

# Tutorial 5 Utiliser Ansible sur IBMi.

## Table des matières

- [Objectifs](#)
- [Ressources](#)
- [Pré-Requis](#)

## Objectifs

Dans ce TP ,nous allons utiliser Ansible sur notre IBMi. Contrairement au TP précédant nous allons spécialiser nos tâches à réaliser en installant des modules spécifiques à l'IBMi via la collection disponible sur galaxy.

Ainsi nous allons :

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

## Ressources

- Environnement
- Temps : 60 mn.

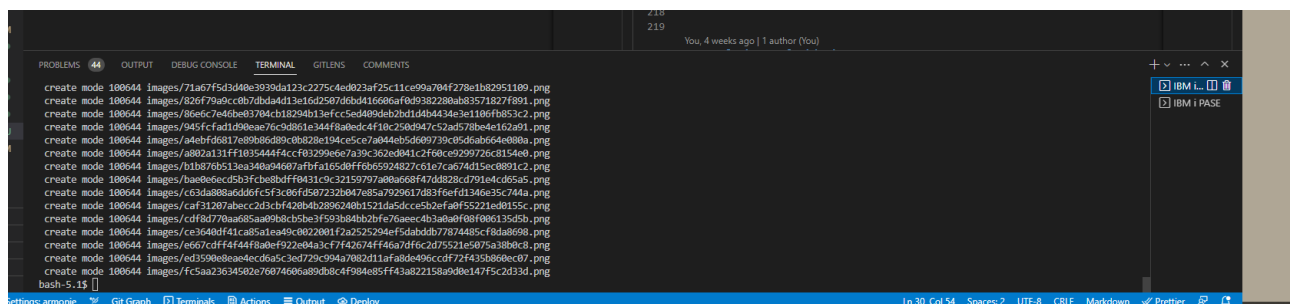
## Pré-Requis

- avoir configurer Ansible pour travailler avec notre l'environnement sur l'IBMi.  
[TP05 - Hello IBMi](#)

## Énoncé

### Etape 1 Installer la Collection Ansible pour IBM i.

1. lancer la commande suivante dans un terminal IBMi.

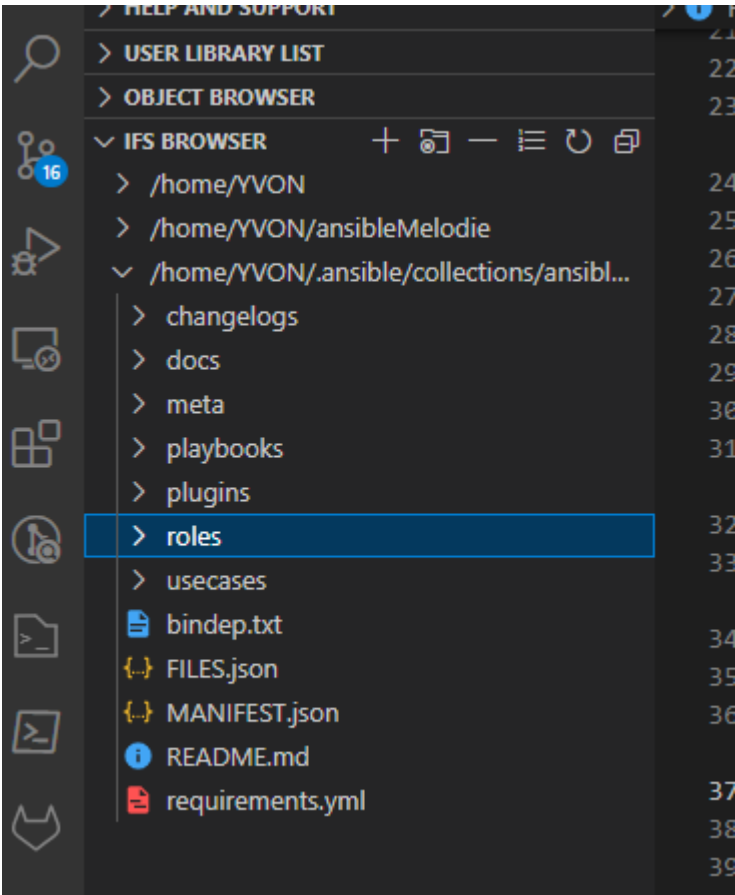
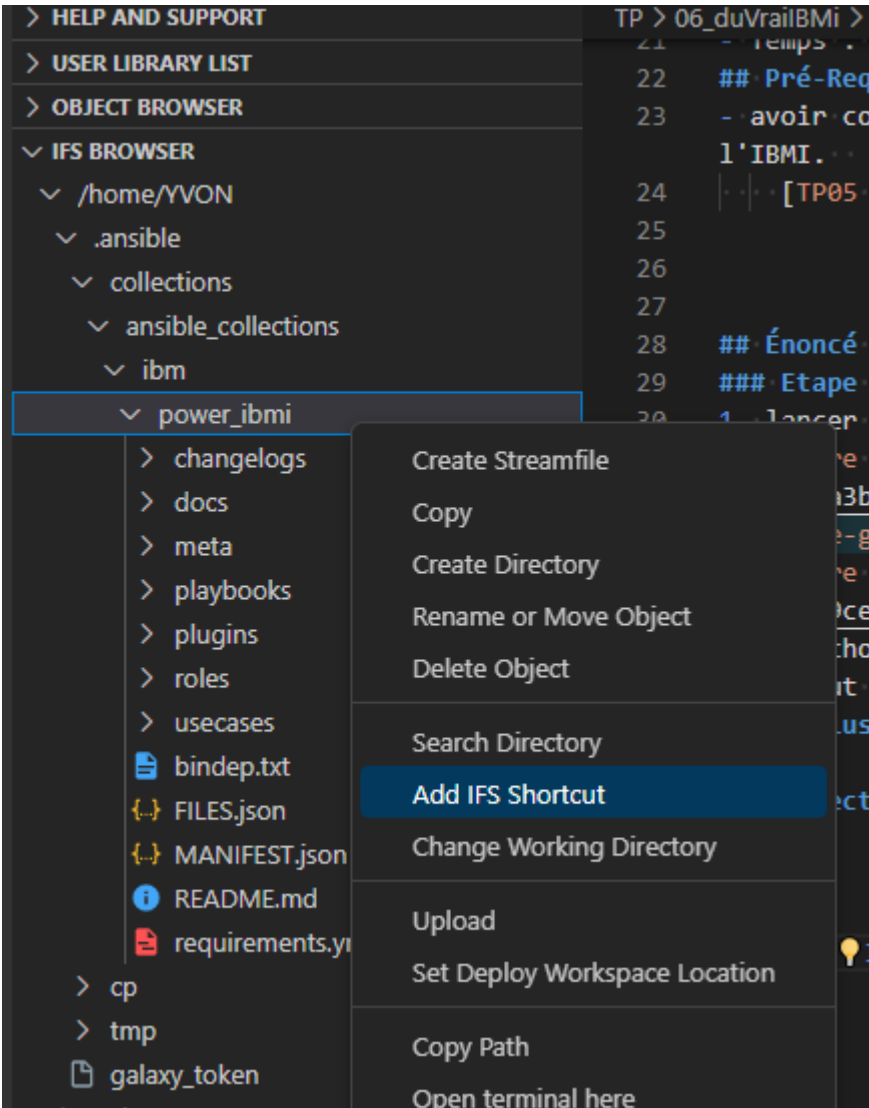


```
create mode 100644 images/71a07f5d3d40e3939da123c22754e0923af75c11cc99a704f278e1b02951109.png
create mode 100644 images/826f79a0ccdb74d3da4113e162507d0d416696af0b938220bab3571827f891.png
create mode 100644 images/86efc7e46be03704cb18294b13efcc5ed409deb2bd144b4434e3e1106fb83c2.png
create mode 100644 images/945fcfad1d90eae76c9d861e344f8a8edc4f10c250d947c52ad578be4e162a91.png
create mode 100644 images/a4ebfd681e89b68d89cb828e194ce5ce7a044eb5d609739c05d5ab664e080a.png
create mode 100644 images/a802a131ff1035444f4ccf03299e6e7a39c362ed041c2f60ce9299726c8154e0.png
create mode 100644 images/b1b0760512a340a94607a7f3f1059ff6b6924827c5107c0674d15ec8091c2.png
create mode 100644 images/bae8e6ecd5b3fcb8bdf0831c9c32159707a80a68f47d828cd791e4cd65a5.png
create mode 100644 images/c63da88a6dd6fc5f3c06fd507232b047e85a7923617d83f6efd1346e35c744a.png
create mode 100644 images/caf31207abecc2d3cbf420b4b2896240b1521da5dccc5b2efa0f55221ed0155c.png
create mode 100644 images/cdf8d77baa685aa9b8cb5be3f593b84bb2bf676aee4b3a0a0f08f006135d5b.png
create mode 100644 images/ce3640df41ca85a1ea49c0d280172a2525294ef5dabdb77674485cf8da8698.png
create mode 100644 images/e607cdf1f44f8a8e922e04a3c7142674ff6a0dfc2d75321e5095a386c8.png
create mode 100644 images/ed3590e8ea4ecd9a5c3ed72c994a7082d11afabde409ccdf72f435b86ec07.png
create mode 100644 images/fc5aa23634502e76074606a89db8c4f984e85ff43a822158a0d0e147f5c2d33d.png
bash-5.1$
```

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

```
create mode 100044 images/eb598eb0ac4c00a3c3ed729c94a7082011a7a0de490cc072145b000ec07.png
create mode 100644 images/fc5aa23634502e76074606a89db8c4f984e85ff43a822158a9d0e147f5c2d33d.png
bash-5.1$ ansible-galaxy collection install ibm.power_ibmi
Process install dependency map
Starting collection install process
Installing 'ibm.power_ibmi:1.9.1' to '/home/YVON/.ansible/collections/ansible_collections/ibm/power_ibmi'
bash-5.1$
```

2. affichons le contenu de cette collection dans le browser IFS de C4I Cela peut être pratique de ce créer dans le même temps un raccourci.



3. testons en appelant un module de cette collection via le CLI d'ansible. `ansible localhost -m ibm.power_ibmi.ibm_sysval -a "sysvalue={'name':'qccsid'}"`

```
bash-5.1$ ansible-galaxy collection install ibm.power_ibmi
Process install dependency map
Starting collection install process
Installing 'ibm.power_ibmi:1.9.1' to '/home/YVON/.ansible/collections/ansible_collections/ibm/power_ibmi'
bash-5.1$ ansible localhost -m ibm.power_ibmi.ibm_sysval -a "sysvalue={'name':'qccsid'}"
[WARNING]: No inventory was parsed, only implicit localhost is available
localhost | SUCCESS => {
  "changed": false,
  "fail_list": [],
  "job_log": [],
  "message": "",
  "rc": 0,
  "sysval": [
    {
      "msg": "+++ success QSYS QWCRSVAL ",
      "name": "QCCSID",
      "rc": 0,
      "type": "10i0",
      "value": "65535"
    }
  ]
}
```

Etape 2 utiliser la Collection Ansible pour IBM i dans un playbook.

1. Créons le dossier TP06 dans myWork en recopiant le dossier TP05.
2. Dans le playbook 10\_premierPlaybook.yml, ajoutons l'usage des modules de notre collection

```
---
- name: tests de notre configuration.
  hosts: all
  tasks:
    - name: test de la connexion
      ping:
    - name: affichage d'un message avec ma variable.
      debug:
        msg: Le contenu de ma variable est {{ maVariable }}.
    - name: ajout du dossier.
      file:
        state: directory
        path: ~/tmp
...

```

devient

```
---
- name: tests de notre configuration.
  hosts: all
  collections:
    - ibm.power_ibmi
  tasks:
    - name: test de la connexion
      ping:

```

```

- name: affichage d'un message avec ma variable.
  debug:
    msg: Le contenu de ma variable est {{ maVariable }}.
- name: ajout du dossier.
  file:
    state: directory
    path: ~/tmp
...

```

- testons `ansible-playbook 10_premierPlaybook.yml -i 00_inventory.yml` ok mais rien de neuf ! mais on n'a rien cassé !

```

bash-5.1$ ansible-playbook 10_premierPlaybook.yml -i 00_inventory.yml

PLAY [tests de notre configuration.] *****

TASK [Gathering Facts] *****
ok: [armonie]

TASK [test de la connexion] *****
ok: [armonie]

TASK [affichage d'un message avec ma variable.] *****
ok: [armonie] => {
  "msg": "Le contenu de ma variable est all."
}

TASK [ajout du dossier.] *****
ok: [armonie]

PLAY RECAP *****
armonie                : ok=4    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

bash-5.1$

```

1. ajoutons une tâche utilisant un module de l'IBMi.

```

- name: Display a system value
  ibmi_sysval:
    sysvalue:
      - {'name':'qccsid'}
  register: dspsysval_ccsid_result
- name: affichage du resultat.
  debug:
    msg: Le contenu du dspsysval est {{ dspsysval_ccsid_result }}.
- name: affichage du CCSID.
  debug:
    msg: Le contenu du CCSID est {{ dspsysval_ccsid_result.sysval[0].value }}.

```

```

TASK [affichage d'un message avec ma variable.] *****
ok: [armonie] => {
  "msg": "Le contenu de ma variable est all."
}

TASK [ajout du dossier.] *****
ok: [armonie]

TASK [Display a system value] *****
ok: [armonie]

TASK [affichage du resultat.] *****
ok: [armonie] => {
  "msg": "Le contenu du dspysval est {'rc': 0, 'message': '', 'sysval': [{'rc': 0, 'name': 'QCCSID', 'type': '1010', 'msg': '+++ success QSYS QMCRSVAL ', 'value': '65535'}], 'fail_list': [], 'job_log' : [], 'failed': False, 'changed': False}."
}

TASK [affichage du CCSID.] *****
ok: [armonie] => {
  "msg": "Le contenu du CCSID est 65535."
}

PLAY RECAP *****
armonie      : ok=7   changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

```

## correction

Etape 2 adapter notre miniCMDB pour afficher aussi le CCSID.

1. modifier le playbook 20\_miniCMDB.yml

- Ajout de la collection IBMi.

```

---
- name: creation d'un dossier de travail
  hosts: all
  collections:
    - ibm.power_ibmi

```

- Ajout de la tâche collectant le CCSID.

```

- name: Display a system value
  ibmi_sysval:
    sysvalue:
      - {'name':'qccsid'}
  register: dspysval_ccsid_result

```

juste avant la tache de Génération du rapport en markdown

-testons `ansible-playbook 20_miniCMDB.yml -i 00_inventory.yml` ok mais notre variable n'est pas reportée dans notre rapport ?

1. ajoutons le CCSID dans notre rapport

- modification de notre template (./TP06/templates/report.md.j2)

```

# Rapport du système {{ inventory_hostname }}

| CCSID | Architecture | OS | Nom du node |
| :----- | :-----: | :-----: | :-----: |
| {{ dspysval_ccsid_result.sysval[0].value }} | {{ ansible_architecture }}

```

```
| {{ ansible_distribution }} {{ ansible_distribution_release }} | {{
ansible_nodename }} |
```

1. lançons et vérifions.

```
}

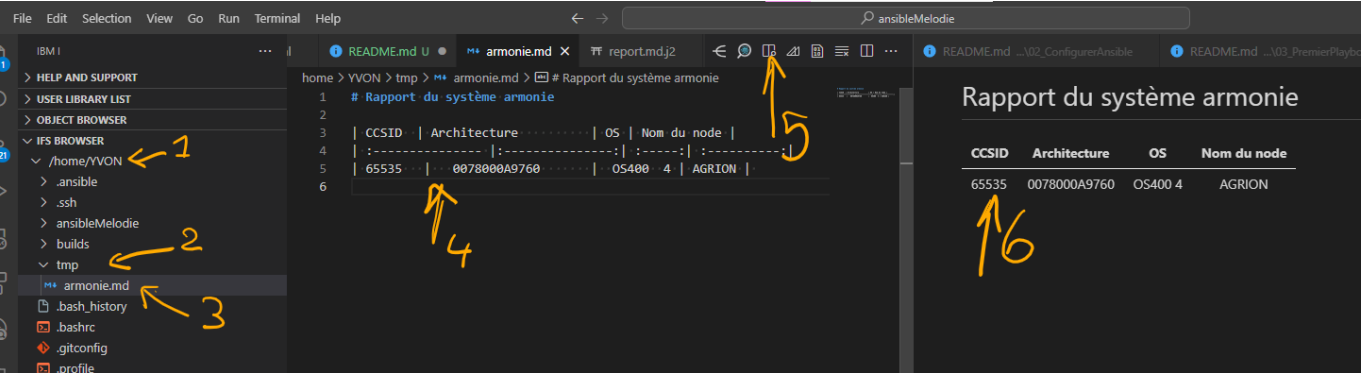
TASK [affichage du nom de host] *****
ok: [armonie] => {
  "ansible_hostname": "AGRION"
}

TASK [Display a system value] *****
ok: [armonie]

TASK [Génération du rapport en markdown] *****
changed: [armonie]

PLAY RECAP *****
armonie                : ok=7   changed=3   unreachable=0   failed=0   skipped=0   rescued=0   ignored=0

bash-5.1$
```



correction

Conclusion et feed-back

Correction

💡💡💡💡 Idées

-