (TCP/8140)
- Agent en exécution sur chaque ordinateur administré

Ansible : Pré-requis système

- Sur chaque ordinateur administré
  - Installer *python*
  - Disposer d'un compte avec privilèges (*sudo*)
  - Déposer sa clé publique SSH
- La station de controle doit pouvoir accéder à chaque ordinateur administré
- Installer *Ansible* sur la station de controle
  - Via les paquets de la distribution
  - Via l'outil *pip*
  - Via les paquets binaires proposés par *Ansible*
  - Via l'outil *virtualenv* (Méthode recommandée)

mathrice

Puppet : Pré-requis système

- Installer l'agent sur chaque ordinateur administré
- Valider le certificat de chaque nouvel agent sur le serveur
- Les ordinateurs administrés doivent accéder au port TCP 8140 du *puppet master*

Synthèse

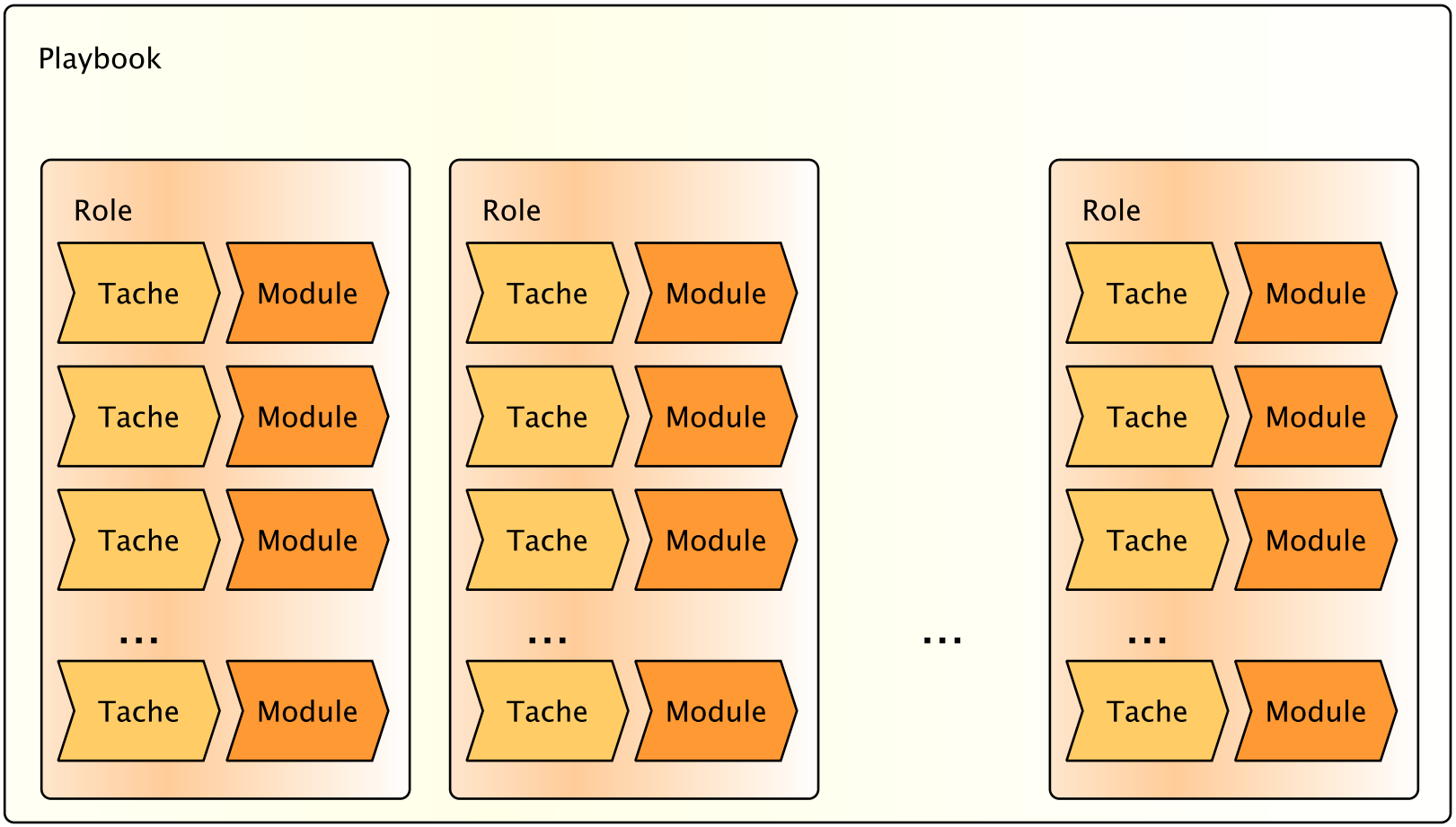
- Modèle sans agent
- Disposer d'un compte privilégié sur chaque ordinateur
- Initiative de la station de controle
- Adapté aux procédures *one-shot*

- Agent à déployer sur le parc
- Déclenchement à l'initiative de l'ordinateur administré (manuel ou automatique)
- Gestion des certificats clients

mathrice

Progression

Ansible : Les objets



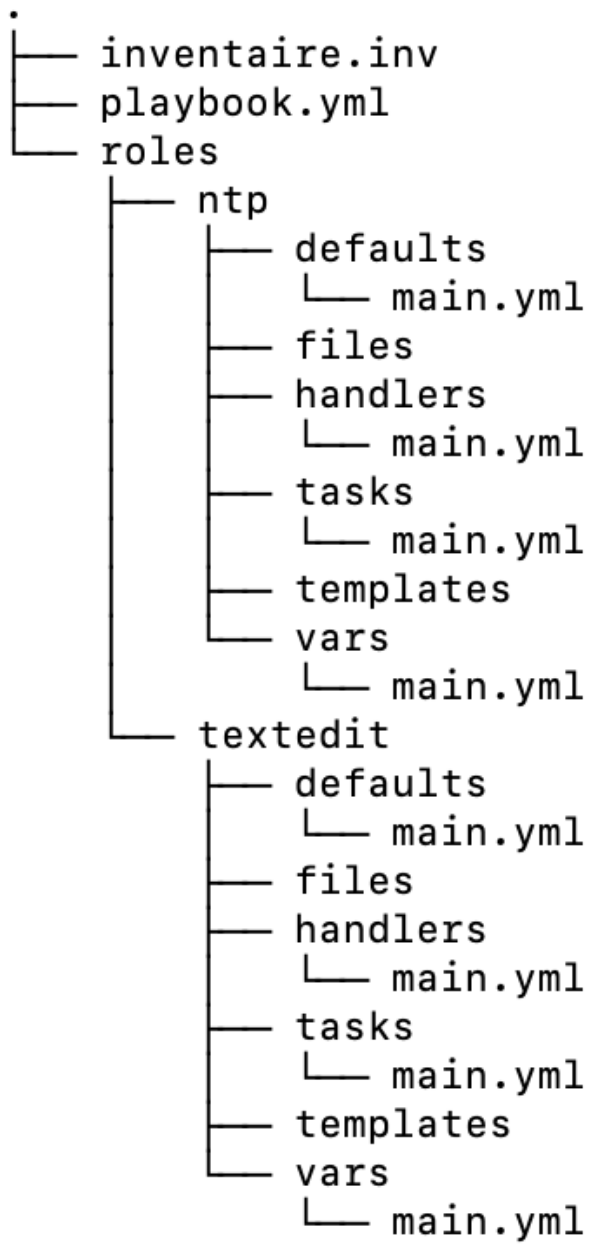
Ansible : Articulation des objets

- Le **Playbook** est le point d'entrée d'Ansible. Il sera appliqué par l'outil à une ou plusieurs machines.
- Au moins un **role** compose le playbook. Il représente la configuration d'un composant du système (un sgbd, un service web, la synchronisation NTP, le réseau)
- Le role aggrège une ou plusieurs **taches** (*task*). Son exécution est soumise éventuellemnt à la validation d'une condition.
- Une tache s'appuie sur un **module** pour appliquer une opération à un ordinateur. Des paramètres permettent de préciser l'opération.
  - module *apt* $\rightarrow$ installation d'un paquet (Debian / Ubuntu)
  - module *file* $\rightarrow$ création d'un répertoire, d'un lien, postionnement de droits
  - module *copy* $\rightarrow$ copie d'un fichier
  - module *lineinfile* $\rightarrow$ modification d'une ligne dans un fichier texte
  - ...

Ansible : Le formalisme

- Ensemble de fichiers écrit au format YAML
- Python est le langage sous-jacent
- Collecte d'informations depuis la machine cibles $\rightarrow$ variables
- Utilisation de filtres et de Tests Jinja pour l'interprétation des variables
- Les taches sont exécutés selon l'ordre où elles apparaissent

Ansible : Organisation des fichiers



Ansible : Exemple - inventaire.inv

```
[all:vars]
ansible_user=root
ansible_ssh_private_key_file=~/.ssh/id_ansible

[servers]
client1                 ansible_host=172.16.173.134
client2                 ansible_host=172.16.173.140
client3                 ansible_host=172.16.173.137
```

Ansible : Exemple - playbook.yml

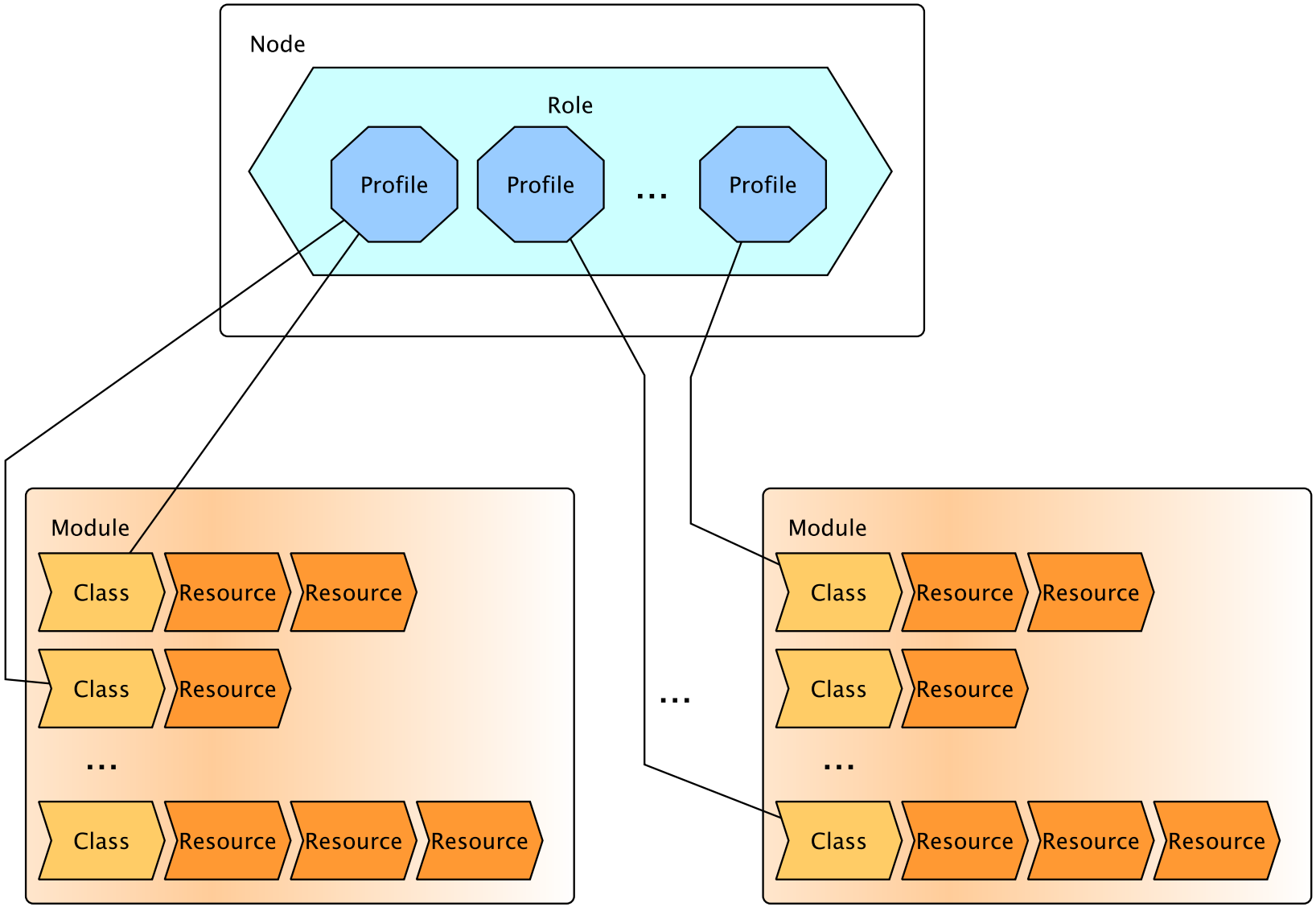
```
---

- name: "Playbook d'exemple"
  hosts:
    - "client1"
    - "client2"
    - "client3"
  roles:
    - role: "roles/ntp"
    - role: "roles/textedit"
```

Ansible : Injection de données

- Utilisation de variables dans les arguments des modules
- Dans l'inventaire les machines sont classées par groupe
- Une variable peut recevoir sa valeur
  - depuis la ligne de commande
  - en fonction de son appartenance à un groupe
  - spécifquement pour une machine
  - depuis le fichier *defaults/main.yml* du role (valeur par défaut)

Puppet : Les objets



Puppet : Articulation des objets I

- Le **node** est le point d'entrée de puppet. C'est à partir de là que le *puppetmaster* va compiler le catalogue destiné à l'ordinateur administré.
- Un **role** et un seul est affecté à un node. Il définit un ensemble de composants à installer sur un ordinateur.
- Le role aggrège plusieurs **profiles**. Un profile correspond à la configuration d'un ensemble de composants de l'ordinateur.
- Une ou plusieurs **classes** sont déclarées dans un profile. C'est à ce moment qu'elles reçoivent les valeurs des paramètres de classes.
- Un **module** définit une ou plusieurs classes qui seront utilisées par les profiles via une déclaration.

mathrice

Puppet : Articulation des objets II

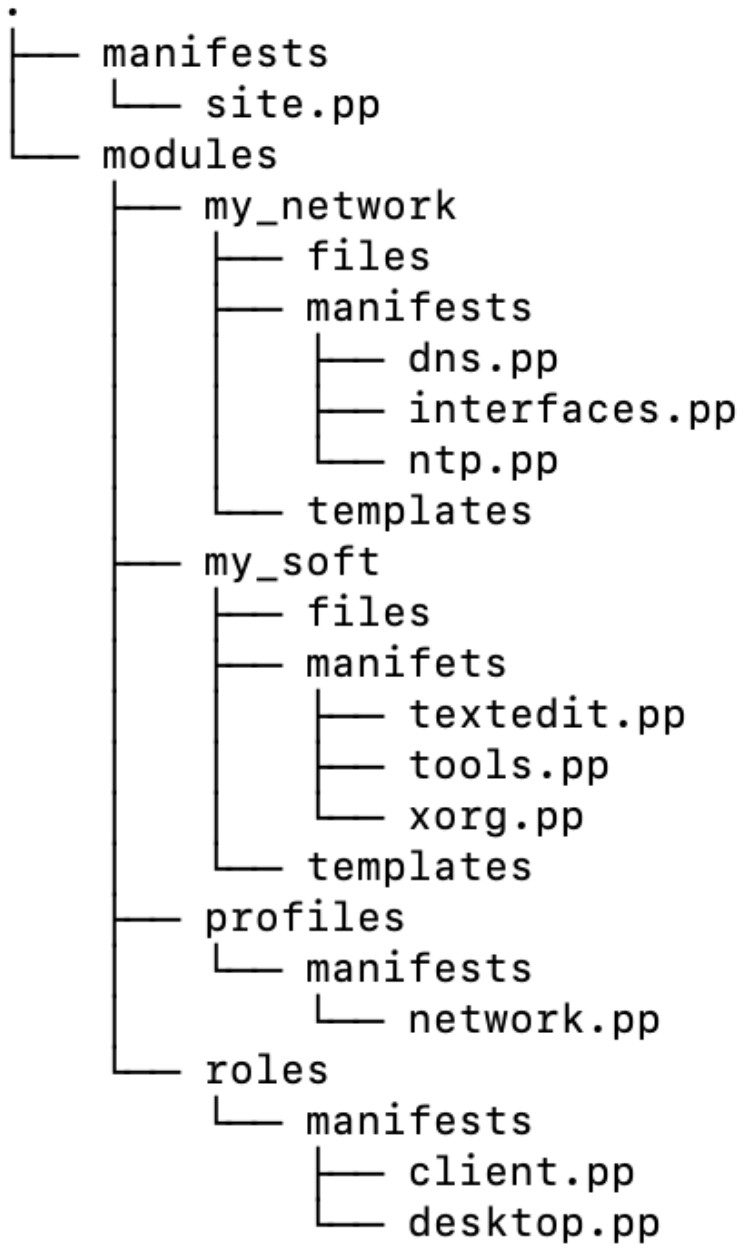
- Une classe se compse de **ressources** qui agissent sur le comportement de la machine
  - ressource *package* $\rightarrow$ installation d'un paquet (Debian / Ubuntu)
  - ressource *file* $\rightarrow$ création d'un répertoire, d'un lien, postionnement de droits
  - ressource *copy* $\rightarrow$ copie d'un fichier
  - ...

Puppet : Le formalisme

- Ensemble de fichiers manifests (.pp) écrit dans un pseudo-langage
- Langage purement déclartif (impossible de réaffecter des variables)
- Existence de boucles et de structures de controles pour conditionner l'application de ressources
- Collecte d'informations depuis la machine cible $\rightarrow$ variables
- Applications des ressources dans un ordre arbitraire $\rightarrow$ mécanismes de dépendances

mathrice

Puppet : Organisation des fichiers



Puppet : Exemple - manifests/site.pp

```
node client1 {
  include role::client
}

node client2 {
  include role::client
}

node client3 {
  include role::client
}
```

Puppet : Exemple - modules/roles/manifests/client.pp

```
class role::client {
  include profile::network
  include profile::applications
}
```

Puppet : Exemple - modules/profiles/manifests/network.pp

```
class profile::network {

  class { 'my_network::interfaces':
  }
  class { 'my_network::dns':
  }
  class { 'my_network::ntp':
  }

}
```

Puppet : Injection de données

- Variables
  - définie localement à une classe
  - recherche hierarchique dans une source de données de type clé/valeur $\rightarrow$ *hieradata*
- Paramètres de classes
  - Spécifié dans la déclaration de classe (au niveau du profile)
  - Recherche automatique dans une source de données de type clé/valeur $\rightarrow$ *hieradata*
  - Valeur par défaut dans la définition de classe

mathrice

Synthèse

- Déroulé linéaire des taches à effectuer sur les modules
- Réfléchir en termes d'opérations à effectuer sur la machine
- Formalisation des opérations en YAML
- Constitution d'un inventaire des ordinateurs administrés

- Bon modèle d'abstraction à 3 niveaux (*role*,*profile*,*class*)
- Concevoir en fonction de l'état cible dans lequel doit être la machine
- Nécessité de définir des relations de dépendance pour le bon déroulé des opérations

Progression

Bilan

- Le retour sur investissement n'est pas immédiat (délai d'apprentissage)
- Deux outils avec deux philosophies différentes
  - Ansible $\rightarrow$ configuration unique
  - Puppet $\rightarrow$ maintient permanent d'une configuration
- Ansible semble plus facile d'accès que Puppet
- Attention à l'idempotence (modification la première fois, aucun changement les fois suivantes)
- Le couplage avec un outil de *versioning* est indispensable

Perspectives

- Se pencher sur les possiblités offertes par les outils pour les systèmes Windows
- Coupler l'outil de gestion de configuration avec un outil de reporting
- Articulation ds ces outils avec les conteneurs (Kubernetes) ?