

Ansible

Executer des tâches sur des machines hôtes.

Yvon Vieville

2023-05-24

Volubis

Ansible

POEI Exploitation organisé par Armonie.

Yvon Vieville

2023-05-24

Volubis

Imaginez



Sur n Machines



en automatisant

Reduction Erreur

Tâches répétitives

Gestion de version

Réutilisation

Amélioration

Fiabilité

moins chronophage

Objectifs

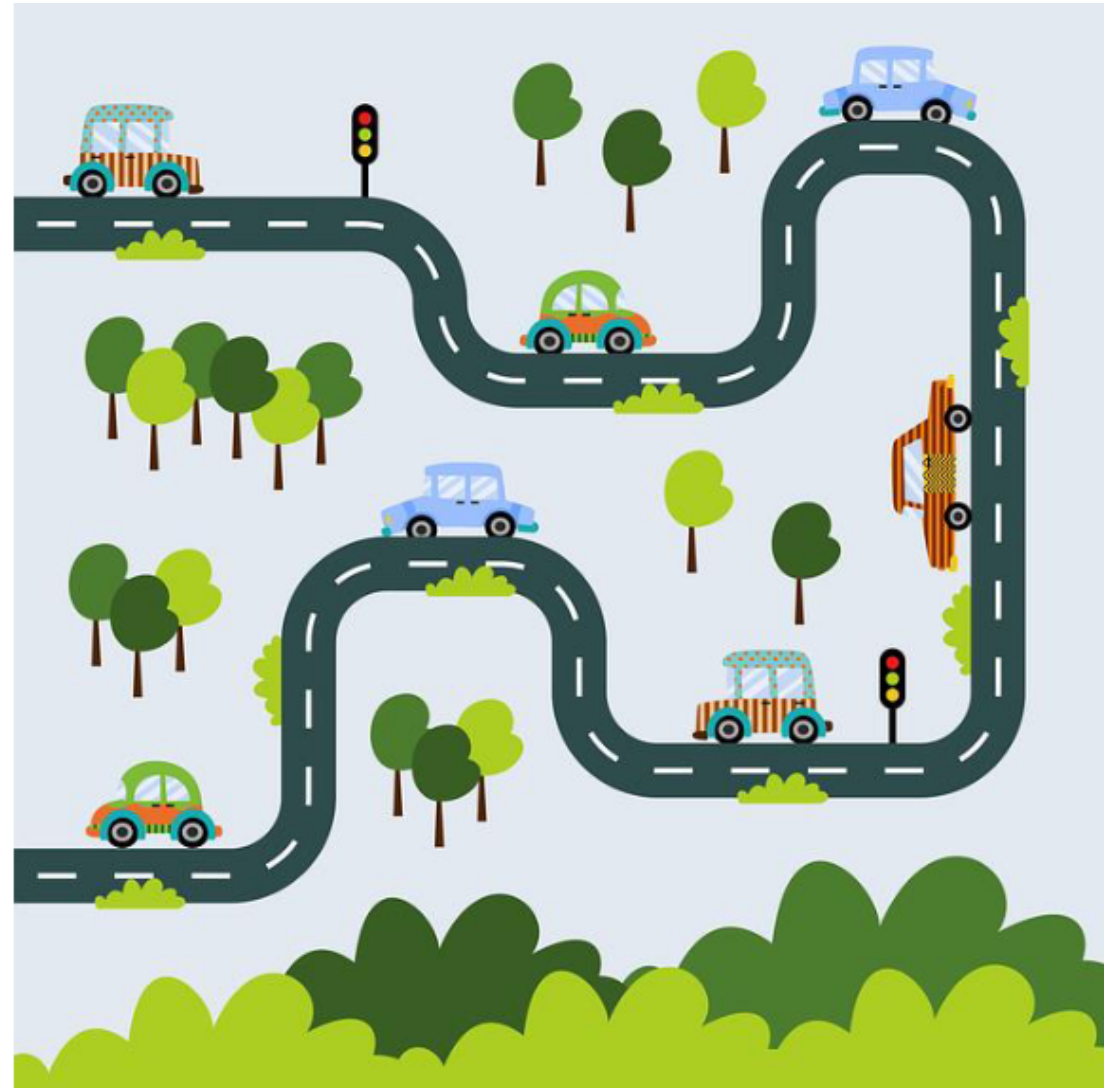
2 jours pour :

- 1 Découvrir Ansible.**
- 2 Découvrir quelques cas d'utilisations sur un/des IBMi(s).**
- 3 ...**

Agenda

**Concepts généraux d'Ansible
(infrastructure Linux)**

Ansible appliqué à IBM i



Organisation

- Les outils
 - diveInto Playground, l'IBMI en local
- Les supports... ressources
- Les horaires
 - 🕒 9H à 12H30 puis de 13h30 à 17H
- Pauses
 - ¼ h par demi journée, 😊 N'hésitez pas à m'y faire penser.!...
- Les activités
 - Cours théoriques, Travaux pratiques ➔ moi ,moi/vous ,vous....

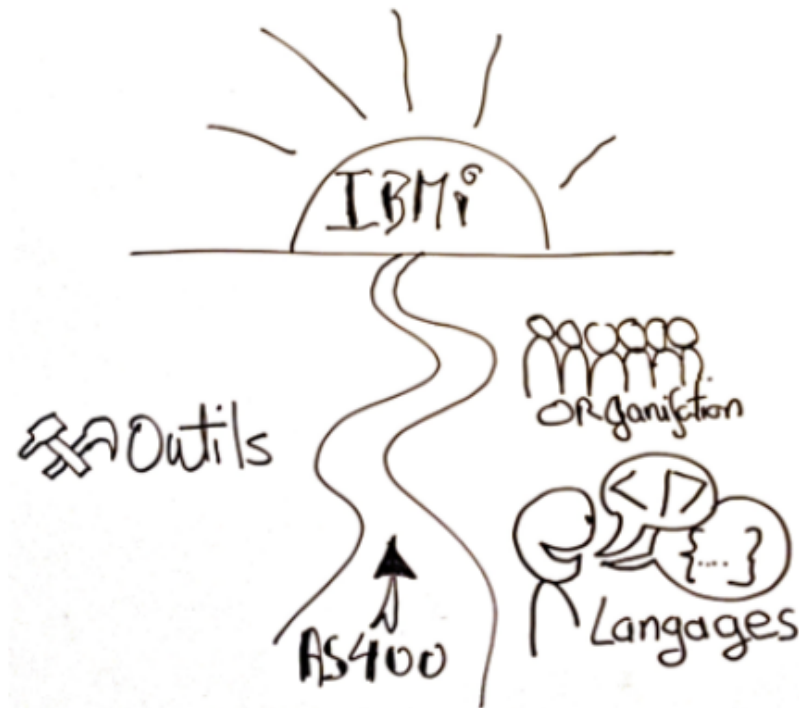
Faisons connaissances....

Moi ?

les routes de l'IBMi

 yvon vieville

 06 80 14 45 17



Faisons connaissances....

Vous

- ▶ Je suis...
- ▶ Je travaille...Projets, Postes...
- ▶ J'ai déjà utilisé Ansible (?) pour...
- ▶ Mes attentes sont...
- ▶ Mes questions...Moi



Environnement

DiveInto



Concepts généraux d'Ansible On linux !

Introduction à Ansible et à sa mise en place

J1 Concepts généraux d'Ansible (infrastructure Linux)

Introduction à Ansible

Configuration et inventaire

Les bases des Playbooks

Les modules Ansible de base

Les variables et les faits

Gestion des erreurs et débogage



Module 1 - Concepts généraux d'Ansible (infrastructure Linux)

Introduction à Ansible et à sa mise en place

Qu'est-ce qu'Ansible ?



- 1 Outil Open source
- 2 Automatisation tâches IT
- 3 Infrastructure as a code
- 4 Python
- 5 Agentless
- 6 SSH

Ses avantages



1

Facilité d'utilisation

2

Idempotence

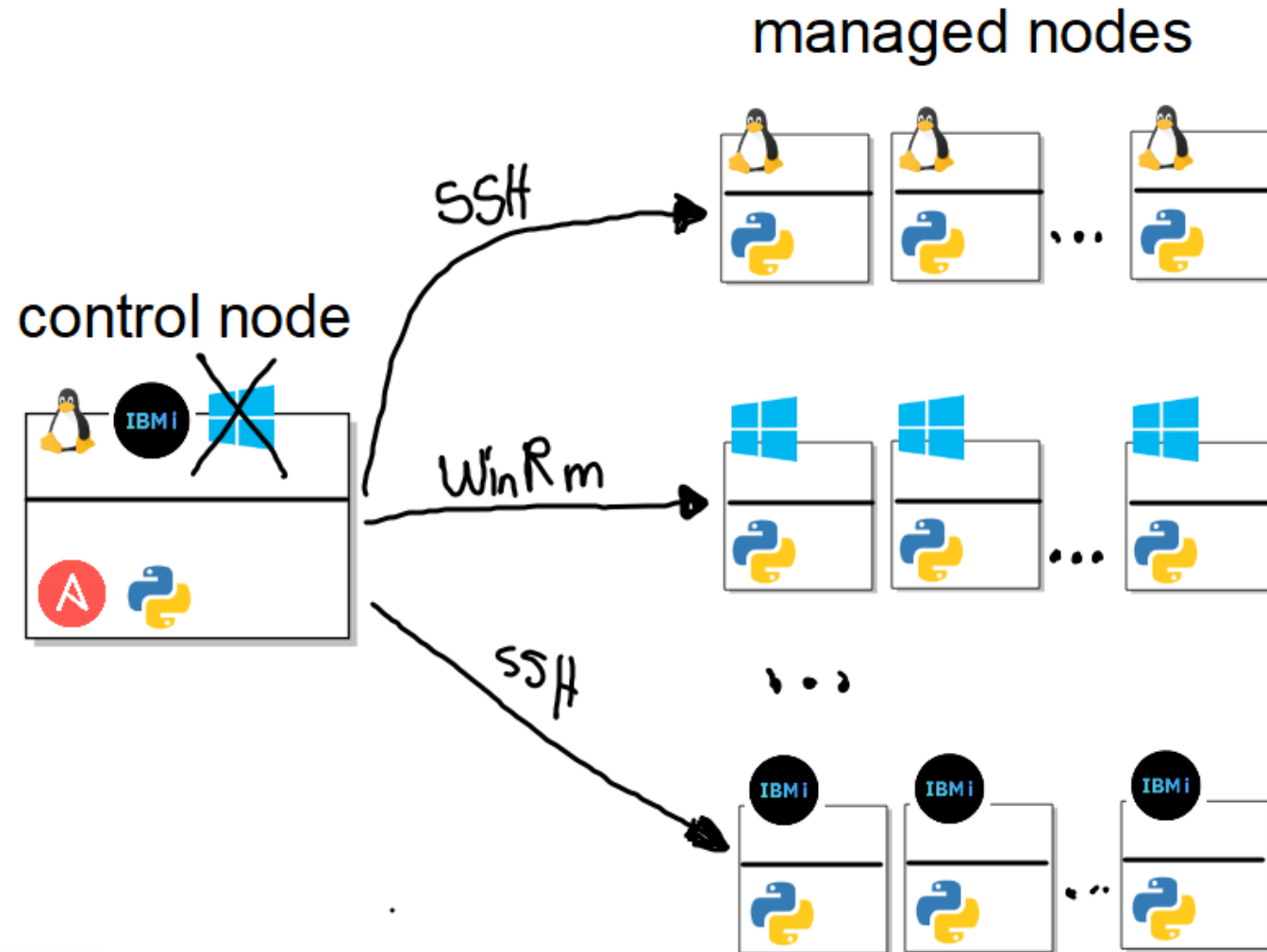
3

Agentless

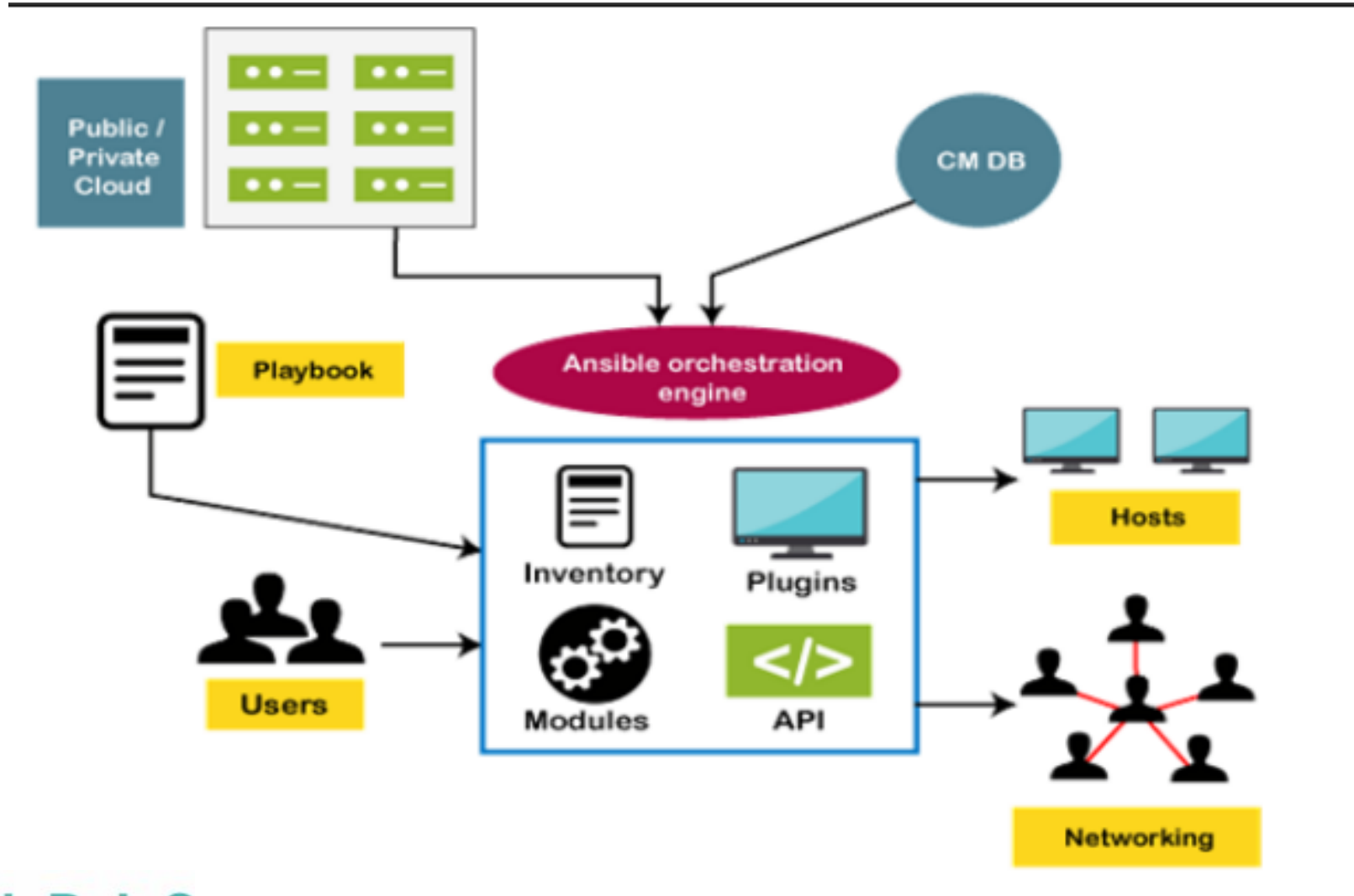
4

et IBMi 😊 !

Architecture



Comment ça marche !



Un projet Type !

```
novy@DESKTOP-HS6CE0A:~/ansibleMelodie/templates$ tree
.
├── ansible_simple
│   ├── 00_inventory.yml
│   ├── 10_playbook.yml
│   ├── README.md
│   ├── ansible.cfg
│   ├── group_vars
│   └── host_vars
└── 3 directories, 4 files
novy@DESKTOP-HS6CE0A:~/ansibleMelodie/templates$
```

Cas d'utilisation



DevOps

1

Déploiement d'applications

2

Gestion de la configuration

3

Orchestration

4

et IBMi 😊 !

5

....

Découverte du Lab DIVEINTO

TP01 Mise en place du lab

Objectifs :

- Découvrir Google Shell
- Installer l'environnement.
- Découvrir l'environnement.
- Utiliser l'environnement.
- Savoir éditer un source avec google editor
- Vérifier les connexions ssh



Questions ?

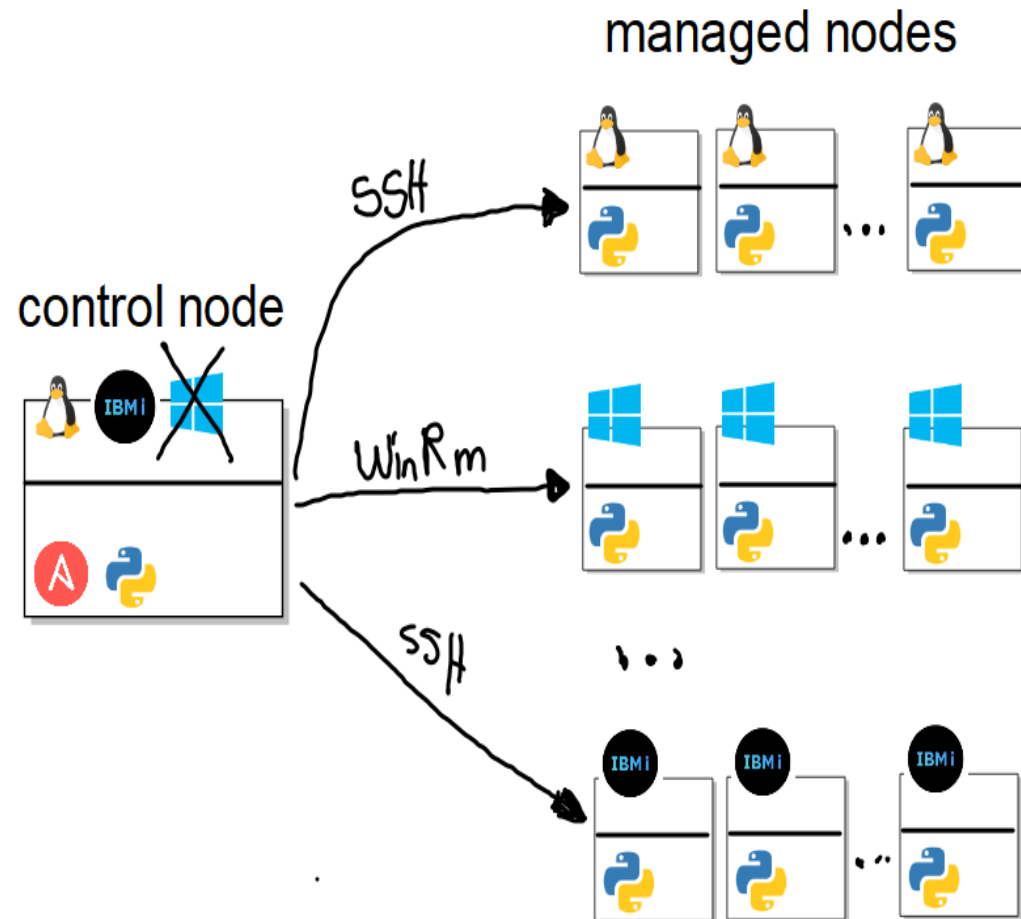
Quizz

intro prochain chapitre

Module 2 - Installation et configuration d'Ansible

De l'installation à la gestion des hôtes

Installation



documentation Ansible

pip (python)

```
sudo apt install python3-pip  
pip3 install ansible
```

avec les paquets des distributions (debian)

```
sudo apt install ansible
```


Configuration

```
inventory      = /etc/ansible/hosts
forks          = 5
sudo_user      = root
ask_sudo_pass  = True
ask_pass       = True
gathering      = implicit
gather_subset  = all
roles_path     = /etc/ansible/roles
log_path       = /var/log/ansible.log
vault_password_file = /path/to/vault_password_file
fact_caching_connection = /tmp
pipelining     = False
```

ansible.cfg

- ANSIBLE_CONFIG
- dossier du playbook
- ~/.ansible/ansible.cfg
- /etc/ansible/ansible.cfg

configuration

Commandes


documentation Ansible

```
ansible-config
ansible-config --version # afficher le chemin de la configuration
ansible-config view  # voir le ansible.cfg pris en compte
ansible-config dump  # liste toutes les variables ansible
ansible-config dump --only-changed #valeurs par défaut modifiée
```

Configuration

```
[defaults]
inventory      = hosts
host_key_checking = False
forks          = 5

[ssh_connection]
pipelining = True
ssh_args = -o PreferredAuthentications=publickey
```

- host key checking  fingerprint
- pipelining
- fork = parallélisation
- ...

L'inventaire

```
templates > ansible_simple > 00_inventory.yml > ...
Yvon Vieville, last week | 2 authors (Yvon Vieville and othe
1  ---
2  control:
3    · hosts:
4      · ubuntu-c:
5        · ansible_connection: local
6  centos:
7    · hosts:
8      · centos1:
9  ubuntu:
10   · hosts:
11     · ubuntu1:
12 managed:
13   · children:
14     · centos:
15     · ubuntu:
16   ...
17 |
```

- Le catalogue de votre infrastructure.
- Serveurs ➡ les "hosts"
- Type de serveurs ➡ les groupes
- ini, ✓ yaml et json
- Statique ou Dynamique
- Arbre **all** ➡ la racine
- inventory c'est
 - fichier d'inventaire
 - répertoire group_vars
 - répertoire host_vars

Les variables

```
---
control:
  hosts:
    ubuntu-c:
      ansible_connection: local
centos:
  hosts:
    centos1:
      ansible_port: 2222
    centos2:
    centos3:
  vars:
    ansible_user: root
ubuntu:
  hosts:
    ubuntu1:
    ubuntu2:
    ubuntu3:
  vars:
    ansible_become: true
    ansible_become_pass: password
linux:
  children:
    centos:
    ubuntu:
  ...
```

```
---
control:
  hosts:
    ubuntu-c:
centos:
  hosts:
    centos1:
    centos2:
    centos3:
ubuntu:
  hosts:
    ubuntu1:
    ubuntu2:
    ubuntu3:
linux:
  children:
    centos:
    ubuntu:
  ...
```

novy@DESKTOP-HS6CE0A:~/ansible/

- 00_inventory.yml
- 01_inventory.yml
- 10_playbook.yml
- README.md
- ansible.cfg
- group_vars
 - centos
 - variables.yml
 - ubuntu
 - variables.yml
- host_vars
 - ubuntu-c
 - variables.yml

5 directories, 8 files

variables.yml U X

exemple > inventaire > ansible_simple > group_vars

```
1  ansible_become: true
2  ansible_become_pass: password
3
4
```

Les variables

Commandes

documentation Ansible

```
ansible-inventory -i <inventory_file> --list  
ansible-inventory -i <inventory_file> --list --yaml  
ansible-inventory -i 01_inventory.yml --list --export  
ansible-inventory -i 00_inventory.yml --graph --vars
```

CLI

- (Command Line Interface) est l'interface en ligne de commande
- [documentation Ansible](#)
- pratique pour tester (ping, debug)
- exécuter des tâches ad-hoc
- gérer les fichiers de configuration
-

Tester une connexion

```
ansible -i "centos1," all -u ansible -m ping
ansible -i 01_inventory.yml centos1 -u ansible -m ping
```

- u : user distant utilisé
- b : passer les commandes en élévation de privilèges (sudo)
- k ou --ask-pass > password SSH
- K ou --ask-become-pass > password pour élévation privilèges
- C ou --check : faire un dry run
- D ou --diff : avoir un output de la diff

Tester une connexion

- key-file : lien direct vers la clef privée
- e ou --extra-vars : définir des variables
- ask-vault-pass : déchiffrer un secret vault
- vault-password-file : fichier pour déchiffrer
- f x ou --forks : paralléliser
- vvv : verbose

Debougner

```
ansible centos1 -m debug -a "msg='Bonjour!'"  
ansible -i "centos2," all -b -e "var1=Bonjour !" -m debug -a 'msg={{ var1 }}'
```

Cataloguons notre infrastructure !

TP02 Configurer Ansible

Objectifs :

- Éditer un fichier de configuration.
- Tester notre configuration avec la commande `ansible-config`.
- Écrire un fichier d'inventaire pour cataloguer notre infrastructure.
- Tester notre inventaire avec la commande `ansible-inventory`.
- Utiliser le module Ping en mode commande pour tester notre inventaire.
- Utiliser le module Debug en mode commande pour tester nos variables d'inventaire.



Questions ?

Quizz

intro prochain chapitre

Module 3 - Les bases des Playbooks

Comprendre et créer votre premier Playbook Ansible

Le playbook

```
#ssh-addkey.yml
---
- hosts: all
  gather_facts: false
  collections:
    - ibm.power_ibmi
  vars:
    target_user: '{{ ansible_ssh_user }}'

  tasks:
    - name: install ssh key
      authorized_key: user={{ target_user }}
      key="{{ lookup('file', '~/.ssh/id_rsa.pub') }}"
      state=present

    - name: find the home directory
      ibmi_user_and_group:
        operation: 'display'
        user: '{{ target_user }}'
      register: user_result

    - name: set home_dir
      set_fact:
        home_dir: "{{ user_result['result_set'][0]['HOME_DIRECTORY'] }}"

    - name: the authority of home directory has to be 0700
      file:
        path: '{{ home_dir }}'
        state: directory
        mode: '0700'
```

- Un fichier structuré au format YAML.
- Déclenche les tâches à réaliser.
- Spécifier quel user et comment (become) ?
- peut inclure des variables (😞❌)
- peut inclure des tasks (actions) (😞❌)
- sert à articuler l'inventary avec les rôles

Structure d'un Playbook

```
---
# Les documents YAML commencent par le séparateur de document ---

# Le - dans YAML indique un élément de liste. Le playbook contient une liste de jeu, chaque jeu étant un dictionnaire
-

  # Hosts : où notre pièce s'exécutera et les options avec lesquelles elle s'exécutera
  hosts: centos
  user: root
  # Vars : variables qui s'appliqueront à la pièce, sur tous les systèmes cibles

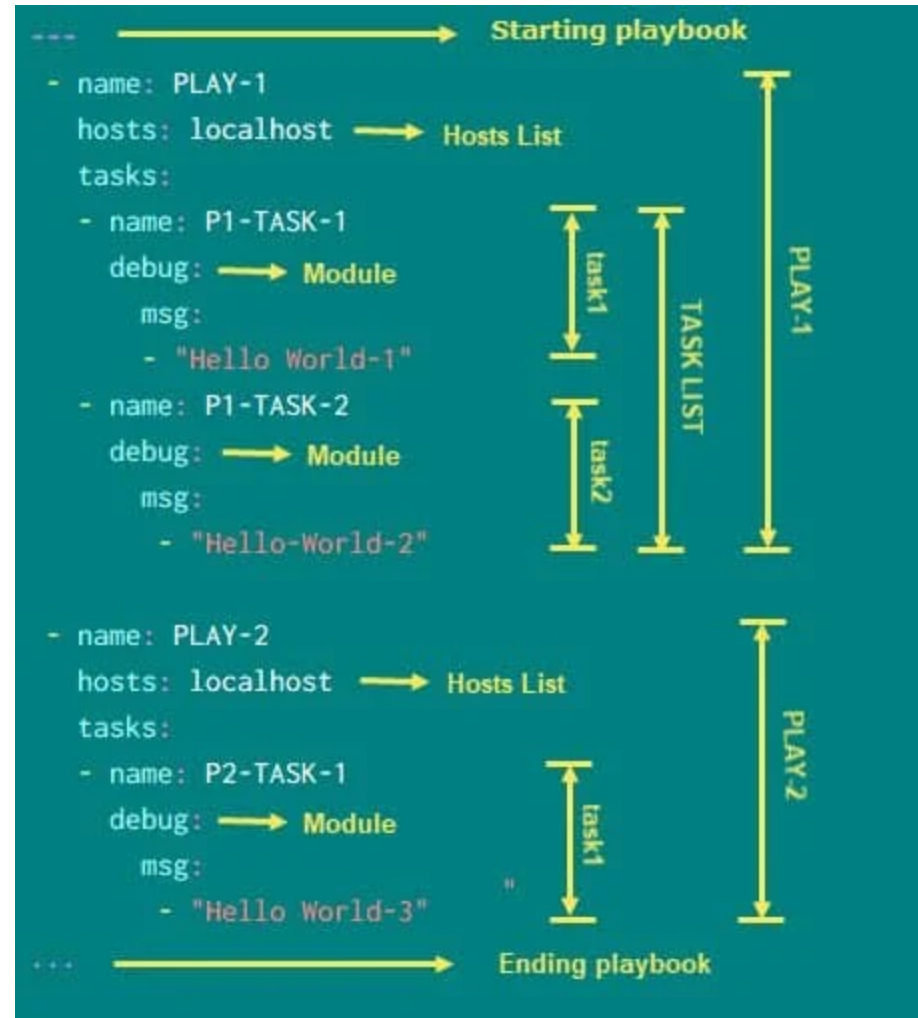
  # Tasks : la liste des tâches qui seront exécutées dans la pièce.
  # Cette section peut également être utilisée pour les tâches préalables et postérieures.
  tasks:
    - name: Copie le fichier Toto
      copy:
        src: toto.txt
        dest: /tmp
      notify: Nouveau Toto

  # Handlers : la liste des handlers qui sont exécutés en tant que clé de notification d'une tâche.
  handlers:
    - name: Nouveau Toto
      debug:
        msg: toto a changé

# Trois points indiquent la fin d'un document YAML
```



Structure d'un Playbook




```
#ssh-addkey.yml
---
- hosts: all
  gather_facts: false
  collections:
    - ibm.power_ibmi
  vars:
    target_user: '{{ ansible_ssh_user }}'

  tasks:
    - name: install ssh key
      authorized_key: user={{ target_user }}
      key="{{ lookup('file', '~/.ssh/id_rsa.pub') }}"
      state=present

    - name: find the home directory
      ibmi_user_and_group:
        operation: 'display'
        user: '{{ target_user }}'
      register: user_result

    - name: set home_dir
      set_fact:
        home_dir: "{{ user_result['result_set'][0]['HOME_DIRECTORY'] }}"

    - name: the authority of home directory has to be 0700
      file:
        path: '{{ home_dir }}'
        state: directory
        mode: '0700'
```

- Un fichier structuré au format YAML.
- Déclenche les tâches à réaliser.
- Spécifier quel user et comment (become) ?
- peut inclure des variables (😞❌)
- peut inclure des tasks (actions) (😞❌)
- sert à articuler l'inventary avec les rôles

YAML !

- Un langage de sérialisation de données structurées.
- Clé - Valeur : `name: install ssh key`
- Liste d'éléments.

```
- element 1  
- element 2  
- element 3
```

- dictionnaire.

```
ibmi_user_and_group:  
  operation: 'display'  
  user: '{{ target_user }}'
```

YAML !

- Commentaires : `# Ceci est un commentaire`
- L'indentation est cruciale en YAML pour définir les niveaux de structure. Deux espaces sont généralement utilisés pour l'indentation.
- Type de données
 - Chaines de caractères : `'Bonjour'` , `"Bonjour"`
 - Nombres : `3` , `3.14`
 - Booléens : `true` , `false`
 - Null : `null`

une tâche

- Une tâche est une opération à effectuer.
- La plus petite unité d'action automatisable.
- Idempotente
- Statut d'exécution
 - changed,succeeded,failed,skipped
- Comprend :
 - un nom
 - une action (module)
 - des paramètres
 - ...

Exécution de Playbooks

Commandes

[documentation Ansible](#)

```
ansible-playbook recupFactsIBMi.yml -i ./00_inventory.yml --vault-id password.txt --limit itest9  
ansible-playbook recupFactsIBMi.yml -i ./00_inventory.yml --vault-id password.txt --check
```

nombreuses options !

- l : limit > spécifier un/des groupes ou serveurs ou patterns
- u : user
- b : become > sudo
- k : password de ssh (à éviter)
- K : password du sudo
- C : check > dry run
- D : diff > afficher les différences avant/après les ks (actions)

nombreuses options !

- ask-vault : prompt pour le password vault
- syntax-check : vérifier la syntax
- vault-password-file : passer le vault password par un fichier
- e : surcharger n'importe quelle variable
- f : nombre de parallélisation
- t : filtrer sur les tags (--skip-tags)
- flush-cache : éviter l'utilisation du cache
- step : une tâche à la fois (confirmation via prompt)
- start-at-task : commencer à une tâche spécifiquement
- list-tags : lister tous les tags rencontrés
- list-tasks : liste les tâches qui vont être exécutées

nombreuses options !

```
ansible-playbook recupFactsIBmi.yml -i ./00_inventory.yml --vault-id password.txt --list-tasks
```

```
novy@DESKTOP-HS6CE0A:~/testsAnsible$ ansible-playbook recupFactsIBmi.yml -i ./00_inventory.yml --vault-id password.txt --list-tasks

playbook: recupFactsIBmi.yml

  play #1 (all): all    TAGS: []
    tasks:
      return facts IBMi TAGS: []
      affichage des facts IBMi TAGS: []
novy@DESKTOP-HS6CE0A:~/testsAnsible$
```


Premier playbook !

TP03 Configurer Ansible

Objectifs :

- Créer un playbook basique en utilisant le module ping et debug utilisé dans le TP02.
- Le lancer.
- Analyser les résultats.



Questions ?

Quizz

intro prochain chapitre

Module 4 - Les modules Ansible de base

Comprendre et utiliser les modules Ansible

Qu'est-ce qu'un module Ansible ?



- Unité de code réutilisable ▶ Tâches spécifiques
- Ensemble d'actions avec un retour (id,résultat,...)
- Automatiser des tâches.
 - copier un fichier
 - installer un package
 - créer un profil utilisateur
 - démarrer un service
- Combinés dans un Playbook.
- Fournis par ansible pour l'essentiel (builtin)
- on peut développer son module !

Pourquoi utilise-t-on des modules dans Ansible ?



- 1 Efficacité
- 2 Réutilisabilité
- 3 Lisibilité
- 4 Extensibilité

Types de modules disponibles

documentation Ansible

nom	description
ping	Teste la connexion à un hôte
debug	Affiche des contenus de variables ou messages
setup	Collecte infos de configurations des hôtes
command	exécute une commande sur un hôte cible
copy	copie un fichier de l'hôte local ou distant à un emplacement sur l'hôte distant
file	gère les fichiers et les répertoires sur l'hôte cible

Exemples d'utilisation de modules dans un Playbook

```
---  
- name: Playbook d'exemple  
  hosts: servers  
  
  tasks:  
    - name: Assurer l'existence du répertoire /tmp/directory  
      file:  
        path: /tmp/directory  
        state: directory  
  
    - name: Copier le fichier /tmp/file1 dans le répertoire /tmp/directory  
      copy:  
        src: /tmp/file1  
        dest: /tmp/directory/file1  
  
    - name: Exécuter la commande 'ls' pour lister le contenu du répertoire /tmp/directory  
      command:  
        cmd: ls /tmp/directory  
  
...
```



Questions ?

Quizz

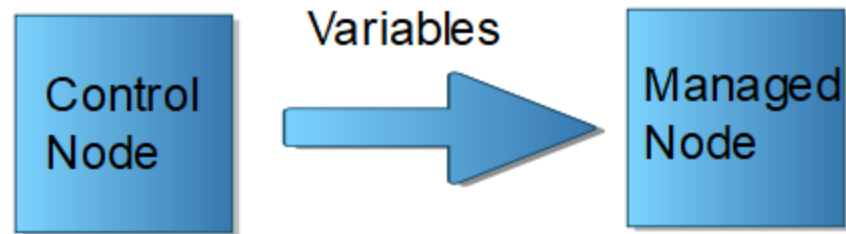
intro prochain chapitre

Module 5 - Les variables et les faits

Comprendre et utiliser les variables et les faits dans Ansible

Qu'est-ce qu'une variable dans Ansible ?

documentation Ansible



Définition d'une variable ?

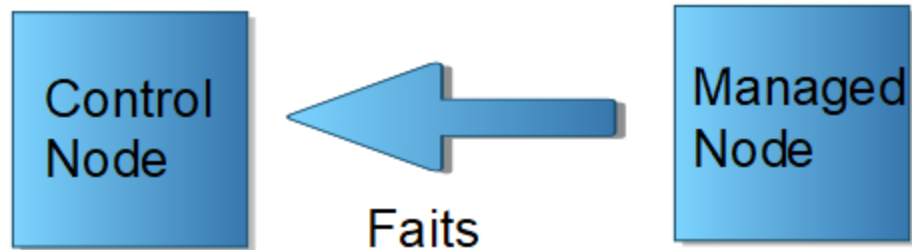
Utilisation ?

Déclaration d'une variable ?

- flexibles
- réutilisables
- abstraire les détails

Qu'est-ce qu'un fait dans Ansible ?

documentation Ansible



Définition d'un fait ?

Utilisation ?

recueil des faits ?

- flexibles
- réutilisables
- abstraire les détails

Déclaration des variables

fichiers d'inventaires

```
|— 00_inventory.yml  
|— group_vars  
    |— all.yml
```

dans group_vars/all.yml

```
src_path: /tmp/src  
dest_path: /tmp/dest
```

playbook

```
vars:  
  src_path: /tmp/src  
  dest_path: /tmp/dest
```

Utilisation des variables

```
tasks:  
  - name: Copier un fichier  
    copy:  
      src: "{{ src_path }}"  
      dest: "{{ dest_path }}"
```

Comment Récolter les faits des hôtes ?

```
bash-4.2$ ansible -m setup localhost

[WARNING]: No inventory was parsed, only implicit localhost is available
localhost | SUCCESS => {
  "ansible_facts": {
    "ansible_apparmor": {
      "status": "disabled"
    },
    "ansible_architecture": "x86_64",
    "ansible_date_time": {
      "date": "2023-03-02",
      "day": "02",
      "epoch": "1677755762",
      "hour": "12",
      "iso8601": "2023-03-02T11:16:02Z",
      "iso8601_basic": "20230302T121602037290",
      "iso8601_basic_short": "20230302T121602",
      "iso8601_micro": "2023-03-02T11:16:02.937290Z",
      "minute": "16",
      "month": "03",
      "second": "02",
      "time": "12:16:02",
      "tz": "CET",
      "tz_dst": "CEST",
      "tz_offset": "+01:00",
      "weekday": "Thursday",
      "weekday_number": "4",
      "weeknumber": "09",
      "year": "2023"
    }
  }
}
```

`ansible -m setup localhost`
ou `gather facts`

```
#ssh-addkey.yml
---
- hosts: all
  gather_facts: false
  collections:
    - ibm.power_ibmi
  vars:
    target_user: '{{ ansible_ssh_user }}'
```

par défaut c'est 'true'

Comment utiliser les faits dans un playbook ?

```
tasks:  
  - name: installer htop sur Debian  
    apt:  
      name: htop  
      when: ansible_os_family == "Debian"
```

Forte précedence des variables (ordre hiérarchique)

```
command line values (eg "-u user")
role defaults [1]
inventory file or script group vars [2]
inventory group_vars/all [3]
playbook group_vars/all [3]
inventory group_vars/* [3]
playbook group_vars/* [3]
inventory file or script host vars [2]
inventory host_vars/* [3]
playbook host_vars/* [3]
host facts / cached set_facts [4]
play vars
play vars_prompt
play vars_files
role vars (defined in role/vars/main.yml)
block vars (only for tasks in block)
task vars (only for the task)
include_vars
set_facts / registered vars
role (and include_role) params
include params
extra vars (always win precedence)
```


Une mini CMDB

TP04 MiniCMDB

Objectifs :

- Créer un Playbook qui utilise des variables et les gather facts.
- via une template et les modules fetch et copy on crée des fichiers html avec des valeurs systèmes.
- Exécuter le Playbook et analyser les résultats
- (module 6) Provoquer une erreur et la déboguer....mode verbose



Questions ?

Quizz

intro prochain chapitre

Module 6 - Gestion des erreurs et débogage

Apprendre à traiter et à prévenir les erreurs
avec Ansible

Pourquoi la gestion des erreurs est-elle importante ?

Infrastructure as a code !

- pannes et des perturbations
- problèmes plus profonds dans le code ou la configuration

Ansible

- mécanismes intégrés pour traiter les erreurs
- Utiliser cette gestion des erreurs peut aider à maintenir la stabilité du système

Comment Ansible gère les erreurs dans les Playbooks

TODO: image d'un playbook ko syntaxe

TODO: image du résultat au lancement

Les Handlers

```
---
- hosts: webservers
  tasks:
    - name: Installer le serveur web
      apt:
        name: apache2
        state: present
      notify:
        - Redémarrer le serveur web
    - name: Déployer le fichier de configuration du serveur web
      copy:
        src: /files/httpd.conf
        dest: /etc/httpd/conf/httpd.conf
      notify:
        - Redémarrer le serveur web
  handlers:
    - name: Redémarrer le serveur web
      service:
        name: apache2
        state: restarted
```

Le mode verbeux

TODO: Mettre un bon exemple quand cela marchera....avec diveinto

```
ansible-playbook -i hosts playbook.yml -vvv
```

```
PLAYBOOK: playbook.yml *****
1 plays in playbook.yml
```

```
PLAY [all] *****
```

```
TASK [Gathering Facts] *****
ok: [host1]
ok: [host2]
```

```
TASK [Ping hosts] *****
ok: [host1] => {
  "changed": false,
  "ping": "pong"
}
ok: [host2] => {
  "changed": false,
  "ping": "pong"
}
```

Le mode debug

```
---
- hosts: all
  tasks:
    - name: Print system information
      debug:
        msg: "The system {{ ansible_hostname }} is running on {{ ansible_distribution }}"

    - name: Print value of custom variable
      debug:
        var: my_custom_variable
```

```
TASK [Print system information] *****
ok: [localhost] => {
  "msg": "The system localhost is running on Ubuntu"
}
```

```
TASK [Print value of custom variable] *****
ok: [localhost] => {
  "my custom variable": "Value of my custom variable"
```



Bonnes pratiques pour la gestion des erreurs avec Ansible

- [documentation Ansible](#)

Utilisation de 'check mode' et 'dry run'.

- Dry run avec --check

```
ansible-playbook recupFacts.yml -i ./00_inventory.yml --vault-id password.txt --limit itest9 --check
```

- Afficher les différences --diff

```
ansible-playbook recupFacts.yml -i ./00_inventory.yml --vault-id password.txt --limit itest9 --check --diff
```

Bonnes pratiques pour la gestion des erreurs avec Ansible

Traitement d'erreur adéquat.

- `ignore_errors` pour continuer l'exécution du playbook malgré des erreurs.
- `failed_when` pour définir les conditions d'échec

Bonnes pratiques pour la gestion des erreurs avec Ansible

Tester avec différents niveaux de verbosité

- `-v` pour un niveau de verbosité minimal.
- de `-vv` à `-vvv` pour plus de détails.
- `-vvvv` pour le débogage de connexion.

Une mini CMDB

TP04 MiniCMDB

Objectifs :

- Exécuter le Playbook et analyser les résultats
- Provoquer une erreur et la déboguer....mode verbose



Questions ?

Quizz

intro prochain chapitre

Concepts avancés

Introduction

Les rôles

```
├── 00_inventory.yml
├── group_vars
├── host_vars
├── playbook.yml
├── roles
│   └── users
│       ├── defaults
│       │   └── main.yml
│       ├── files
│       ├── handlers
│       │   └── main.yml
│       ├── meta
│       │   └── main.yml
│       ├── README.md
│       ├── tasks
│       │   └── main.yml
│       ├── templates
│       ├── tests
│       │   ├── inventory
│       │   └── test.yml
│       └── vars
│           └── main.yml
```

- name: installation local de la clef ssh
connection: local
hosts: localhost
roles:
 - ssh_keygen
- name: installation des serveurs (users, nginx)
hosts: all
become: yes
roles:
 - users
 - nginx

Les collections

- power ibmi

```
tree /home/novy/.ansible/collections/ansible_collections/ibm
/home/novy/.ansible/collections/ansible_collections/ibm
├── power_ibmi
│   ├── FILES.json
│   ├── MANIFEST.json
│   ├── README.md
│   ├── bindep.txt
│   ├── changelogs
│   │   ├── changelog.yaml
│   │   ├── config.yaml
│   │   └── fragments
│   └── docs
│       ├── Makefile
│       └── source
│           ├── conf.py
│           ├── getting_started.rst
│           ├── index.rst
│           ├── installation.rst
│           └── modules
│               ├── ibmi_at.rst
│               ├── ibmi_cl_command.rst
│               ├── ibmi_copy.rst
│               ├── ibmi_device_vary.rst
│               ├── ibmi_display_fix.rst
│               ├── ibmi_display_subsystem.rst
│               ├── ibmi_download_fix.rst
│               ├── ibmi_download_fix_status.rst
│               ├── ibmi_end_subsystem.rst
│               └── ibmi_ethernet_port.rst
└── st
    └── st
```

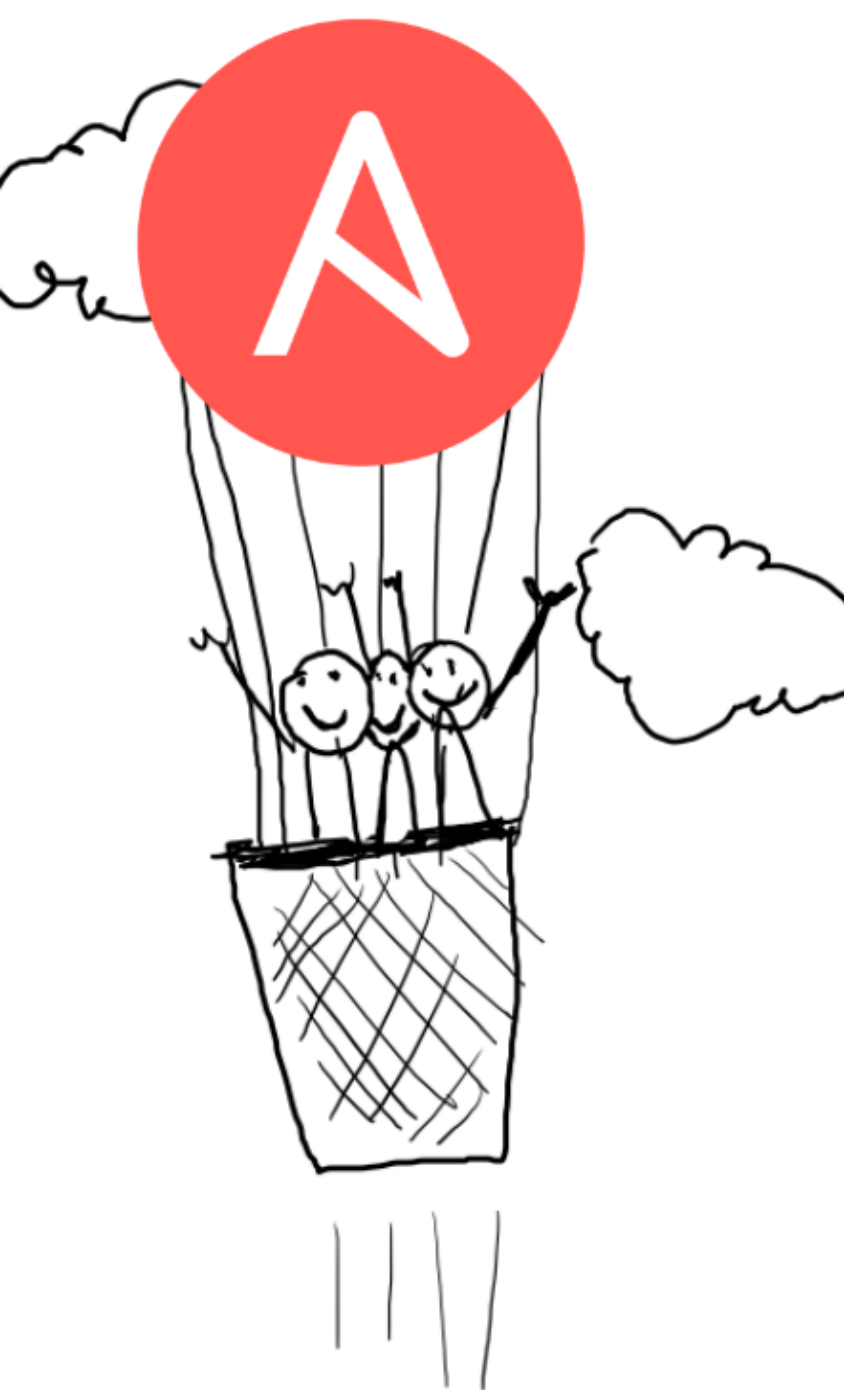
```
---
- hosts: all
  gather_facts: no
  collections:
    - ibm.power_ibmi

  tasks:
    - block:
      - name: Display a system value
        ibmi_sysval:
          sysvalue:
            - {'name': 'qccsid'}
        register: dspsysval_ccsid_result
        tags: display, sysval

      - name: Display the returned parameters
        debug:
          msg: "{{ dspsysval_ccsid_result }}"
```



VOLUMIS



Pour aller plus haut..

TODO: indiquer ressources diveninto,ansible,...xvaki,...

1. un grand merci à James SPURIN de **DIVEINTO**.
2. Xavier de **XAVKI**
chaine Ansible
3. **Volubis**
chaine youtube
les routes de l'IBMi
4. **doc Ansible**

On IBMI !

Quelques cas d'utilisations possibles

J2 Ansible appliqué à IBM i

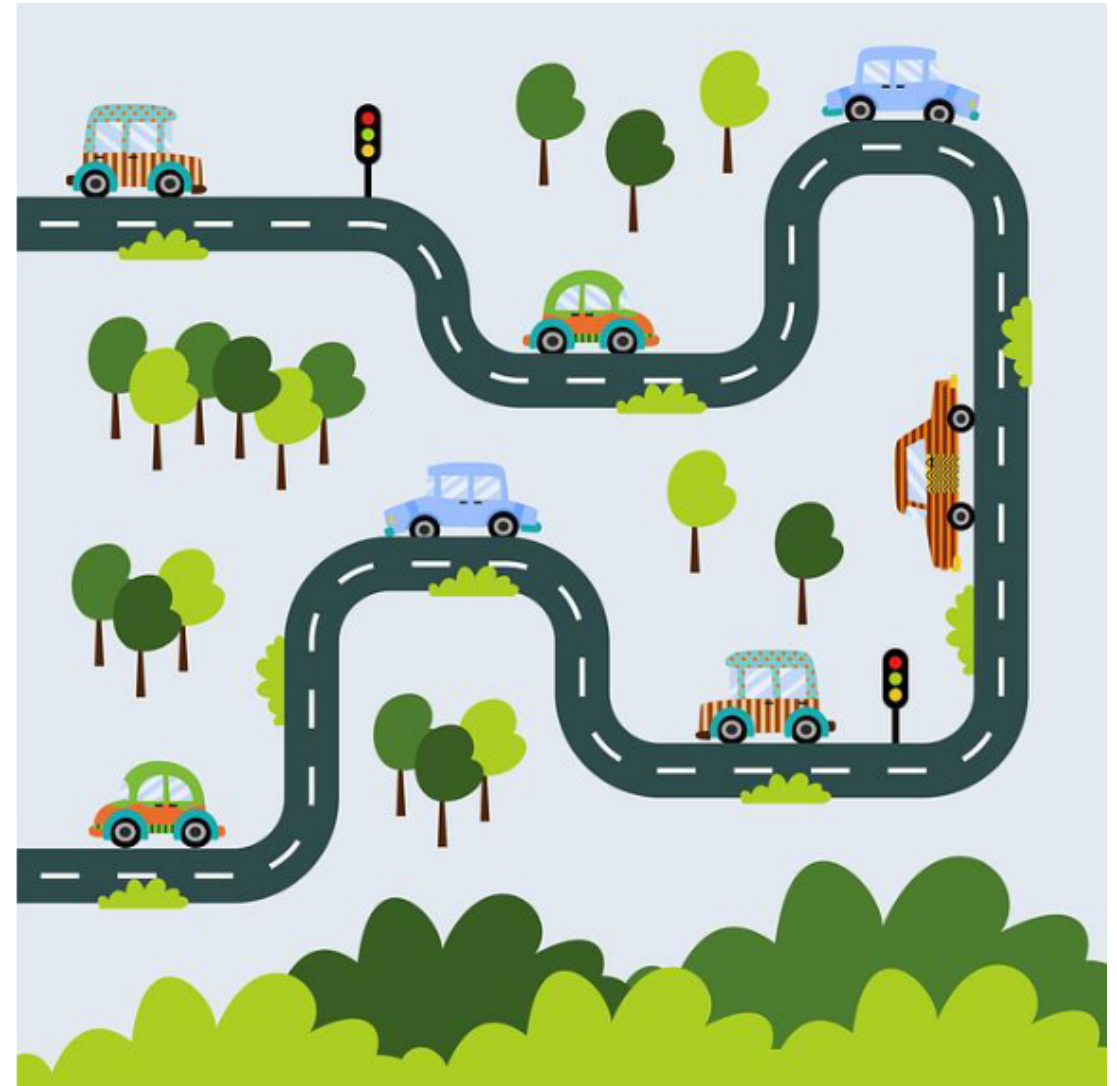
Ansible sur un IBMi ! sérieux ?

Adaptation des Playbooks pour IBM i

Les modules Ansible pour IBM i

Les variables et des faits pour IBM i

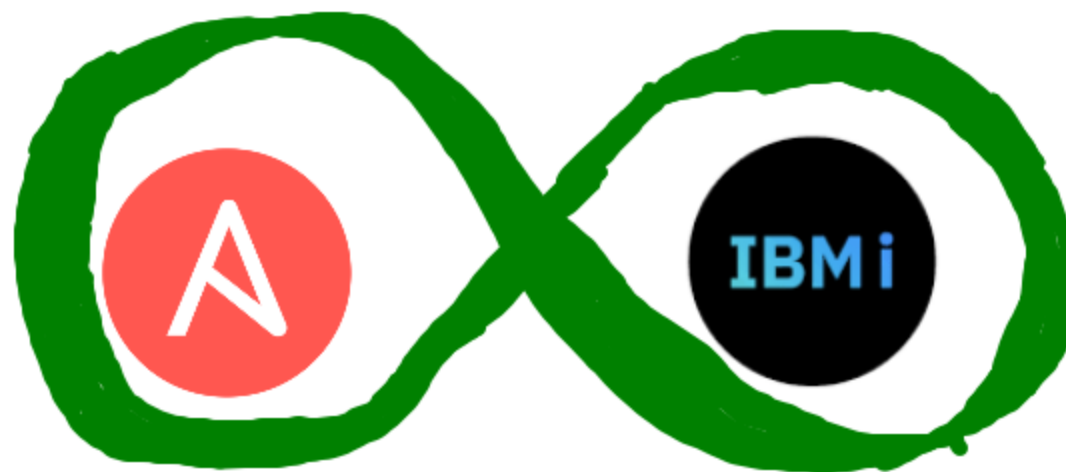
Quelques cas d'utilisation



Module 7 - Ansible pour IBM i

Déploiement d'Ansible dans un environnement IBM i

Ansible sur IBMi sérieux ?



Pourquoi utiliser Ansible pour IBM i ?



Cohérence

- IBMi n'est plus seul....
- SI: Linux, Windows, IBMi, cloud.....

Pourquoi utiliser Ansible pour IBM i ?



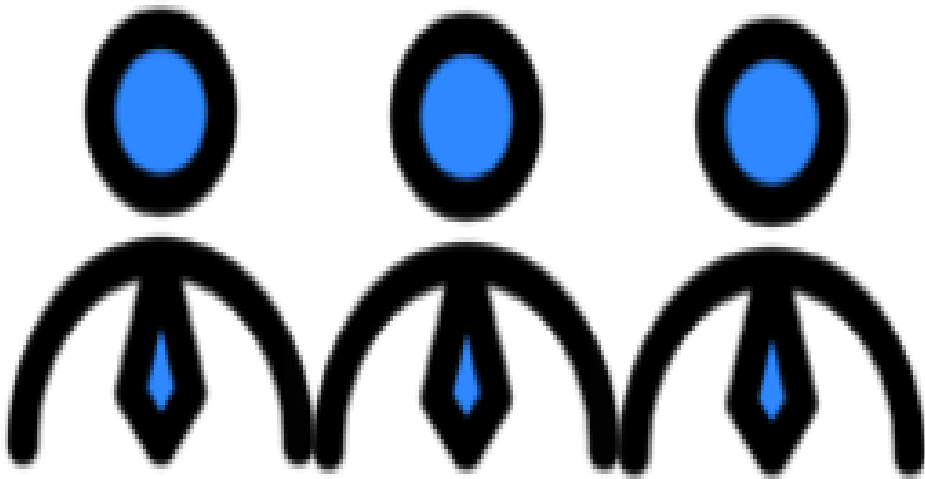
Transparence

- Infrastructure as a code....
- Bonnes pratiques.....

Pourquoi utiliser Ansible pour IBM i ?

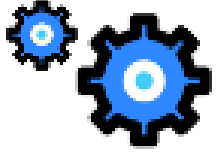
Compétences

- 🚀 Ansible,python
- Tout le SI

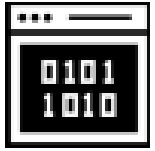




Provisionnement



Configuration



Déploiement



CI/CD



Sécurité



Orchestration

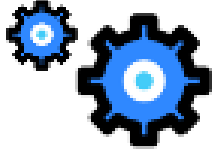
Cas d'utilisation à IBM i

Tâches courantes de l'administrateur IBM i

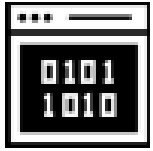
- Installation de correctifs et maintenance du système (PTF)
- Déploiement d'applications et de programmes
- Gestion du travail IBM i
- Gestion de la sécurité
- et autres tâches IBM i courantes
-



Provisionnement



Configuration



Déploiement



CI/CD



Sécurité



Orchestration

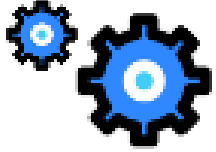
Cas d'utilisation à IBM i

Tâches courantes de développement IBM i

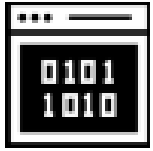
- Développement et test continus
- Automatisation de la construction, des tests unitaires, du processus de déploiement, etc.
- Gestion des environnements
-



Provisionnement



Configuration



Déploiement



CI/CD



Sécurité



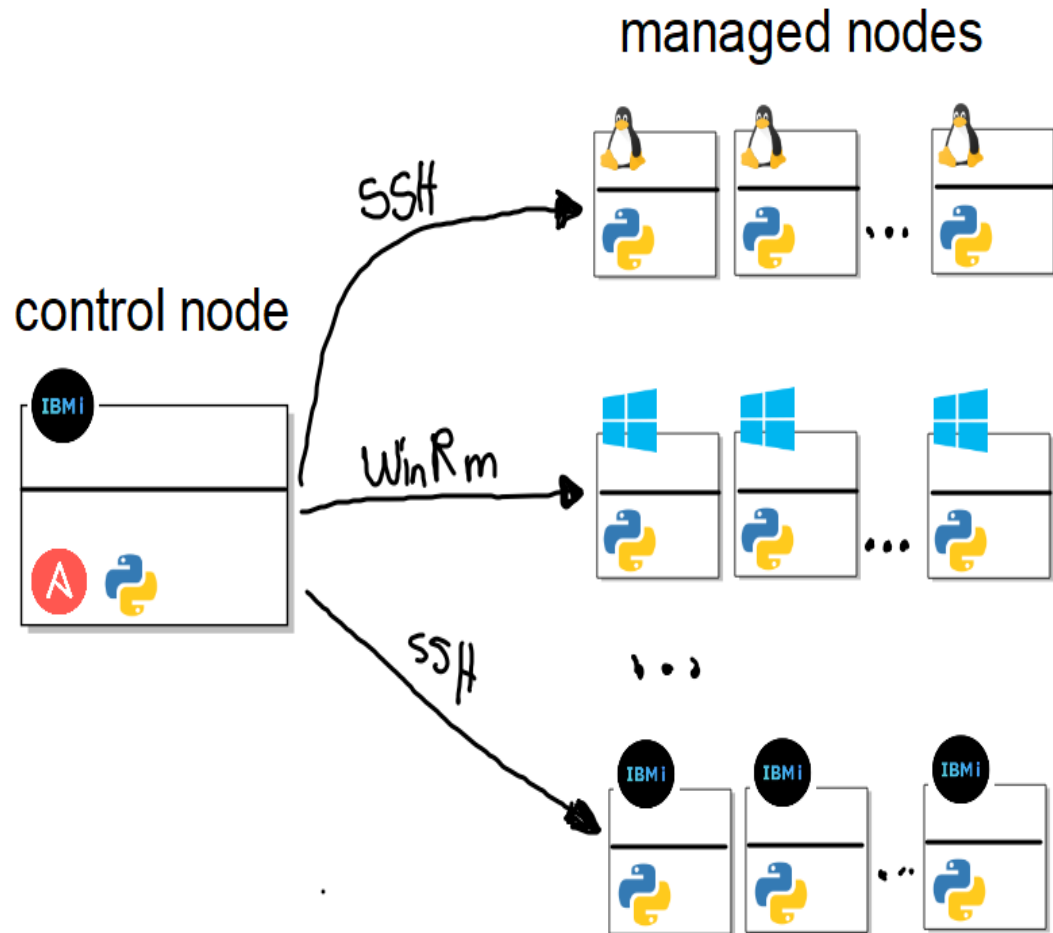
Orchestration

Cas d'utilisation à IBM i

Tâches liées à l'informatique Cloud IBM i

- Approvisionnement et la configuration des VM dans les plateformes IBM cloud
- Orchestrations pour former des solutions cloud
-

Comment fonctionne Ansible avec IBM i ?



collection IBMi

```
ansible-galaxy collection install ibm.power_ibmi
```

🔍 contenu de la collection.

```
tree /home/novy/.ansible/collections/ansible_collections/ibm
```

first on IBMi

TP05 hello IBMi

Objectifs :

- Mise en place .profile et .bashrc
- Installation de visual studio code et code for IBMi
- Creation d'un inventaire en localhost (group_vars python3)
- Test via le CLI
- Appeler un playbook créé au chapitre précédant pour voir ...
- Analyser les résultats



Questions ?

Quizz

intro prochain chapitre

Module 8 : La Collection Ansible pour IBM i

La Collection Ansible pour IBM i



collection IBMi

contenu

🔍 contenu de la collection.

```
tree /home/novy/.ansible/collections/ansible_collections/ibm
```


Installer la Collection

installation

```
ansible-galaxy collection install ibm.power_ibmi
```

mettre à jour

```
ansible-galaxy collection install ibm.power_ibmi -f  
ansible-galaxy collection install ibm.power_ibmi --upgrade
```

configurer

ansible.cfg

```
[defaults]  
collections_path=~/.ansible/collections:/usr/share/ansible/collections
```

tester

```
ansible localhost -m ibm.power_ibmi.ibm_sysval -a "sysvalue={'name':'qccsid'}"
```

Utiliser

```
---
- hosts: all
  gather_facts: no
  collections:
    - ibm.power_ibmi

tasks:
- block:
    - name: Display a system value
      ibmi_sysval:
        sysvalue:
          - {'name': 'qccsid'}
      register: dspsysval_ccsid_result
      tags: display, sysval
```

00_inventory.yml

```
all: # keys must be unique, i.e. only one 'hosts' per group
  hosts:
    volubis:
      ansible_host: as400.gaia.lan
    neptune:
      ansible_host: neptune.gaia.lan
    itest9:
      ansible_host: itest9.gaia.lan
    armonie:
      ansible_host: 178.255.128.61
```

group_vars/all/variable.yml

```
ansible_python_interpreter: /QOpenSys/pkgs/bin/python3.6
```

du vrai IBMi !

TP06 galaxyForIBMi

Objectifs :

- Installation de la Collection Ansible pour IBM i
- Configuration de la Collection Ansible pour IBM i
- Test via le CLI => dspsysval ??
- modifier le Playbook precedant miniCMDB pour afficher le ccsid.
- Analyser les résultats



Questions ?

Quizz

intro prochain chapitre

Module 9 - Les modules Ansible pour IBM i

Exploration des modules Ansible dédiés à IBM i

Les modules Ansible pour IBM i

Power IBM i collection for Ansible

Object Management

- ibmi_copy
- ibmi_fetch
- ibmi_lib_restore
- ibmi_lib_save
- ibmi_object_authority
- ibmi_object_find
- ibmi_object_restore
- ibmi_object_save
- ibmi_sync
- ibmi_sync_files
- ibmi_synchronize
- ibmi_synchronize_files

Fix Management

- ibmi_display_fix
- ibmi_download_fix
- ibmi_fix
- ibmi_fix_group_check
- ibmi_fix_imgclg
- ibmi_fix_repo
- ibmi_install_product_from_savf
- ibmi_save_product_to_savf
- ibmi_uninstall_product

IASP Management

- ibmi_device_vary
- ibmi_get_nonconfigure_disks
- ibmi_iasp

Network

- ibmi_ethernet_port
- ibmi_nrg_link
- ibmi_tcp_interface
- ibmi_tcp_server_service

Command Support

- ibmi_cl_command
- ibmi_rtv_command
- ibmi_script
- ibmi_script_execute
- ibmi_sql_execute
- ibmi_sql_query

Work Management

- ibmi_at
- ibmi_display_subsystem
- ibmi_end_subsystem
- ibmi_host_server_service
- ibmi_job
- ibmi_message
- ibmi_query_job_log
- ibmi_reboot
- ibmi_reply_message
- ibmi_start_subsystem
- ibmi_submit_job

Security

- ibmi_sysval
- ibmi_user_and_group
- ibmi_user_compliance_check

Executer un commande cl

```
- name: Créer une library avec la commande CL CRTLIB
  ibm.power_ibmi.ibm_cl_command:
    cmd: 'CRTLIB LIB(TESTLIB)'
    become_user: 'USER1'
    become_user_password: 'yourpassword'
```

ibmi_cl_command

Exemple Playbook

Executer une requete SQL

```
- name: Query the data of table Persons.  
  ibmi_ibm.power_ibmi.ibm_sql_query:  
    sql: 'select * from Persons'  
    become_user: 'USER1'  
    become_user_password: 'yourpassword'
```

ibmi_sql_query

Exemple Playbook

Récolter les faits spécifiques IBMi.

- name: Return ibmi_facts
ibm.power_ibmi.ibmi_facts:
- name: Assert a fact returned by ibmi_facts
assert:
 that:
 - system_name == 'DB2MB1PA'

ibmi_facts

```
---  
- hosts: all  
  gather_facts: yes  
  collections:  
    - ibm.power_ibmi  
  
  tasks:  
    - name: return facts IBMi  
      ibmi_facts:  
  
    - name: affichage des facts IBMi  
      debug:  
        var: ansible_facts
```

Utiliser les exemples de github

TP07_utliserGithubExemples

Objectifs :

- utiliser l'exemple [ibmi-sysval-sample.yml](#)
- utiliser l'exemple [ibmi-cl-command-sample.yml](#) en utilisant une variable d'inventaire `nomLIBTravail` pour créer le nom de la bibliothèque



Questions ?

Quizz

intro prochain chapitre

Module 10 : un cas d'utilisation

Audit de la sécurité. pas d'ajustement ou

IBMI services miniCMDB ..



Questions ?

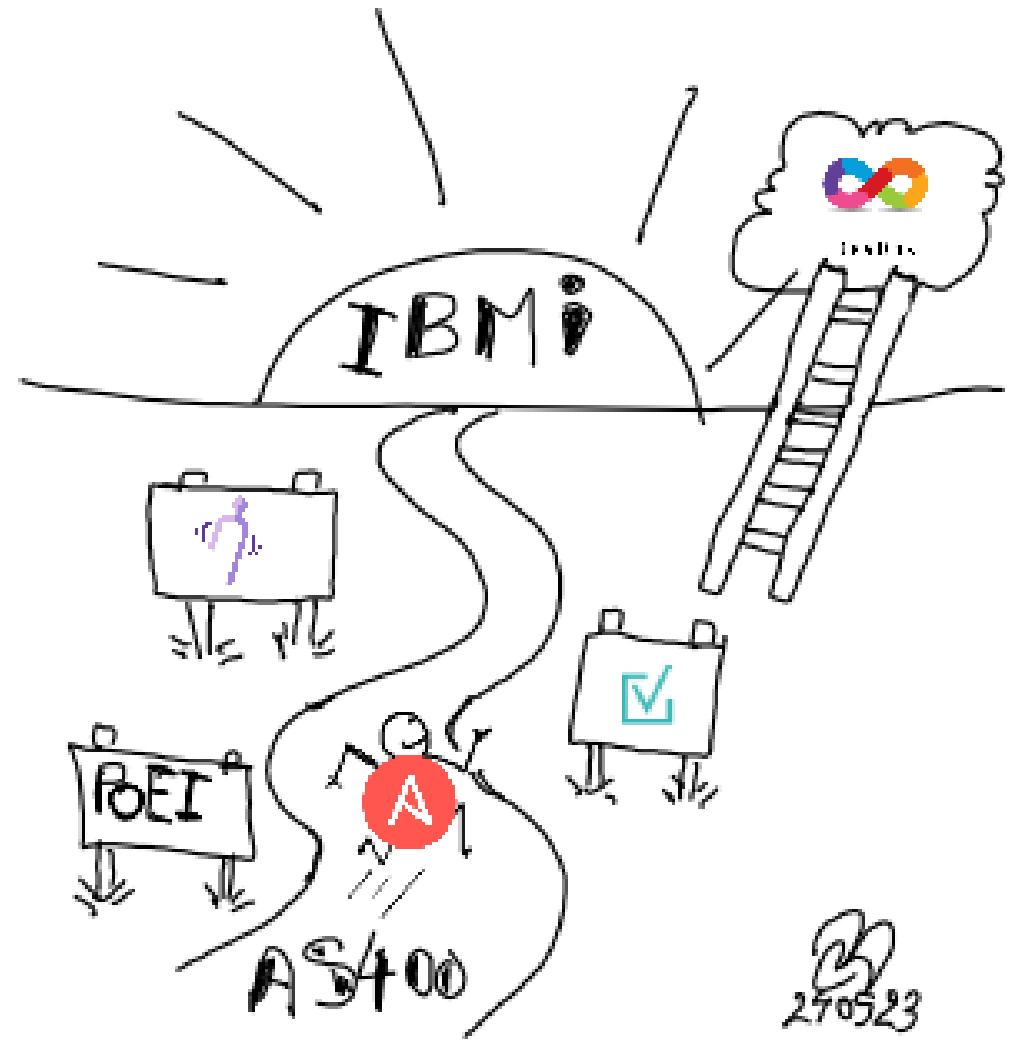
Quizz

Qu'avez vous pensé de cette formation ?

Un grand merci à tous

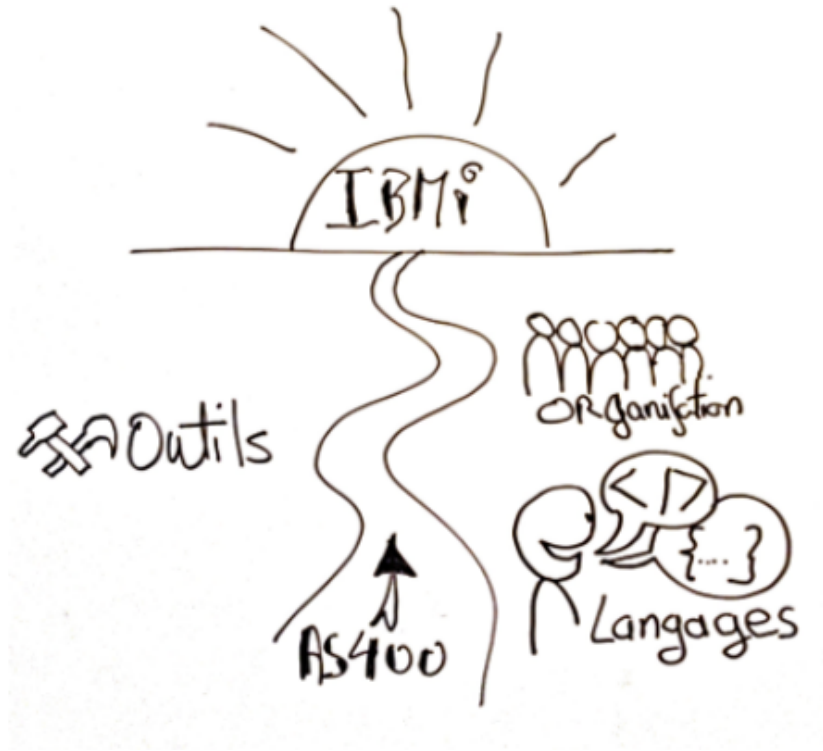
Qu'avez vous pensé de cette formation ?

Echanges , questions....



N'hésitez pas !....

les routes de l'IBMi



 yvon vieville

 06 80 14 45 17

