IBM i 10

Ansible

Comment démarrer rapidement

Benoit MAROLLEAU – Cloud Architect
IBM Client Engineering, Montpellier, France
benoit.marolleau@fr.ibm.com

common
FRANCE
IBM

Agenda

Introduction

Ansible Overview

- Architecture, Engine, Tower
- Ansible for IBM i

Terminology – How Ansible works

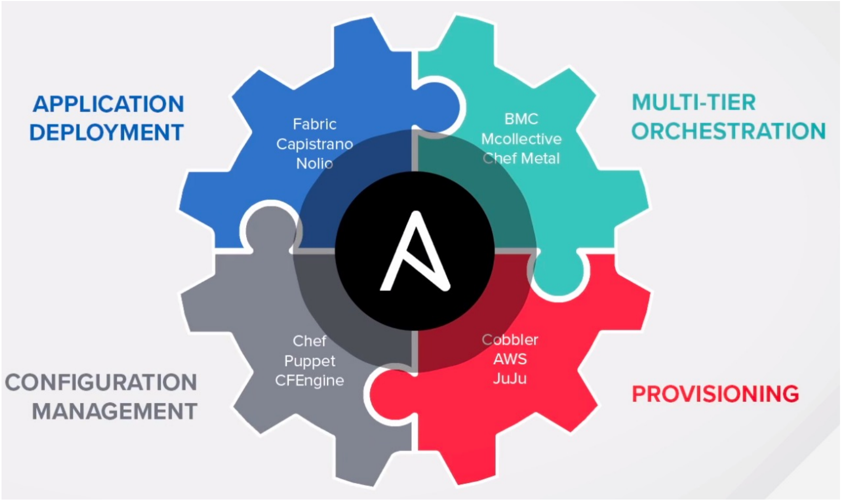
- Inventory
- Configuration file
- Modules
- Playbooks and Roles

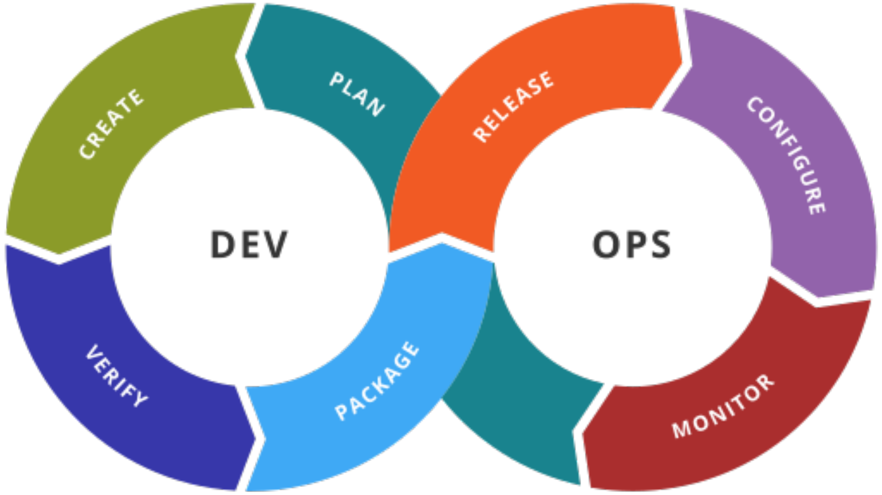
Next Steps : LABS

Ansible Overview

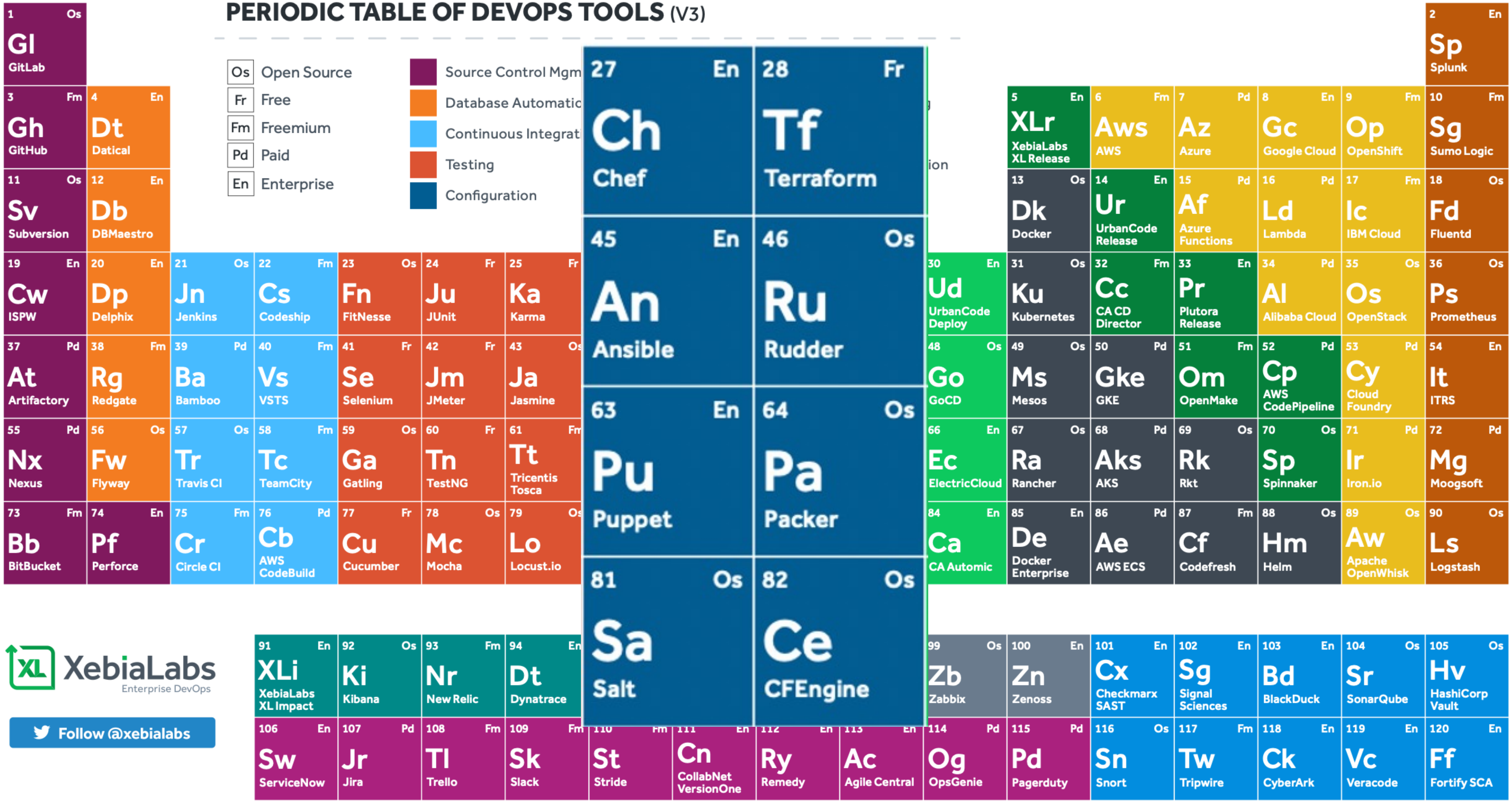
“Ansible is an open source automation tool for provisioning, orchestration, system configuration and patching”

First developed by Michael DeHaan and acquired by Red Hat in 2015.





Ansible Overview



PERIODIC TABLE OF DEVOPS TOOLS (V3)

Os	Open Source
Fr	Free
Fm	Freemium
Pd	Paid
En	Enterprise

- Source Control Mgm
- Database Automatic
- Continuous Integrat
- Testing
- Configuration

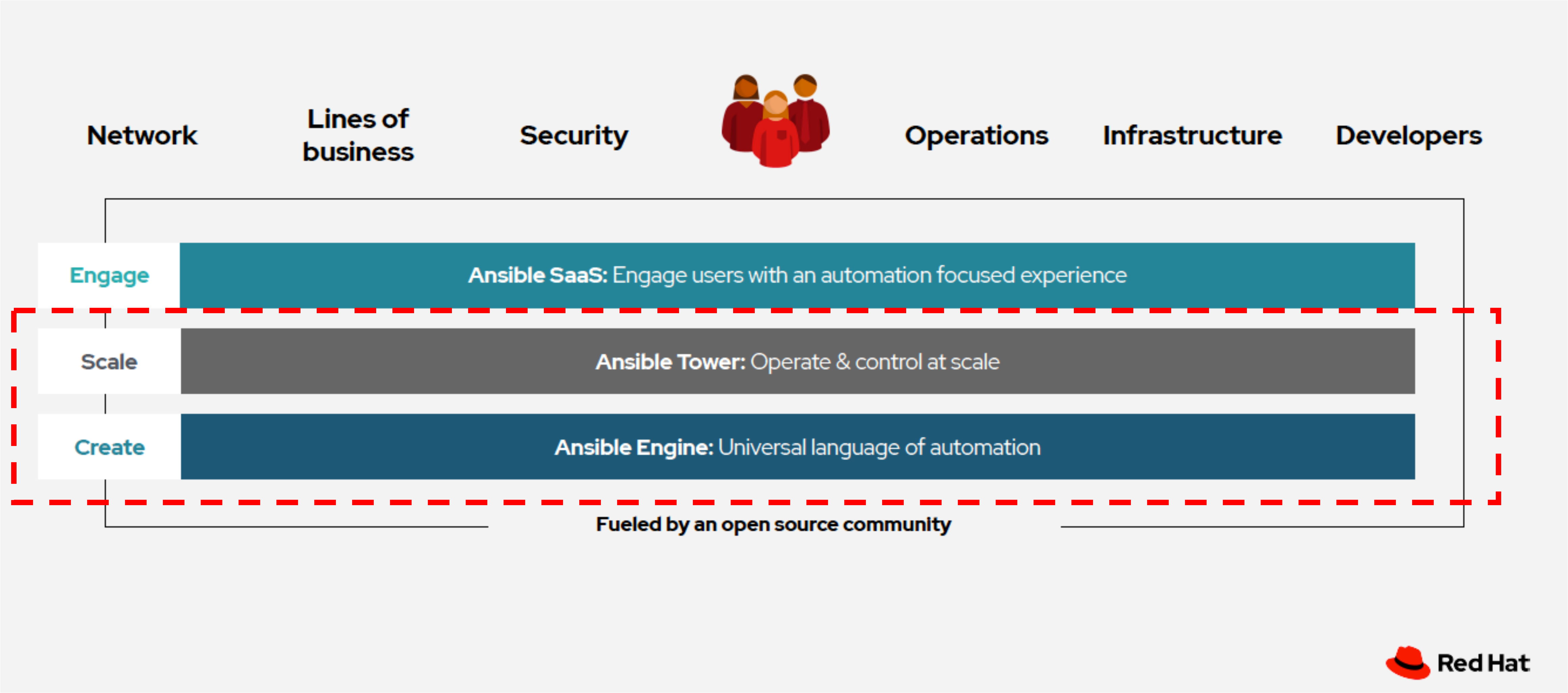
ion

No.	Licence	Symbol	Name
1	Os	Gl	GitLab
2	En	Sp	Splunk
3	Fm	Gh	GitHub
4	En	Dt	Datical
5	En	XLr	XebiaLabs XL Release
6	Fm	Aws	AWS
7	Pd	Az	Azure
8	En	Gc	Google Cloud
9	Fm	Op	OpenShift
10	Fm	Sg	Sumo Logic
11	Os	Sv	Subversion
12	En	Db	DBMaestro
13	Os	Dk	Docker
14	En	Ur	UrbanCode Release
15	Pd	Af	Azure Functions
16	Pd	Ld	Lambda
17	Fm	Ic	IBM Cloud
18	Os	Fd	Fluentd
19	En	Cw	ISPW
20	En	Dp	Delphix
21	Os	Jn	Jenkins
22	Fm	Cs	Codeship
23	Os	Fn	FitNesse
24	Fr	Ju	JUnit
25	Fr	Ka	Karma
27	En	Ch	Chef
28	Fr	Tf	Terraform
30	En	Ud	UrbanCode Deploy
31	Os	Ku	Kubernetes
32	Fm	Cc	CA CD Director
33	En	Pr	Plutora Release
34	Pd	Al	Alibaba Cloud
35	Os	Os	OpenStack
36	Os	Ps	Prometheus
37	Pd	At	Artifactory
38	Fm	Rg	Redgate
39	Pd	Ba	Bamboo
40	Fm	Vs	VSTS
41	Fr	Se	Selenium
42	Fr	Jm	JMeter
43	Os	Ja	Jasmine
45	En	An	Ansible
46	Os	Ru	Rudder
48	Os	Go	GoCD
49	Os	Ms	Mesos
50	Pd	Gke	GKE
51	Fm	Om	OpenMake
52	Pd	Cp	AWS CodePipeline
53	Pd	Cy	Cloud Foundry
54	En	It	ITRS
55	Pd	Nx	Nexus
56	Os	Fw	Flyway
57	Os	Tr	Travis CI
58	Fm	Tc	TeamCity
59	Os	Ga	Gatling
60	Fr	Tn	TestNG
61	Fm	Tt	Tricentis Tosca
63	En	Pu	Puppet
64	Os	Pa	Packer
66	En	Ec	ElectricCloud
67	Os	Ra	Rancher
68	Pd	Aks	AKS
69	Os	Rk	Rkt
70	Os	Sp	Spinnaker
71	Pd	Ir	Iron.io
72	Pd	Mg	Moogsoft
73	Fm	Bb	BitBucket
74	En	Pf	Perforce
75	Fm	Cr	Circle CI
76	Pd	Cb	AWS CodeBuild
77	Fr	Cu	Cucumber
78	Os	Mc	Mocha
79	Os	Lo	Locust.io
81	Os	Sa	Salt
82	Os	Ce	CFEngine
84	En	Ca	CA Automic
85	En	De	Docker Enterprise
86	Pd	Ae	AWS ECS
87	Fm	Cf	Codefresh
88	Os	Hm	Helm
89	Os	Aw	Apache OpenWhisk
90	Os	Ls	Logstash
91	En	XLi	XebiaLabs XL Impact
92	Os	Ki	Kibana
93	Fm	Nr	New Relic
94	En	Dt	Dynatrace
99	Os	Zb	Zabbix
100	En	Zn	Zenoss
101	En	Cx	Checkmarx SAST
102	En	Sg	Signal Sciences
103	En	Bd	BlackDuck
104	Os	Sr	SonarQube
105	Os	Hv	HashiCorp Vault
106	En	Sw	ServiceNow
107	Pd	Jr	Jira
108	Fm	Tl	Trello
109	Fm	Sk	Slack
110	Fm	St	Stride
111	En	Cn	CollabNet VersionOne
112	En	Ry	Remedy
113	En	Ac	Agile Central
114	Pd	Og	OpsGenie
115	Pd	Pd	Pagerduty
116	Os	Sn	Snort
117	Fm	Tw	Tripwire
118	En	Ck	CyberArk
119	En	Vc	Veracode
120	En	Ff	Fortify SCA

XebiaLabs
Enterprise DevOps

Follow @xebialabs

Ansible Overview



Network

Lines of business

Security

Operations

Infrastructure

Developers

Engage	**Ansible SaaS:** Engage users with an automation focused experience
Scale	**Ansible Tower:** Operate & control at scale
Create	**Ansible Engine:** Universal language of automation

Fueled by an open source community

Red Hat

Red Hat Ansible CERTIFIED INTEGRATION: **Ansible and IBM Power Systems**

https://www.ansible.com/integrations/infrastructure/ibm-power-systems

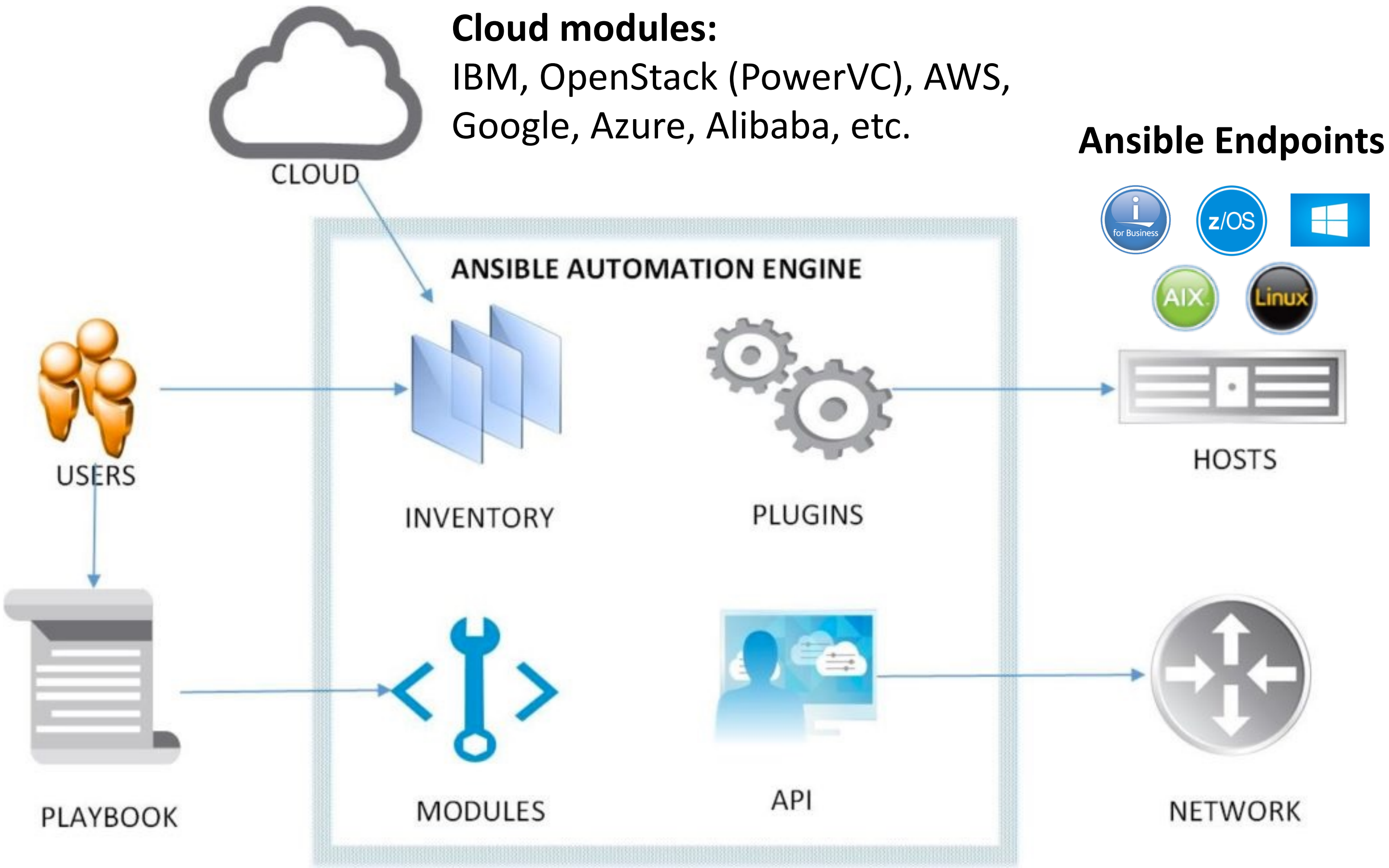
What is Red Hat Ansible Engine?

Ansible Engine provides the core, agentless functionality of Ansible that everything else builds upon

Includes the basic building blocks of Ansible—the control node, managed nodes (endpoints), inventory, modules, tasks and playbooks

Commercial form of Ansible technology

Available for subscription purchase from Red Hat—from a POWER perspective, includes enterprise support options for AIX and IBM i managed endpoints (in June 2020)



Cloud modules:
IBM, OpenStack (PowerVC), AWS, Google, Azure, Alibaba, etc.

Ansible Endpoints

RED HAT® ANSIBLE® Engine

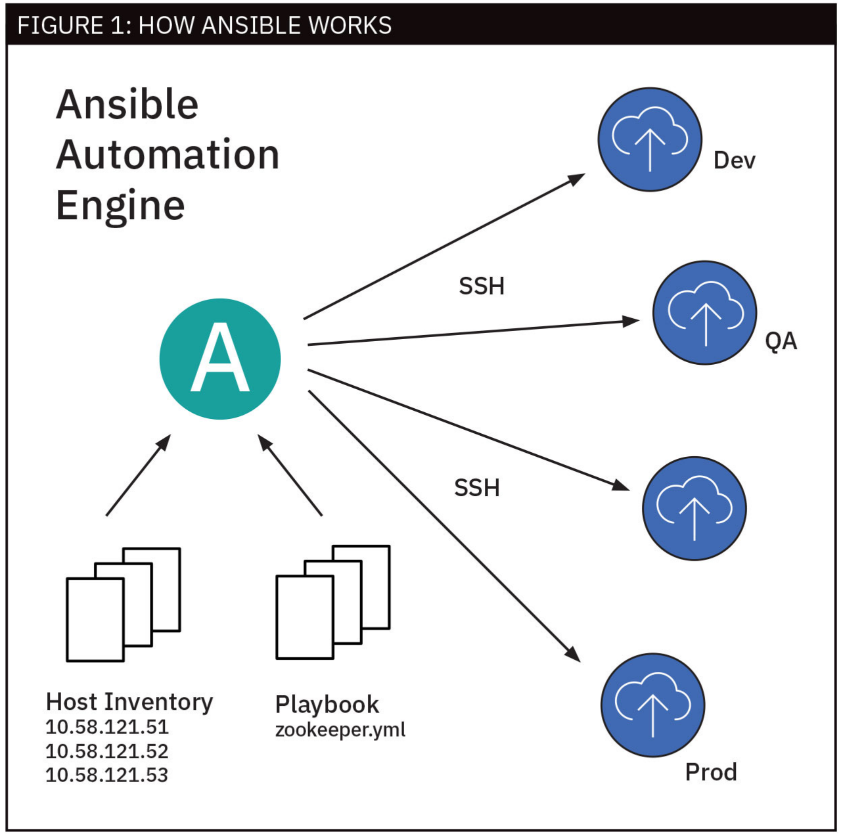
Red Hat Ansible Engine supported on x86 Linux only — manages to endpoints

Introduction to Ansible

Ansible is a radically simple IT automation platform that makes your applications and systems easier to deploy.

- Free open source application
- **Agent-less** – No need for agent installation and management
- Python/YAML based
- Highly flexible and configuration management of systems.
- Configuration roll-back in case of error

FIGURE 1: HOW ANSIBLE WORKS



Introduction to Ansible

ANSIBLE

for Business

Control node – any machine with Ansible installed and is used to run playbooks

Managed node (a.k.a. endpoints) – endpoint devices (e.g., AIX, IBM i, Linux, Windows, etc.) that are managed with Ansible

Inventory – a list of managed nodes so that Ansible understands the overall IT landscape

Modules – units of code that Ansible executes; hundreds of modules out-of-box; thousands of community modules available

Tasks – units of action in Ansible
(invoke a set of modules to do something useful)

Playbooks – ordered list of tasks and written in YAML

Ansible and IBM i

Write your first “playbook” in YAML format to describe what you want on your managed node inventory and Ansible will , for example :

✓ Deploy or clone a new environment on an IBM i VM on either a private or public cloud

✓ Install a new licensed program product or application version containing libraries, database and IFS artifacts

✓ Save or restore objects, manage servers or jobs and check and install PTFs

✓ Control your security settings, like managing user profiles and authorities, or check IFS rights. Ansible gathers facts and can remediate any security deviations.

✓ Orchestrate all of the above or a subset of these tasks

Ansible and IBM i

Core modules in PASE + IBM i Specific Modules

Core Maintained modules are maintained by the Ansible Engineering Team.

- Core modules are owned by RedHat and ship with Ansible installation.
- Many of these modules work for IBM i PASE environment.
- Support PASE but not native IBM i.

- command
- raw
- script
- shell
- pip
- yum
- pause
- wait_for_connection
- ~~at~~
- authorized_key
- gather_facts
- ~~group~~
- ~~Mount~~

- ping
- ~~reboot~~
- setup
- ~~user~~
- assemble
- blockinfile
- copy
- fetch
- file
- find
- lineinfile
- stat
- synchronize
- git

Ansible and IBM i

Core modules in PASE + **IBM i Specific Modules**

- CL Commands
  Executes CL commands and return general and detail job logs
- SQLs executions
  Executes SQL statements and return the results
  Queries – compare the returned single value result
  Inserts / Updates / Deletes
  Functions & Procedures
- Gathering facts and setup changes for IBM i
- Securities – authorization list, user profiles, grant object authorities
- Copy Objects, Fetch Objects, Find Objects
- Reply Message – query and reply
- Reboot system
- Network configurations
- Device configurations and management
- IASP configuration
- System Values, Environment variables, Etc.
- Submit / Schedule Jobs
- Manage fixes / PTFs / LPPs

- …. More to come!!! Check out

https://github.com/IBM/ansible-for-i

ibmi_at
Schedule a batch job on a remote IBMi node.

ibmi_cl_command
Executes a CL command.

ibmi_copy
Copy a save file from local to a remote IBMi node.

ibmi_display_subsystem
Display all currently active subsystems or currently active jobs in a subsystem.

ibmi_end_subsystem
End a subsystem.

ibmi_fetch
Fetch objects or a library from a remote IBMi node and store on local.

ibmi_install_product_from_savf
Install the the licensed program(product) from a save file.

ibmi_lib_restore
Restore one library on a remote IBMi node.

ibmi_lib_save
Save one libary on a remote IBMi node.

ibmi_object_authority
Grant, Revoke and Display the Object Authority.

ibmi_object_restore
Restore one or more objects on a remote IBMi node.

Ibmi_object_save
Save one or more objects on a remote IBMi node.

ibmi_reboot
Reboot IBMi machine.

ibmi_save_product_to_savf
Save the the licensed program(product) to a save file.

ibmi_script
Execute a local cl/sql script file on a remote ibm i node.

ibmi_script_execute
Execute a cl/sql script file on a remote ibm i node.

ibmi_sql_execute
Executes a SQL non-DQL(Data Query Language) statement.

ibmi_sql_query
Executes a SQL DQL(Data Query Language) statement.

ibmi_start_subsystem
Start a subsystem.

ibmi_sync
Synchronize a save file from current ibm i node A to another ibm i node B.

ibmi_synchronize
Synchronize a save file from ibm i node A to another ibm i node B.

ibmi_uninstall_product
Delete the objects that make up the licensed program(product).

ibmi_user_and_group
Create, Change and Display a user(or group) profile.

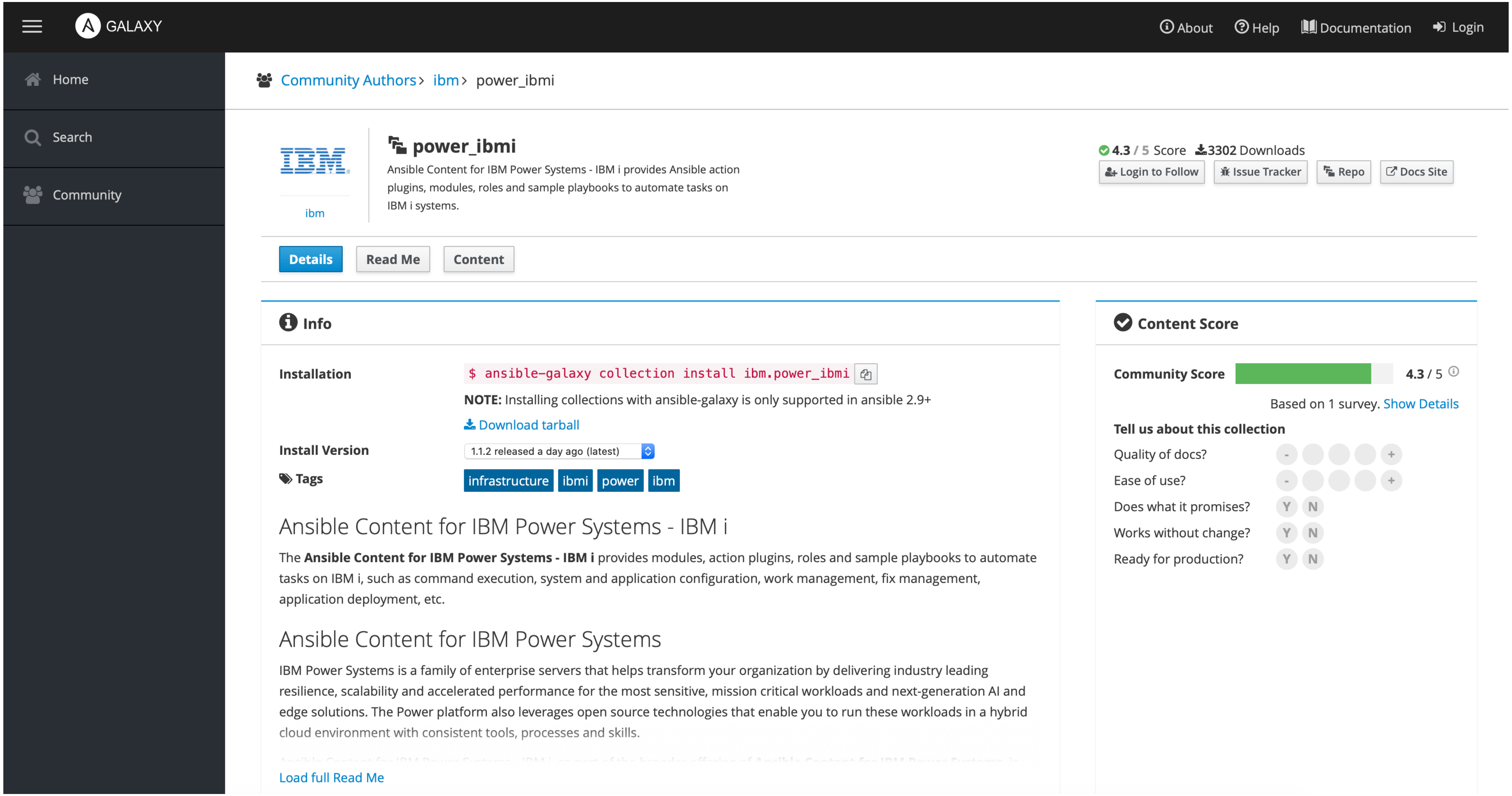
Ansible and IBM i

Playbooks Examples

- enable-ansible-for-i
  - ibmi-install-rpm.yml
  - ibm-install-yum.yml
  - setup.yml
- ibmi-install-nodejs
  - ibmi-install-nodejs.yml
- ibmi-check-default-passwords.yml
- ibmi-cl-command-sample.yml
- ibmi-fix-group-check.yml
- ibmi-fix-repo-cum-package.yml
- ibmi-sysval-sample.yml
- query-iasp-sample.yml
- ibmi-sql-sample.yml

https://github.com/IBM/ansible-for-i

Galaxy – power_ibmi



power_ibmi

Ansible Content for IBM Power Systems - IBM i provides Ansible action plugins, modules, roles and sample playbooks to automate tasks on IBM i systems.

Info

Installation

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

NOTE: Installing collections with ansible-galaxy is only supported in ansible 2.9+

Download tarball

Install Version 1.1.2 released a day ago (latest)

Tags infrastructure ibmi power ibm

Ansible Content for IBM Power Systems - IBM i

The **Ansible Content for IBM Power Systems - IBM i** provides modules, action plugins, roles and sample playbooks to automate tasks on IBM i, such as command execution, system and application configuration, work management, fix management, application deployment, etc.

Ansible Content for IBM Power Systems

IBM Power Systems is a family of enterprise servers that helps transform your organization by delivering industry leading resilience, scalability and accelerated performance for the most sensitive, mission critical workloads and next-generation AI and edge solutions. The Power platform also leverages open source technologies that enable you to run these workloads in a hybrid cloud environment with consistent tools, processes and skills.

Ansible Content for IBM Power Systems - IBM i, as part of the broader offering of **Ansible Content for IBM Power Systems**, is

Load full Read Me

Content Score

Community Score 4.3 / 5

Based on 1 survey. Show Details

Tell us about this collection

Quality of docs? - ○ ○ ○ +

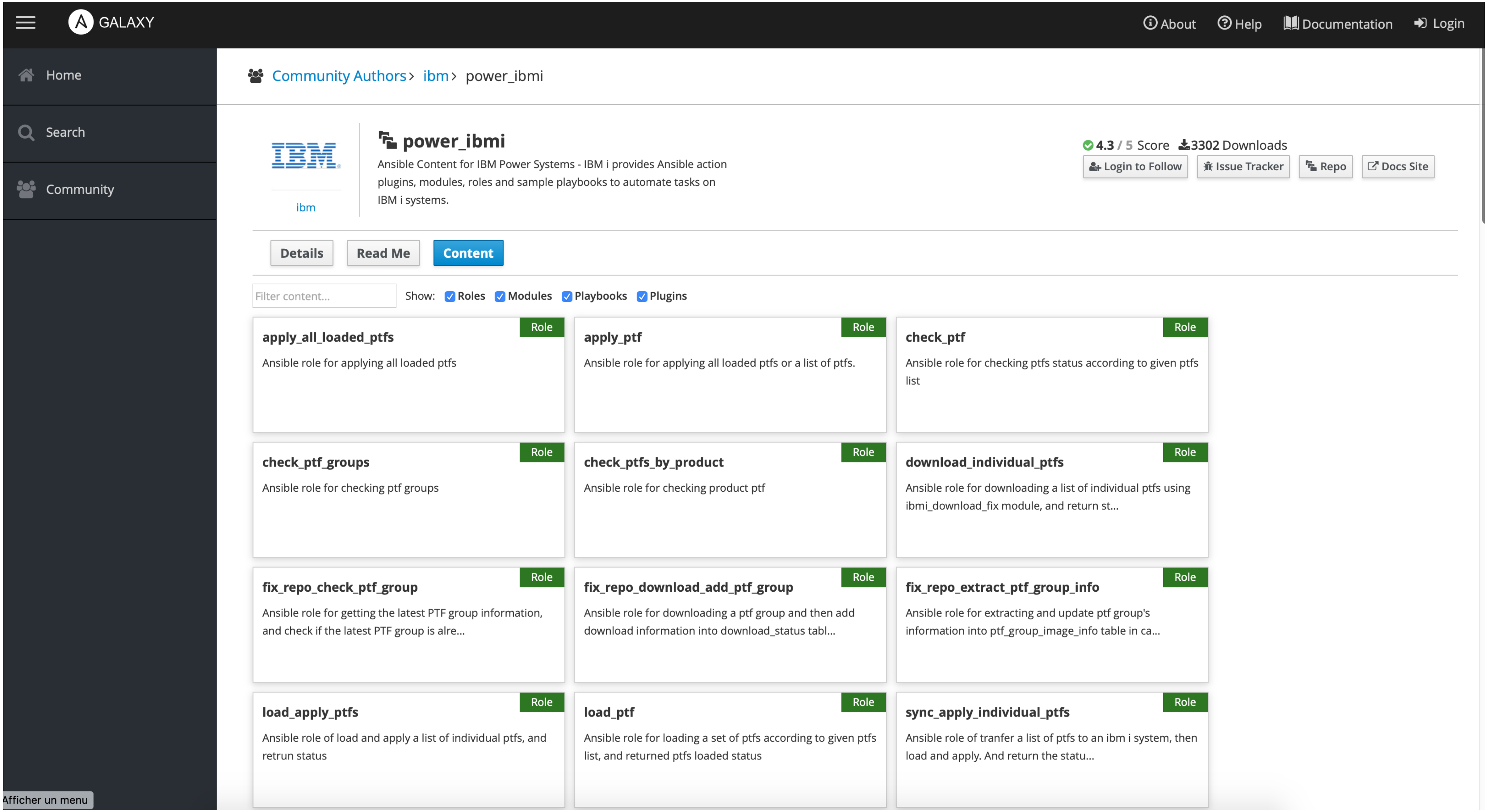
Ease of use? - ○ ○ ○ +

Does what it promises? Y N

Works without change? Y N

Ready for production? Y N

Galaxy – power_ibmi



Community Authors › ibm › power_ibmi

ibm

power_ibmi

Ansible Content for IBM Power Systems - IBM i provides Ansible action plugins, modules, roles and sample playbooks to automate tasks on IBM i systems.

4.3 / 5 Score 3302 Downloads

Show: ☑ Roles ☑ Modules ☑ Playbooks ☑ Plugins

apply_all_loaded_ptfs Role
Ansible role for applying all loaded ptfs

apply_ptf Role
Ansible role for applying all loaded ptfs or a list of ptfs.

check_ptf Role
Ansible role for checking ptfs status according to given ptfs list

check_ptf_groups Role
Ansible role for checking ptf groups

check_ptfs_by_product Role
Ansible role for checking product ptf

download_individual_ptfs Role
Ansible role for downloading a list of individual ptfs using ibmi_download_fix module, and return st...

fix_repo_check_ptf_group Role
Ansible role for getting the latest PTF group information, and check if the latest PTF group is alre...

fix_repo_download_add_ptf_group Role
Ansible role for downloading a ptf group and then add download information into download_status tabl...

fix_repo_extract_ptf_group_info Role
Ansible role for extracting and update ptf group's information into ptf_group_image_info table in ca...

load_apply_ptfs Role
Ansible role of load and apply a list of individual ptfs, and retrun status

load_ptf Role
Ansible role for loading a set of ptfs according to given ptfs list, and returned ptfs loaded status

sync_apply_individual_ptfs Role
Ansible role of tranfer a list of ptfs to an ibm i system, then load and apply. And return the statu...

Ansible Support & Installation

Control node

https://ibm.github.io/ansible-for-i/installation.html

- Ansible on Linux (x86/Power) : Community + Possible Red Hat Subscription and support

- Ansible on IBM i : Community + Possible IBM TSS Support (Open Source package)

- Ansible can be installed via your Linux distribution package manager
  - yum install ansible or apt install ansible
  - If not available, just install python-pip and dependencies and install it with “pip”
    - pip install ansible

- Clone the repository to your Ansible server (or install IBM i Galaxy)
  - https://github.com/IBM/ansible-for-i

- Create your inventory file
  - example can be found in file examples/ibmi/host_ibmi.ini

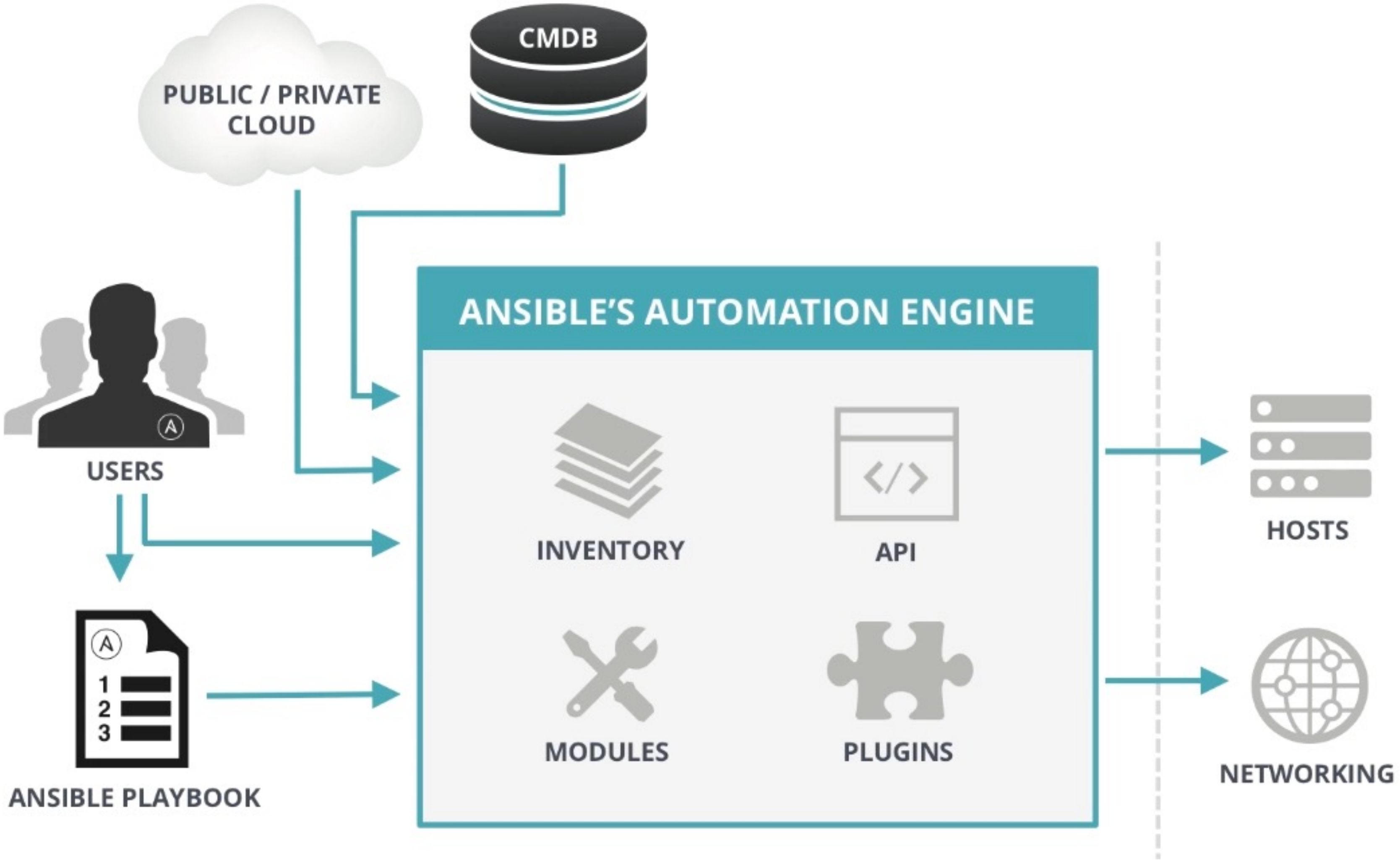
Ansible Overview – key points

ANSIBLE

1. Ansible Engine can manage a large number of clients (via an inventory)
2. It does not require an agent on the clients
3. Uses SSH to communicate with the clients
4. The clients can be AIX, IBM i, RHEL, Ubuntu, SLES, Centos, Fedora, network switches, storage controllers etc.....
5. Human readable automation
6. No special coding skills needed
7. Uses modules to perform tasks, these tasks can be called from the command line or playbooks
8. It is idempotent
9. Simple to get started

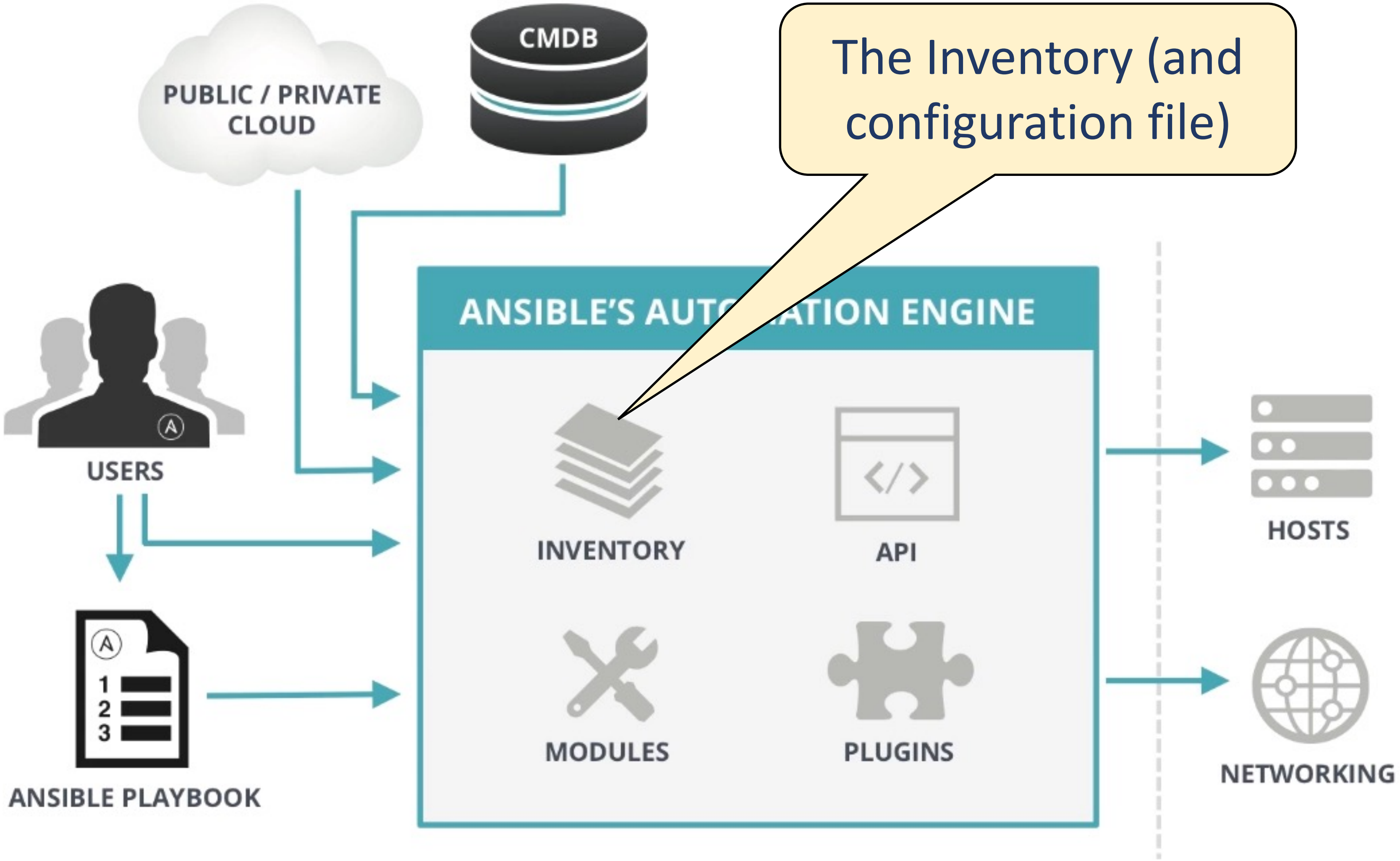
Architecture

1. Ansible Engine
2. Inventory
3. Modules
4. Playbooks
5. Client hosts



How Ansible works

1. Ansible Engine
2. Inventory
3. Modules
4. Playbooks
5. Client hosts

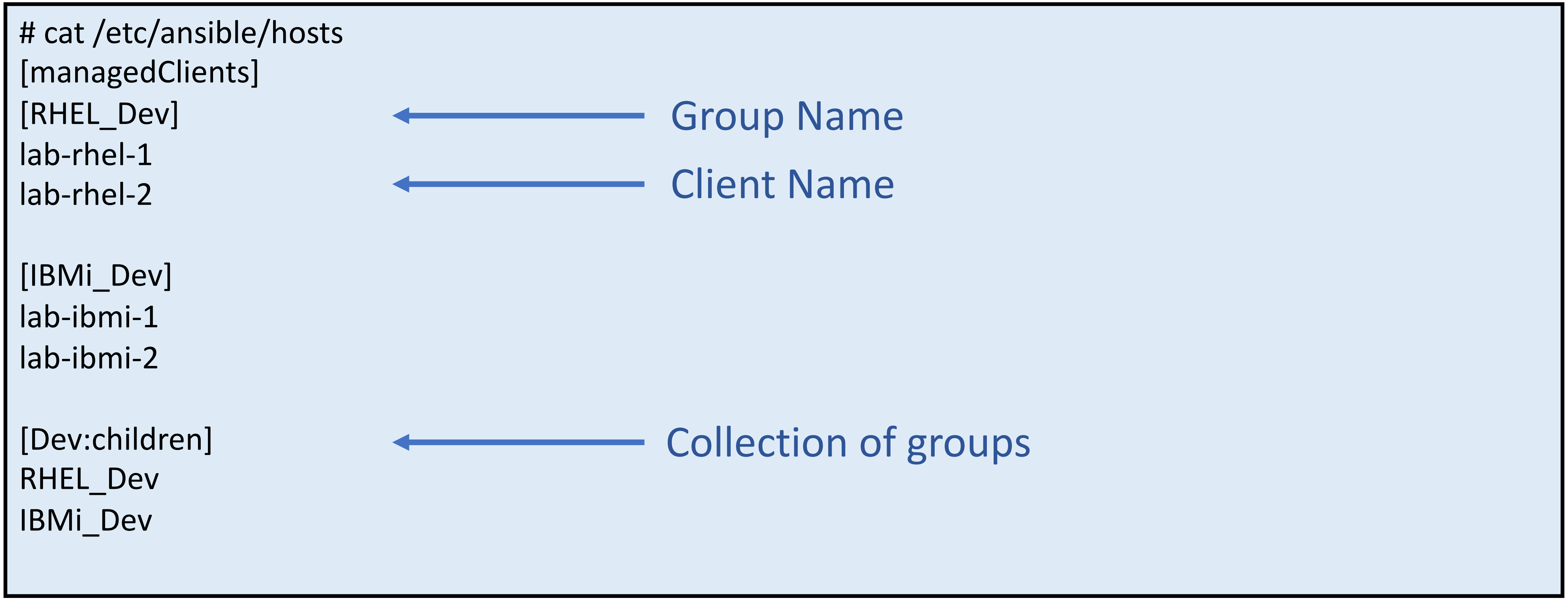


The Inventory (and configuration file)

How Ansible works – The Inventory

1. The client inventory file is a configurable list of VMs/clients that ansible can control.
2. It is written in an INI or YAML format, lists host and groups.
3. Can be static of dynamic.

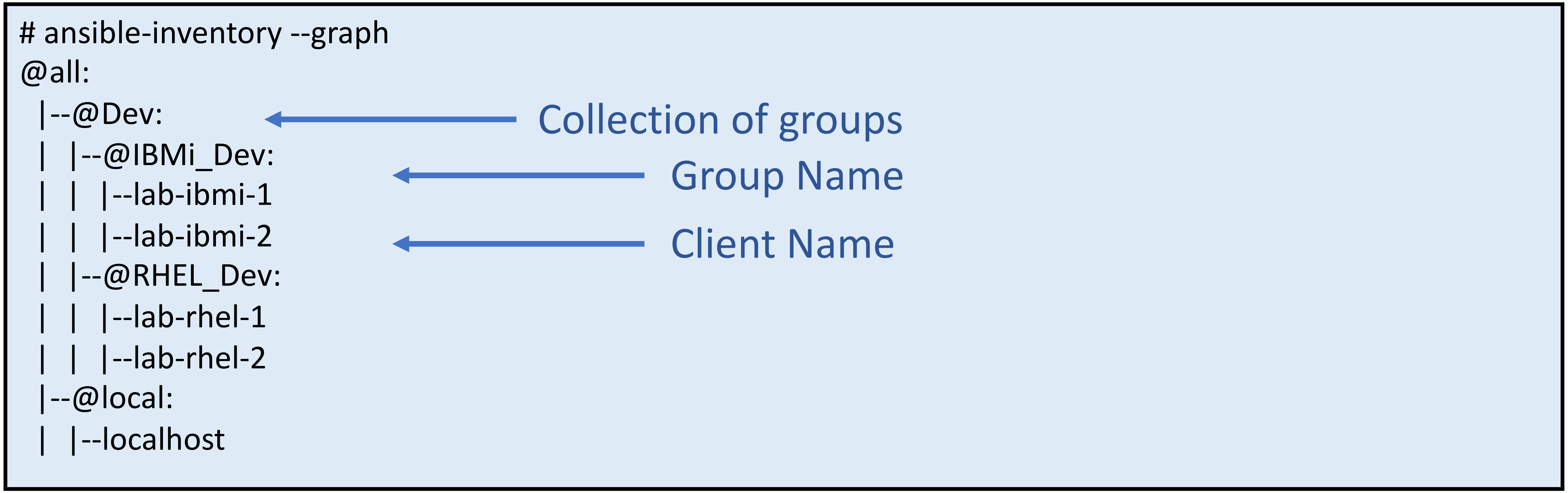
Static Inventory example



```
# cat /etc/ansible/hosts
[managedClients]
[RHEL_Dev]
lab-rhel-1
lab-rhel-2

[IBMi_Dev]
lab-ibmi-1
lab-ibmi-2

[Dev:children]
RHEL_Dev
IBMi_Dev
```

How Ansible works – The Inventory

So we can list the files in the inventory by using ‘ansible-inventory’



```
# ansible-inventory --graph
@all:
  |--@Dev:
  |  |--@IBMi_Dev:
  |  |  |--lab-ibmi-1
  |  |  |--lab-ibmi-2
  |  |--@RHEL_Dev:
  |  |  |--lab-rhel-1
  |  |  |--lab-rhel-2
  |--@local:
  |  |--localhost
```

Collection of groups

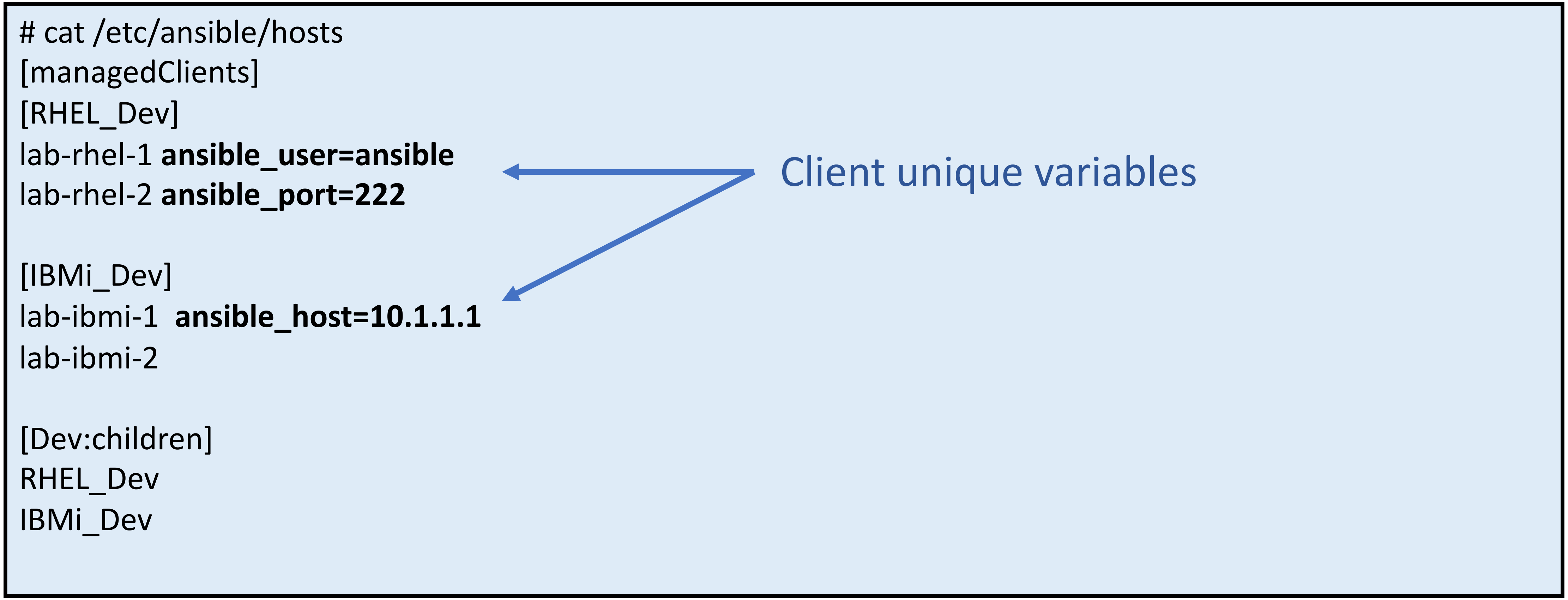
Group Name

Client Name

How Ansible works – The Inventory

We can use the inventory file to configure some connection options to the clients.

Static Inventory example with connection variables



```
# cat /etc/ansible/hosts
[managedClients]
[RHEL_Dev]
lab-rhel-1 ansible_user=ansible
lab-rhel-2 ansible_port=222

[IBMi_Dev]
lab-ibmi-1  ansible_host=10.1.1.1
lab-ibmi-2

[Dev:children]
RHEL_Dev
IBMi_Dev
```

Client unique variables

How Ansible works – The Inventory

We can use the inventory file to configure some connection options to the clients.

```
# ansible-inventory –list
….
 "hostvars": {
            "lab-ibmi-1": {
                "ansible_host": "10.1.1.1"
            },
            "lab-rhel-1": {
                "ansible_user": "ansible"
            },
            "lab-rhel-2": {
                "ansible_port": 222
            }
….
```

How Ansible works – The Inventory

We can use the inventory file to configure group connection options to the clients.

Static Inventory example with group connection variables

```
# cat /etc/ansible/hosts
[managedClients]
[RHEL_Dev]
lab-rhel-1 ansible_user=ansible
lab-rhel-2 ansible_port=222

[IBMi_Dev]
lab-ibmi-1  ansible_host=10.1.1.1
lab-ibmi-2

[Dev:children]
RHEL_Dev
IBMi_Dev

[IBMi_Dev:vars]
proxy=proxy.labs.uk.ibm.com
```

Variable applies to whole group

How Ansible works – The Inventory

We can use the inventory file to configure group connection options to the clients.

```
# ansible-inventory –list
….
 "hostvars": {
 "lab-ibmi-1": {
            "ansible_python_interpreter": "/QOpensys/pkgs/bin/python3",
            "ansible_ssh_common_args": "-o StrictHostKeyChecking=no",
            "ansible_ssh_user": "benoit"
         },
         "lab-ibmi-2": {
            "ansible_python_interpreter": "/QOpensys/pkgs/bin/python3",
            "ansible_ssh_common_args": "-o StrictHostKeyChecking=no",
            "ansible_ssh_user": "benoit"
         }
         },
         "lab-rhel-1": {
            "ansible_user": "ansible"
         },
         "lab-rhel-2": {
            "ansible_port": 222
         }
….
```

Both clients in the group have picked up the new connection variable

How Ansible works – The Inventory

We have a number of ways to tell Ansible which inventory file to use, in precedence:

1. the '-i' flag on the command line (you can call more than one inventory file if needed)
2. The ANSIBLE_INVENTORY environment variable
3. Using "inventory=xxx" in the ansible configuration file
4. If all else fails, the default is /etc/ansible/hosts

Method to check which inventory file you are using

```
# ansible -v -a "echo Inventory File is {{ inventory_file }}" localhost
Using /etc/ansible/ansible.cfg as config file
….
- Inventory
  - File
  - is
  - /etc/ansible/hosts
….
```

How Ansible works – The ansible config file

Ansible looks for a configuration file to determine a number of parameters. As with the inventory file, a number of configuration files can be defined for different projects.

Nearly all parameters in ansible.cfg can be overwritten in playbooks or during ansible calls.

Example ansible.cfg fie

```
# cat /etc/ansible/ansible.cfg
[defaults]
inventory      = /etc/ansible/hosts
library        = /usr/share/ansible/plugins/modules
module_utils   = /usr/share/my_module_utils/
remote_tmp     = ~/.ansible/tmp
local_tmp      = ~/.ansible/tmp
sudo_user      = root
ask_sudo_pass = True
ask_pass       = True
remote_port    = 22
…..
```

How Ansible works – The ansible config file

The active configuration files uses the following locations, in precedence:

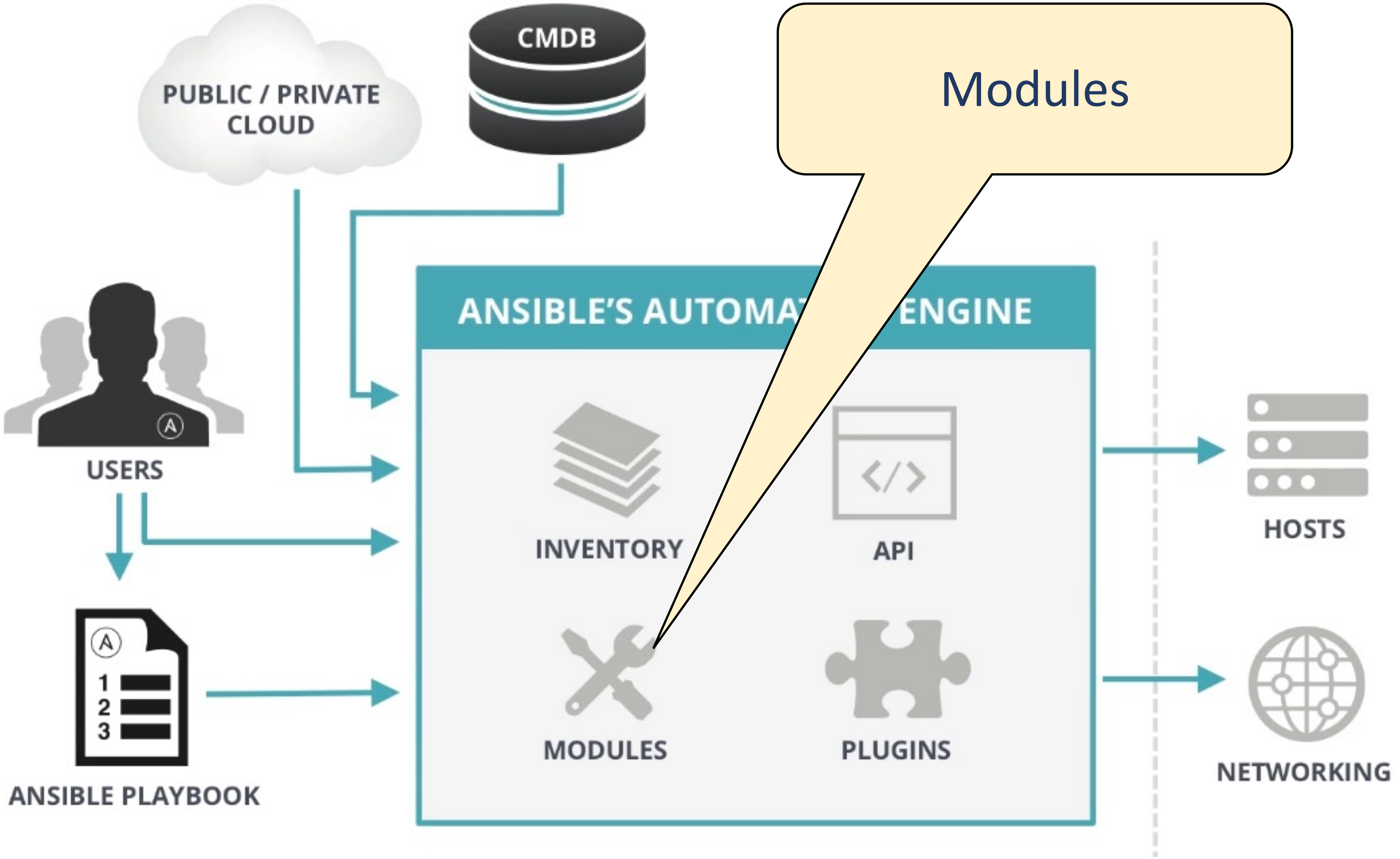
1. The ANSIBLE_CONFIG environment variable
2. ./ansible.cfg - within the current directory
3. ~/.ansible.cfg. - home directory
4. If all else fails, the default is /etc/ansible/ansible.cfg

Method to check which configuration file you are using

```
# ansible --version
ansible 2.9.6
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.5 (default, Jun 11 2019, 14:33:56) [GCC 4.8.5 20150623 (Red Hat 4.8.5-39)]
```

How Ansible works

1. Ansible Engine
2. Inventory
3. Modules
4. Playbooks
5. Client hosts



How Ansible works – Modules

ANSIBLE

Modules are the core of Ansible

1. They perform the real work by executing on the clients.
   - ✓ Ansible engine connects to your clients
   - ✓ It pushes out the module along with parameters
   - ✓ The module is then executed on the client
   - ✓ The module is then removed from the client
2. Ansible comes with thousands of modules covering server, network, storage, files, DB etc.
3. Can be written in Python, Perl, Ruby, Bash, etc. – that return JSON format
4. You can write your own modules
5. Command line syntax: *'ansible –m <module_name> -a <attributes>'*
6. They are idempotent (that word again)….

Dictionary definition:

"denoting an element of a set which is unchanged in value when multiplied or otherwise operated on by itself"

"For Ansible it means after 1 run of a playbook to set things to a desired state, further runs of the same playbook should result in 0 changes. Idempotency means you can be sure of a consistent state in your environment."

How Ansible works – Modules (idempotency)

Add a logical volume – first run

```
# ansible lab-aix-1 -m aix_lvol -a "lv=testlv size=10M vg=rootvg"
PLAY [Ansible Ad-Hoc] *******************************************************************
TASK [aix_lvol] *****************************************************************************
changed: [lab-aix-1]

PLAY RECAP
*****************************************************************************************
lab-ibmi-1              : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

During the first run a change occurs. The LV is created.

Add a logical volume – second run

```
# ansible lab-aix-1 -m aix_lvol -a "lv=testlv size=10M vg=rootvg"
PLAY [Ansible Ad-Hoc]*******************************************************************
TASK [aix_lvol] *****************************************************************************
ok: [lab-aix-1]

PLAY RECAP
****************************************************************************************
lab-ibmi-1              : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

During the second run a change does NOT occur. The LV already exists.

How Ansible works – Modules Ad-hoc Execution

crtlib – first run

```
# ansible IBMi_Dev  -m ibmi_cl_command --args="cmd='crtlib ansiblei' asp_group=*SYSBAS"

lab-ibmi-1 | SUCCESS => {
 "changed": false,
   "cmd": "crtlib ansiblei",
   "job_name": "402155/QUSER/QSQSRVR",
"stdout": "{'success': '+++ success crtlib ansiblei'}",
}
```

During the first run, lib created

Conclusion : Module ibmi_cl_command not idempotent

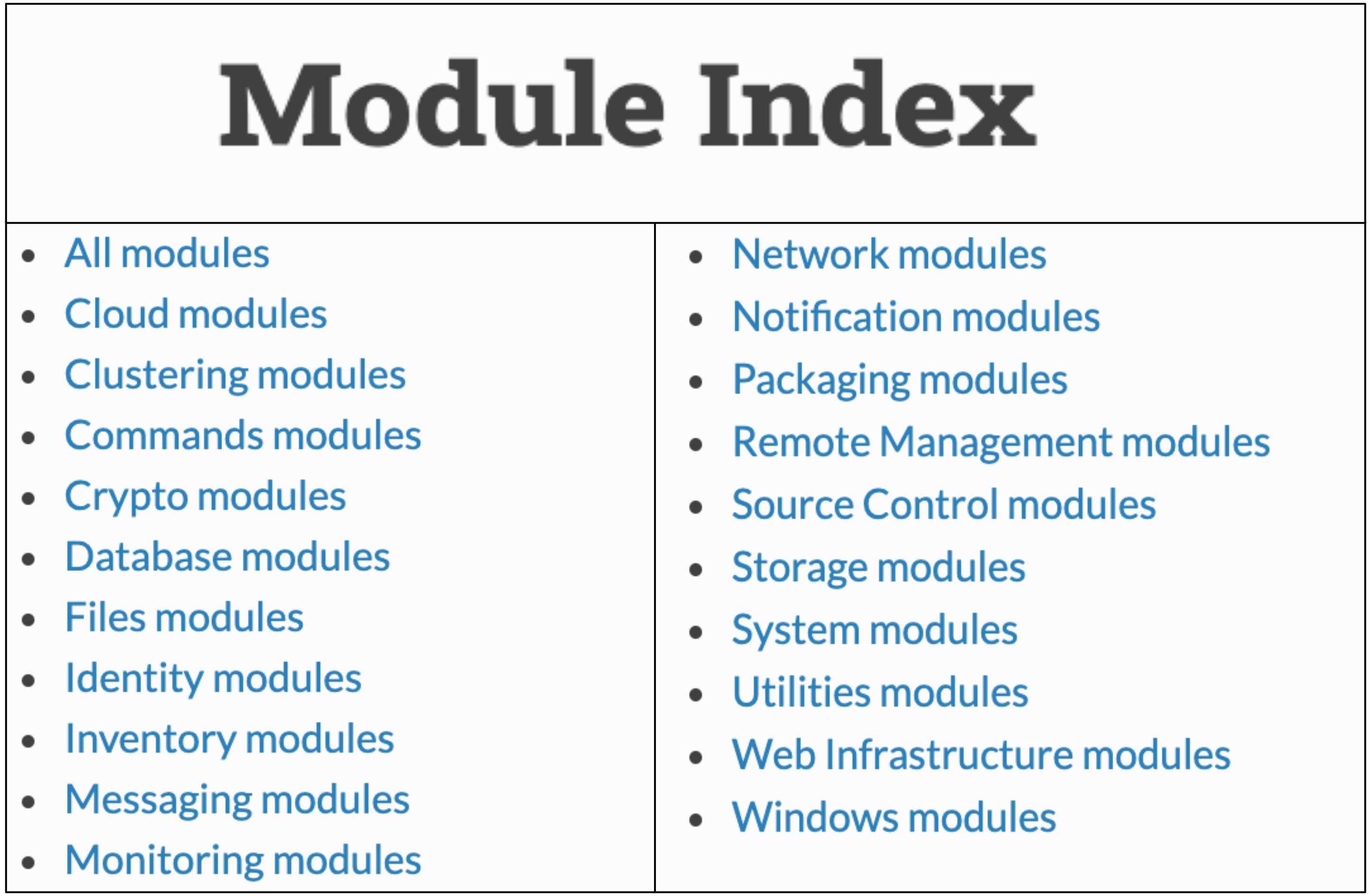
Crtlib – second run

```
# ansible IBMi_Dev  -m ibmi_cl_command --args="cmd='crtlib ansiblei' asp_group=*SYSBAS"

lab-ibmi-1 | FAILED! => {
   "changed": false,
 "job_log": [
       {   "FROM_PROGRAM": "QLICRLIB", "FROM_USER": "BENOIT",  "MESSAGE_ID": "CPF2111",
          "MESSAGE_SECOND_LEVEL_TEXT": "&N Recovery  . . . :   Before creating or renaming this library, change the new library name
or delete the existing library (DLTLIB command). &P -- Use DSPLIB ASPDEV(*ALLAVL) to search for the library.
```

During the second run, lib not created
but need to catch the error

How Ansible works – Modules

Ansible comes with thousands of ‘core’ modules, divided into categories:
https://docs.ansible.com/ansible/latest/modules/modules_by_category.html#modules-by-category



Module Index

- All modules
- Cloud modules
- Clustering modules
- Commands modules
- Crypto modules
- Database modules
- Files modules
- Identity modules
- Inventory modules
- Messaging modules
- Monitoring modules
- Network modules
- Notification modules
- Packaging modules
- Remote Management modules
- Source Control modules
- Storage modules
- System modules
- Utilities modules
- Web Infrastructure modules
- Windows modules

How Ansible works – Modules

As well as Anisble’s website we can also use the Ansible Engine server to show modules, how they are supported, options available etc.

Using ‘ansible-doc’ to review a module

```
# ansible-doc ibmi_cl_command
>IBMI_CL_COMMAND    (/Users/Benoit2/.ansible/collections/ansible_collections/ibm/power_ibmi/plugins/modules/ibmi_cl_command.py)


The `ibmi_cl_command' module takes the CL command followed by a list of space-delimited arguments.
For PASE(Portable Application Solutions Environment for i) or QSHELL(Unix/Linux-liked) commands,
like 'ls', 'chmod', use the `command' module instead.

- become_user
      The name of the user profile that the IBM i task will run under.
      Use this option to set a user with desired privileges to run the task.
      [Default: (null)]
      type: str

= cmd
      The CL command to run.
      type: str

- joblog
      If set to `true', output the available job log even the rc is 0(success).
      ……
```

Shows the location of the module and support level.

The “=“ indicates mandatory parameters.

How Ansible works – Modules

What happens if we call an invalid module?

Calling an IBM i module

```
# ansible lab-ibmi-1   -m ibmi_lib_save --args="lib_name=ansiblei format=*SAVF"

lab-ibmi-1 | FAILED! => {
   "changed": false,
   "msg": "missing required arguments: savefile_lib, savefile_name"
}

ansible lab-ibmi-1   -m ibmi_lib_save --args="lib_name=ansiblei format=*SAVF savefile_lib=QGPL savefile_name=ansiblei"

lab-ibmi-1 | SUCCESS => {
   "changed": false,
   "command": "QSYS/SAVLIB LIB(ansiblei) DEV(*SAVF) SAVF(QGPL/ansiblei) TGTRLS(*CURRENT)",  "format": "*SAVF",
   "stdout": "{'success': '+++ success QSYS/SAVLIB LIB(ansiblei) DEV(*SAVF) SAVF(QGPL/ansiblei) TGTRLS(*CURRENT)'}",
 ……
}
```

Module with missing parameters

How Ansible works – Modules

✓ **script** module – Runs a local script on a remote node after transferring it

Simple ‘script’ module example

```
# cat ./show_date.sh
#!/bin/sh
date

# ansible lab-ibmi-1 -m script -a "./date.sh"
lab-ibmi-1 | CHANGED => {
    "changed": true,
    "rc": 0,
    "stderr": "Shared connection to lab-ibmi-1 closed.\r\n",
    "stderr_lines": [
        "Shared connection to lab-ibmi-1 closed."
    ],
    "stdout": "Wed Sep 21 16:40:47 CEST 2022\r\n",
    "stdout_lines": [
        "Wed Sep 21 16:40:47 CEST 2022"
    ]
}
```

Script on the Ansible Engine

Script is copied over and executed on the client

How Ansible works – Modules (setup and facts)

✓ **setup** module – Gathers facts about remote hosts (~100 lines for a IBM i LPAR)

Setup module

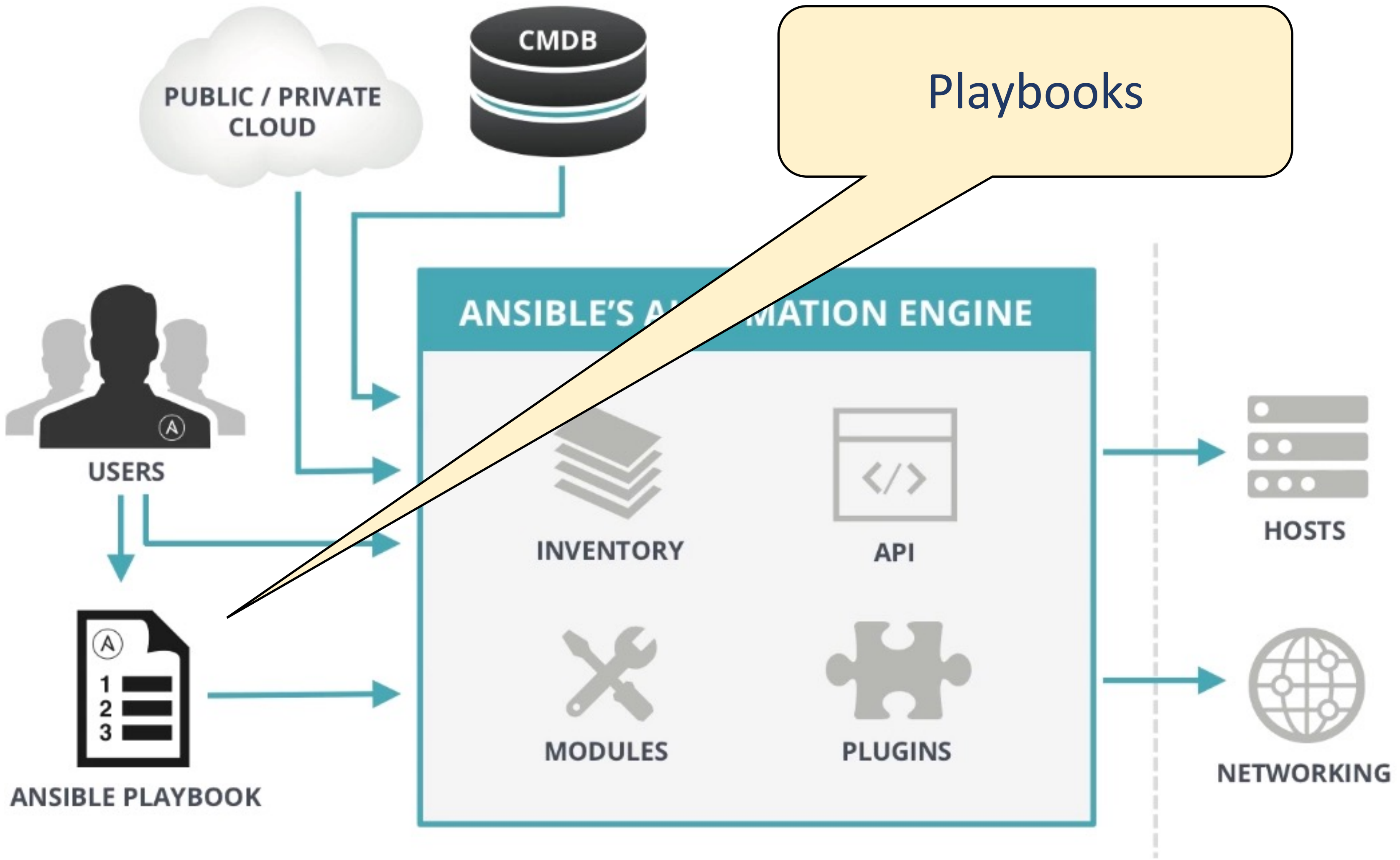
```
# ansible lab-ibmi-1 -m setup
lab-ibmi-1 | SUCCESS => {
    "ansible_facts": {
        "ansible_distribution": "OS400",
        "ansible_distribution_release": "3",
        "ansible_distribution_version": "7",
        "ansible_dns": {},
        "ansible_domain": "dcry.iccmop",
        "ansible_effective_group_id": 0,
        "ansible_effective_user_id": 150,
        "ansible_env": {
            "HOME": "/home/BENOIT",
            "LOGIN": "benoit",
            "USER": "benoit",
            "_": "/QOpensys/pkgs/bin/python3"
        },
        "ansible_machine": "00100002BABV",
        "ansible_nodename": "BENOIT",
        "ansible_os_family": "OS400",
.....
```

Thousands of facts about h/w, OS, network and storage devices etc. can be gathered.

These can be used to filter which clients to run a task against in a playbook.

How Ansible works

1. Ansible Engine
2. Inventory
3. Modules
4. Playbooks
5. Client hosts



Playbooks

How Ansible works – Playbooks

Modules might be the core, but Playbooks are how we drive Ansible

- ✓ Playbooks are Ansible’s configuration, deployment, and orchestration language.
- ✓ They are the instruction manual describing the configuration you want your remote clients to enforce.
- ✓ Written in YAML format, so should be readable.

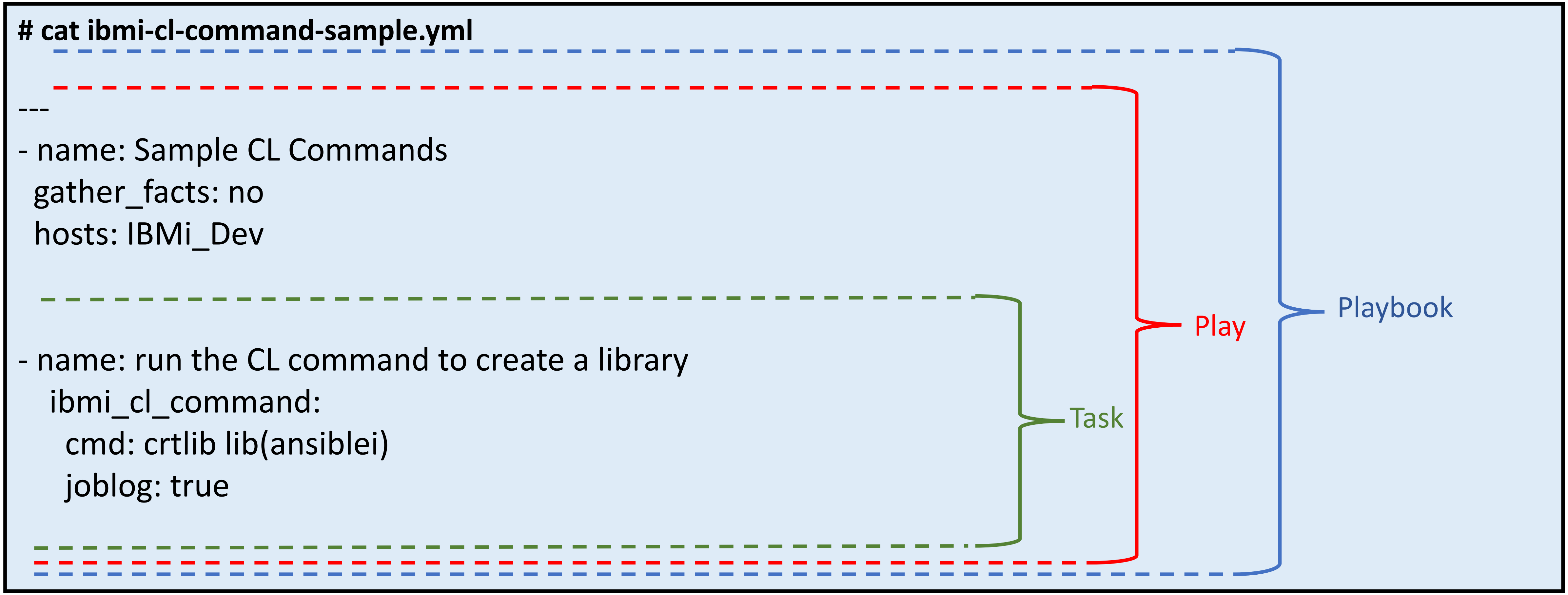
Basic playbooks:
Used to manage configurations of and deployments to remote machines.

Advanced playbooks:
They can sequence multi-tier rollouts involving rolling updates, and can delegate actions to other hosts, interacting with monitoring servers and load balancers along the way.

How Ansible works – Playbooks

A playbook consists of ‘plays’, which in turn consist of ’tasks’, which contain ’modules’.

Simple playbook

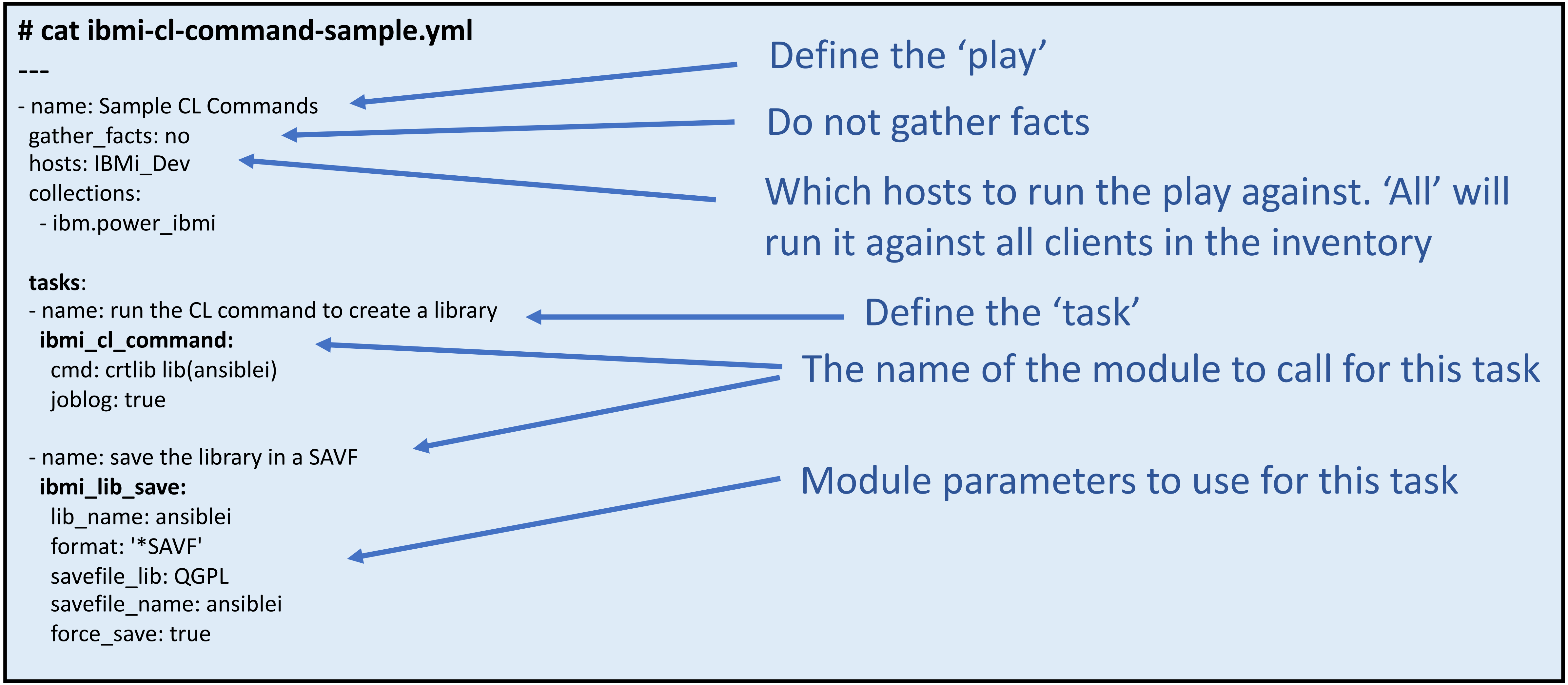


```
# cat ibmi-cl-command-sample.yml
---
- name: Sample CL Commands
  gather_facts: no
  hosts: IBMi_Dev

- name: run the CL command to create a library
    ibmi_cl_command:
      cmd: crtlib lib(ansiblei)
      joblog: true
```

How Ansible works – Playbooks

A playbook consists of ‘plays’, which in turn consist of ’tasks’, which contain ’modules’.

Simple playbook



```
# cat ibmi-cl-command-sample.yml
---
- name: Sample CL Commands
  gather_facts: no
  hosts: IBMi_Dev
  collections:
   - ibm.power_ibmi

  tasks:
  - name: run the CL command to create a library
    ibmi_cl_command:
      cmd: crtlib lib(ansiblei)
      joblog: true

  - name: save the library in a SAVF
    ibmi_lib_save:
      lib_name: ansiblei
      format: '*SAVF'
      savefile_lib: QGPL
      savefile_name: ansiblei
      force_save: true
```

Define the ‘play’

Do not gather facts

Which hosts to run the play against. ‘All’ will run it against all clients in the inventory

Define the ‘task’

The name of the module to call for this task

Module parameters to use for this task

How Ansible works – Playbooks

A playbook consists of ‘plays’, which in turn consist of ’tasks’, which contain ’modules’.

Simple playbook

```
# ansible-playbook  ibmi-cl-command-sample2.yml

PLAY [Sample CL Commands]
********************************************************************************

TASK [run the CL command to create a library]
********************************************************************************
ok: [lab-ibmi-1]
ok: [lab-ibmi-2]

TASK [save the library in a SAVF]
********************************************************************************
ok: [lab-ibmi-1]
ok: [lab-ibmi-2]

PLAY RECAP ********************************************************************************
lab-ibmi-1            : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
lab-ibmi-2            : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

The name of the ‘play’

The name of the ‘task’

Completed on 2 clients

How Ansible works – Playbooks (tasks and tags)

We can list the tasks in a playbook without actually running it:

Task in a playbook

```
# ansible-playbook ./ibmi-cl-command-sample2.yml --list-tasks

playbook: ibmi-cl-command-sample2.yml


  play #1 (IBMi_Dev): Sample CL Commands TAGS: []
    tasks:
      run the CL command to create a library TAGS: []
      save the library in a SAVF TAGS: []
```

All the tasks are listed but not executed

How Ansible works – Playbooks (tasks and tags)

We can also ‘tag’ tasks with identifiers :

Task and tags in a playbook

```
# cat  ./ibmi-cl-command-sample2.yml
…
 tasks:
 - name: run the CL command to create a library
   ibmi_cl_command:
     cmd: crtlib lib(ansiblei)
     joblog: true
   tags: crtlib
 - name: save the library in a SAVF
   ibmi_lib_save:
     lib_name: ansiblei
     format: '*SAVF'
     savefile_lib: QGPL
     savefile_name: ansiblei
     force_save: true
   tags: savefile

….
```

We can add ‘tag’ names to each task.

How Ansible works – Playbooks (tasks and tags)

We can also ‘tag’ tasks with identifiers, and list them:

Task and tags in a playbook

```
# ansible-playbook ./ibmi-cl-command-sample2.yml --list-tasks

playbook: ibmi-cl-command-sample2.yml


  play #1 (IBMi_Dev): Sample CL Commands TAGS: []
    tasks:
      run the CL command to create a library TAGS: [crtlib]
      save the library in a SAVF TAGS: [savefile]
```

How Ansible works – Playbooks (tasks and tags)

We can then just run certain tasks, by giving a tag:

List savefile tasks only

```
# ansible-playbook  ibmi-cl-command-sample2.yml --list-tasks -t savefile
playbook: ./ibmi-cl-command-sample2.yml
  play #1 (IBMi_Dev): Sample CL Commands TAGS: []
    tasks:
      save the library in a SAVF TAGS: [savefile]
```

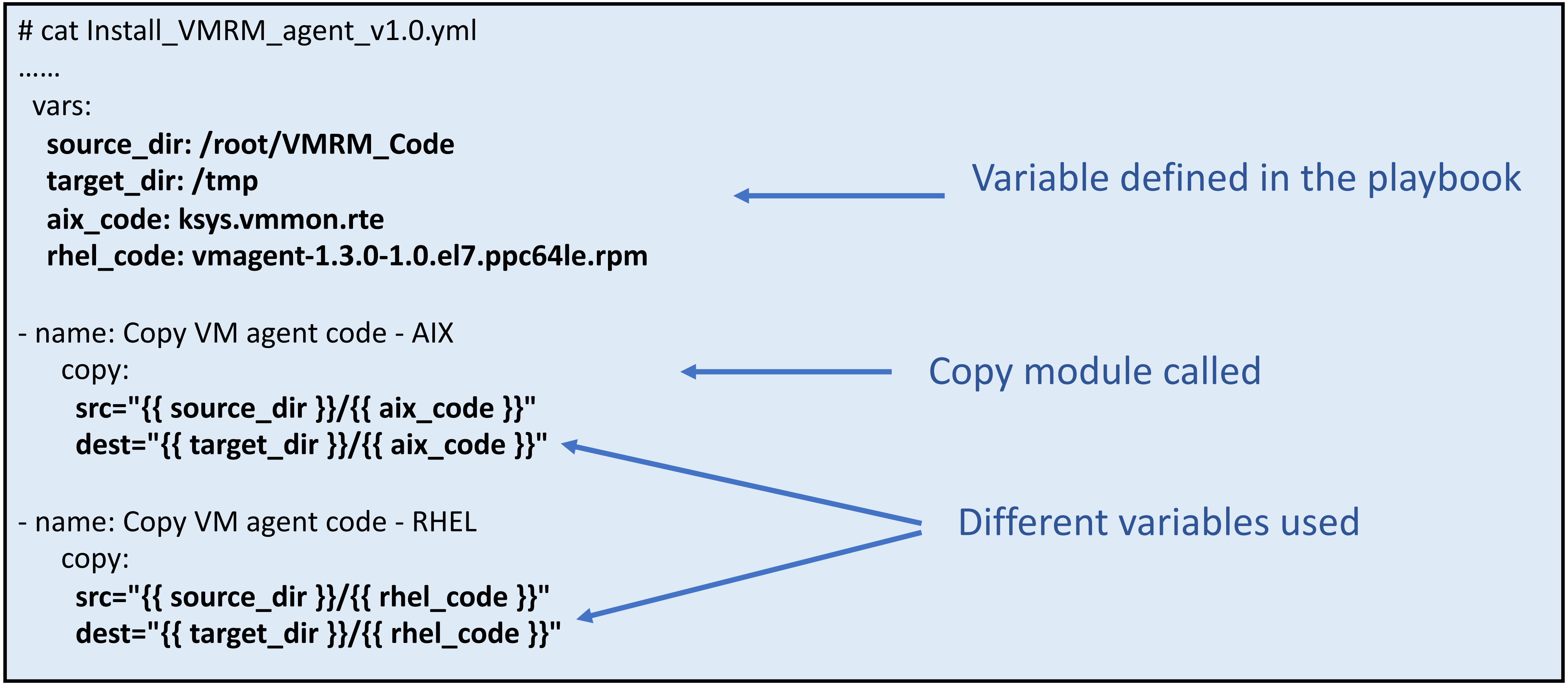
Run ‘savefile’ tasks only

```
# ansible-playbook  ibmi-cl-command-sample2.yml  -t savefile
PLAY [Sample CL Commands]
***************************************************************************
TASK [save the library in a SAVF]
***************************************************************************
ok: [lab-ibmi-1]
ok: [lab-ibmi-2]

PLAY RECAP ***************************************************************************
lab-ibmi-1              : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
lab-ibmi-2              : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

How Ansible works – Playbooks (variables)

We can define variables from within the playbook

Playbook variables example



```
# cat Install_VMRM_agent_v1.0.yml
……
  vars:
    source_dir: /root/VMRM_Code
    target_dir: /tmp
    aix_code: ksys.vmmon.rte
    rhel_code: vmagent-1.3.0-1.0.el7.ppc64le.rpm

- name: Copy VM agent code - AIX
      copy:
        src="{{ source_dir }}/{{ aix_code }}"
        dest="{{ target_dir }}/{{ aix_code }}"

- name: Copy VM agent code - RHEL
      copy:
        src="{{ source_dir }}/{{ rhel_code }}"
        dest="{{ target_dir }}/{{ rhel_code }}"
```

Variable defined in the playbook

Copy module called

Different variables used

How Ansible works – Playbooks (variables)

We can ‘include’ variables from an external file. There is a ‘priority’ order of var definition

Imported variables example

```
# cat OSlevel_check.yml
---
- hosts: all
  tasks:
    - name: Load IBMi specific variables
      include_vars: IBMi.yml

    - name: Check OS
      command: "{{ os_check_command }}”
```

We include an external variables file

The command modules needs a variable called ‘os_check_command’

```
# cat IBMi.yml
---
# variables for script
os_check_command:    "oslevel -s"
args_variable_name:  ”IBMi_OS"
```

The ‘os_check_command’ is defined in this variable file and passed back to the main playbook.

How Ansible works – Playbooks (conditions)

We can run tasks against ‘facts’ gathered from the clients, for example OS type

Playbook ‘when’ example

```
# cat OSlevel_check.yml
---
- hosts: Dev
  tasks:
    - name: Load AIX specific variables
      include_vars: AIX.yml
      when: ansible_distribution == "AIX"

    - name: Load RHEL specific variables
      include_vars: RHEL.yml
      when: ansible_distribution == "RedHat”

    - name: Load IBM i specific variables
      include_vars: IBMi.yml
      when: ansible_distribution == ”OS400"

    - name: Check OS
      command: "{{ os_check_command }}"
      register: os_check_result
      args:
        creates:  "{{ args_variable_name }}"
```

Include a different variable file depending on the clients OS type

The relevant OS command is passed back

How Ansible works – Playbooks (Roles)

As we start out with Ansible we tend to create one or two large playbooks

Although this is a good start we may want to reuse file and avoid repeating code.

Roles, import and includes are a good way to do this.

Roles allow us to automatically load certain variables, tasks and handlers based on a know file structure. These can then be shared amongst other uses and projects.

How Ansible works – Playbooks (Roles)

Creating a role:

```
# ansible-galaxy init db-server-role
- Role db-server-role was created successfully
```

Directory structure of a role:

```
# tree
.
└── db-server-role
    ├── defaults
    │   └── main.yml
    ├── files
    ├── handlers
    │   └── main.yml
    ├── meta
    │   └── main.yml
    ├── README.md
    ├── tasks
    │   └── main.yml
    ├── templates
    ├── tests
    │   ├── inventory
    │   └── test.yml
    └── vars
        └── main.yml
```

If main.yml playbooks exist within the role, the tasks, handlers, variable etc. listed within will be added to the play that called it.

How Ansible works – Playbooks (Roles)

Why do we need roles?? If we look at our OpenStack playbook that creates AIX, Linux or IBMi VMs, its complex:

```
# ansible-playbook playbooks/VM_build.yml --list-tasks
play #1 (localhost): Build new VM via PowerVC/OpenStack      TAGS: []
   tasks:
     Prompt for new VM Name    TAGS:                                        [VM_Create]
     Set VM Variables  TAGS:                                                [VM_Create]
     Display VM Name   TAGS:                                                [VM_Create]
     VM_network_list : Retrieve list of all networks   TAGS:                [VM_Create, VM_Network]
     VM_network_list : Generate Network list   TAGS:                        [VM_Create, VM_Network]
     VM_network_list : Debug - Output Network list     TAGS:                [VM_Create, VM_Network]
     VM_network_list : Display Network list    TAGS:                        [VM_Create, VM_Network]
     ……
     VM_image_list : Retrieve list of all OS Distributions     TAGS:        [VM_Create, VM_Images]
     VM_image_list : Filter OS Distribution list       TAGS:                [VM_Create, VM_Images]
     …..
     VM_flavor_list : Retrieve list of all public flavors      TAGS:        [VM_Flavor, always, never]
     ….
     VM_name_list : Retrieve list of all VMs   TAGS:                        [VM_Create, VM_List]
     VM_name_list : Retrieve VM list   TAGS:                                [VM_Create, VM_List]
     ….
     VM_create_vm : Create a new VM instance   TAGS:                        [VM_Create]
     VM_create_vm : Print VM's public IP address       TAGS:                [VM_Create]
```

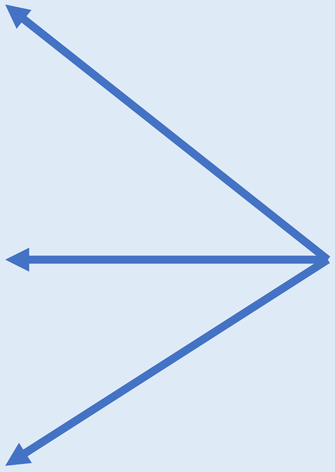
Each group of tasks is in its own role

65 tasks in total

How Ansible works – Playbooks (Roles)

These roles can be used multiple times from other playbooks, other users or other projects:

```
# cat playbooks/VM_build.yml
---
- name: Build new VM via PowerVC/OpenStack

  tasks:
  - name: List Available Networks
    import_role:
       name: VM_network_list
    tags: VM_Create, VM_Network

  - name: Pick Network for VM
    import_role:
      name: VM_network_pick
    tags: VM_Create

  - name: List VM images
    import_role:
      name: VM_image_list
     tags: VM_Create, VM_Images
......
```



Within the tasks we import each role

How Ansible works – Other features

Handlers

Handlers are lists of tasks, that are referenced by a globally unique name, and are notified by notifiers. If nothing notifies a handler, it will not run. Regardless of how many tasks notify a handler, it will run only once, after all of the tasks complete in a particular play.

Blocks

Blocks allow for logical grouping of tasks and in play error handling. Most of what you can apply to a single task can be applied at the block level, which also makes it much easier to set data or directives common to the tasks.

Vaults

Ansible Vault is a feature of ansible that allows you to keep sensitive data such as passwords or keys in encrypted files, rather than as plaintext in playbooks or roles. These vault files can then be distributed or placed in source control.

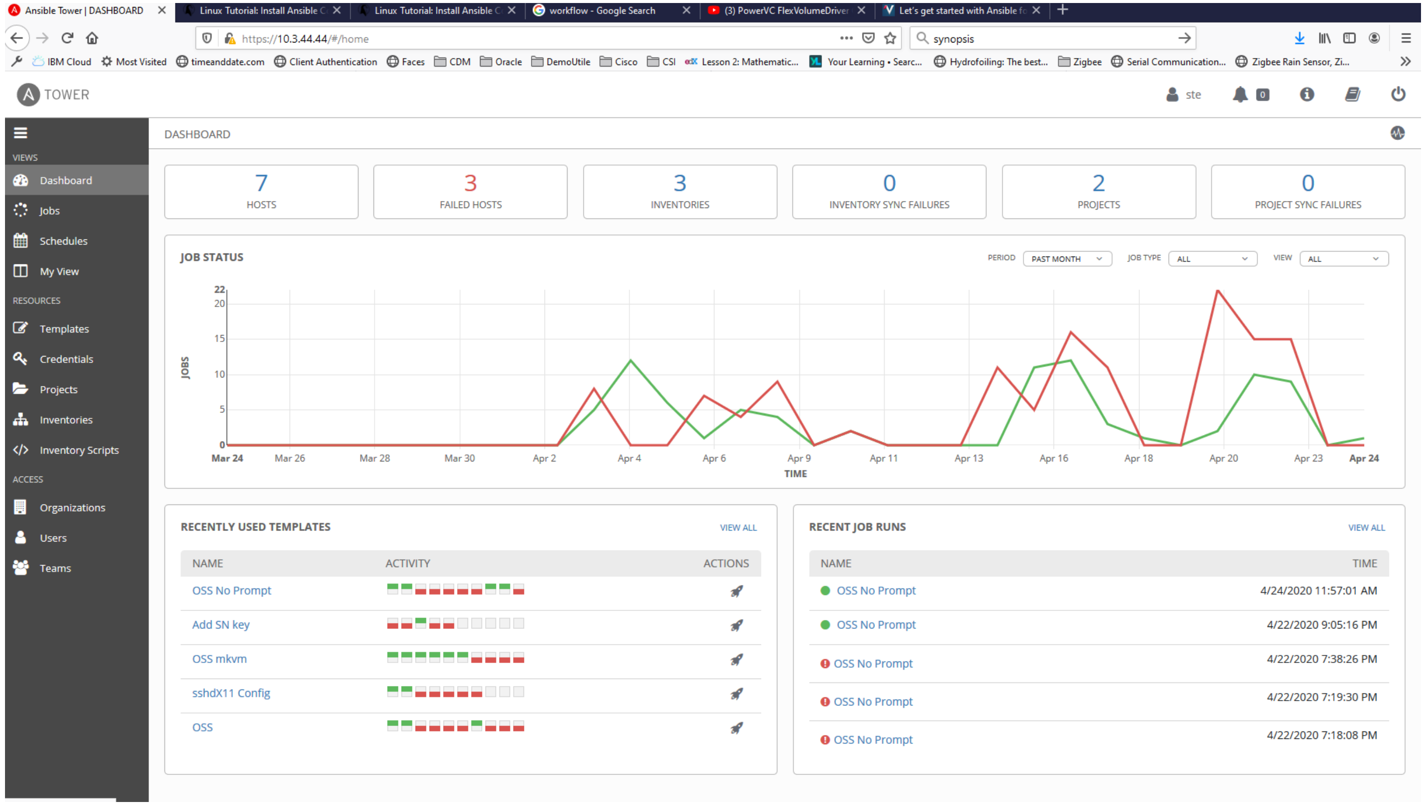
Galaxy

Ansible Galaxy refers to the Galaxy website, a free site for finding, downloading, and sharing community developed roles.

https://galaxy.ansible.com/home

You don’t like putty and ssh screen ?

- Ansible tower helps to launch ansible playbook using a GUI



7	3	3	0	2	0
HOSTS	FAILED HOSTS	INVENTORIES	INVENTORY SYNC FAILURES	PROJECTS	PROJECT SYNC FAILURES

JOB STATUS

PERIOD: PAST MONTH   JOB TYPE: ALL   VIEW: ALL

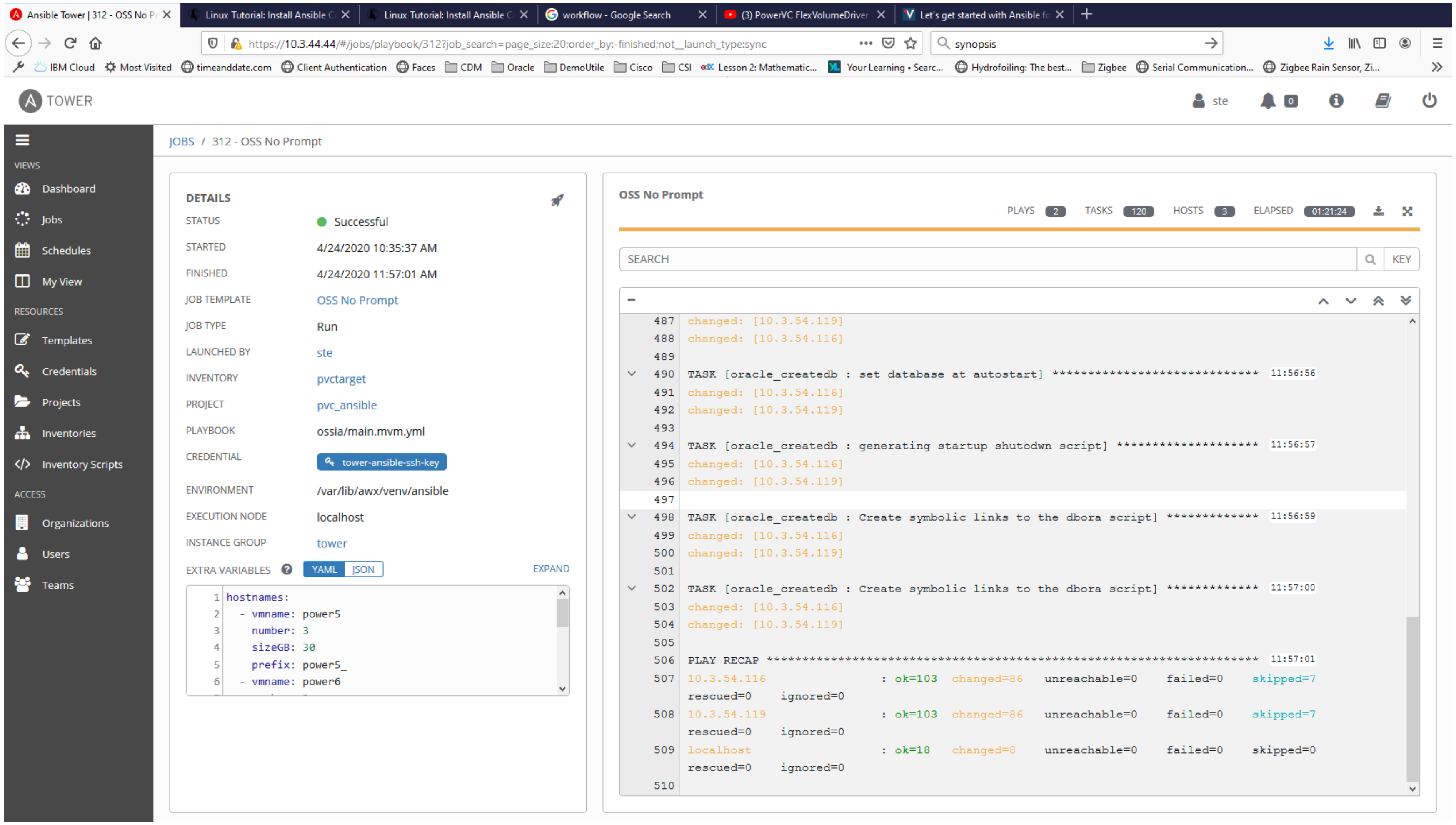
RECENTLY USED TEMPLATES

NAME	ACTIVITY	ACTIONS
OSS No Prompt	🟩🟩🟥🟥🟥🟥🟥🟩🟩🟥	🚀
Add SN key	🟥🟥🟩🟥🟥□□□□□	🚀
OSS mkvm	🟩🟩🟩🟩🟩🟩🟥🟥🟥🟥	🚀
sshdX11 Config	🟩🟩🟥🟥🟥🟥🟥□□□	🚀
OSS	🟩🟩🟥🟥🟥🟥🟩🟥🟥🟥	🚀

RECENT JOB RUNS

NAME	TIME
● OSS No Prompt	4/24/2020 11:57:01 AM
● OSS No Prompt	4/22/2020 9:05:16 PM
! OSS No Prompt	4/22/2020 7:38:26 PM
! OSS No Prompt	4/22/2020 7:19:30 PM
! OSS No Prompt	4/22/2020 7:18:08 PM

But still running ansible playbooks !



JOBS / 312 - OSS No Prompt

DETAILS

STATUS ● Successful

STARTED 4/24/2020 10:35:37 AM

FINISHED 4/24/2020 11:57:01 AM

JOB TEMPLATE OSS No Prompt

JOB TYPE Run

LAUNCHED BY ste

INVENTORY pvctarget

PROJECT pvc_ansible

PLAYBOOK ossia/main.mvm.yml

CREDENTIAL tower-ansible-ssh-key

ENVIRONMENT /var/lib/awx/venv/ansible

EXECUTION NODE localhost

INSTANCE GROUP tower

EXTRA VARIABLES YAML JSON EXPAND

```
hostnames:
  - vmname: power5
    number: 3
    sizeGB: 30
    prefix: power5_
  - vmname: power6
```

OSS No Prompt

PLAYS 2 TASKS 120 HOSTS 3 ELAPSED 01:21:24

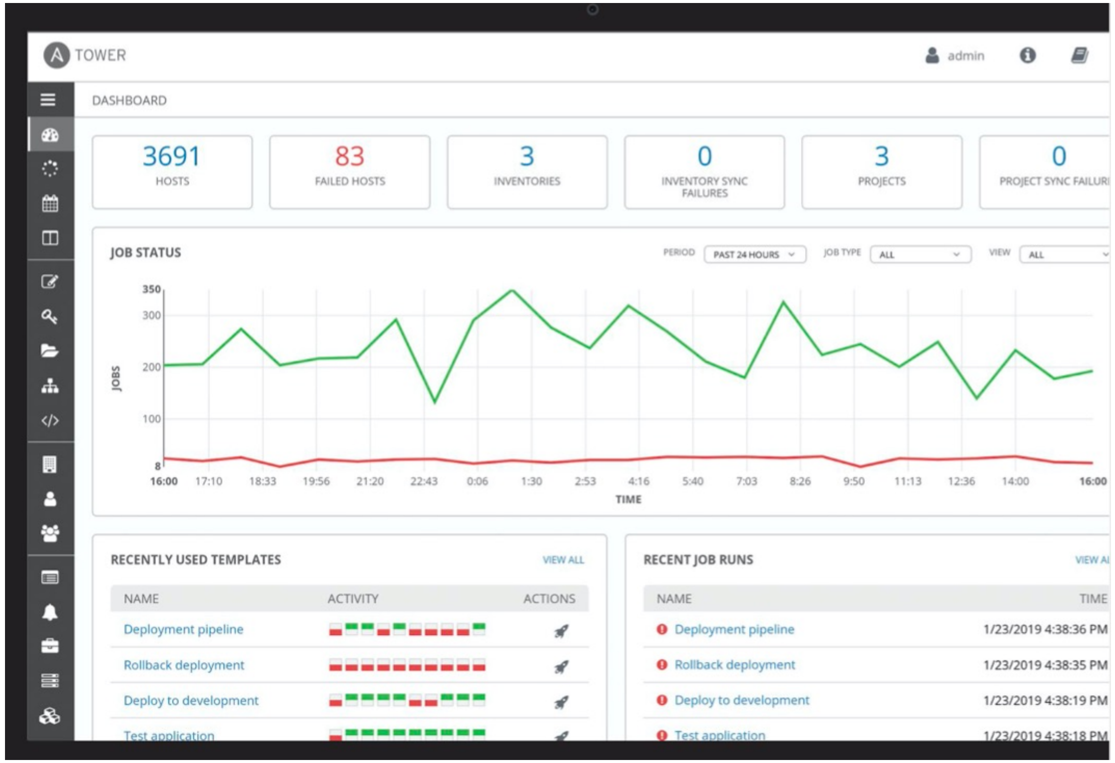
SEARCH KEY

```
changed: [10.3.54.119]
changed: [10.3.54.116]

TASK [oracle_createdb : set database at autostart] ****************************** 11:56:56
changed: [10.3.54.116]
changed: [10.3.54.119]

TASK [oracle_createdb : generating startup shutodwn script] ********************* 11:56:57
changed: [10.3.54.116]
changed: [10.3.54.119]

TASK [oracle_createdb : Create symbolic links to the dbora script] ************** 11:56:59
changed: [10.3.54.116]
changed: [10.3.54.119]

TASK [oracle_createdb : Create symbolic links to the dbora script] ************** 11:57:00
changed: [10.3.54.116]
changed: [10.3.54.119]

PLAY RECAP ********************************************************************** 11:57:01
10.3.54.116                : ok=103  changed=86   unreachable=0    failed=0    skipped=7
rescued=0    ignored=0
10.3.54.119                : ok=103  changed=86   unreachable=0    failed=0    skipped=7
rescued=0    ignored=0
localhost                  : ok=18   changed=8    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0
```

Ansible Tower

Ansible Tower is a UI and RESTful API allowing you to scale IT automation, manage complex deployments and speed productivity.

- Role-based access control
- Deploy entire applications with push-button deployment access
- All automations are centrally logged
- Powerful workflows match your IT processes



3691 HOSTS

83 FAILED HOSTS

3 INVENTORIES

0 INVENTORY SYNC FAILURES

3 PROJECTS

0 PROJECT SYNC FAILUR

PERIOD: PAST 24 HOURS

JOB TYPE: ALL

VIEW: ALL

RECENTLY USED TEMPLATES

NAME	ACTIVITY	ACTIONS
Deployment pipeline	🟥🟩🟩🟥🟩🟥🟥🟥🟥🟩	🚀
Rollback deployment	🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥	🚀
Deploy to development	🟥🟩🟩🟩🟩🟥🟥🟩🟩🟩	🚀
Test application	🟥🟩🟩🟩🟩🟩🟩🟩🟩🟩	🚀

RECENT JOB RUNS

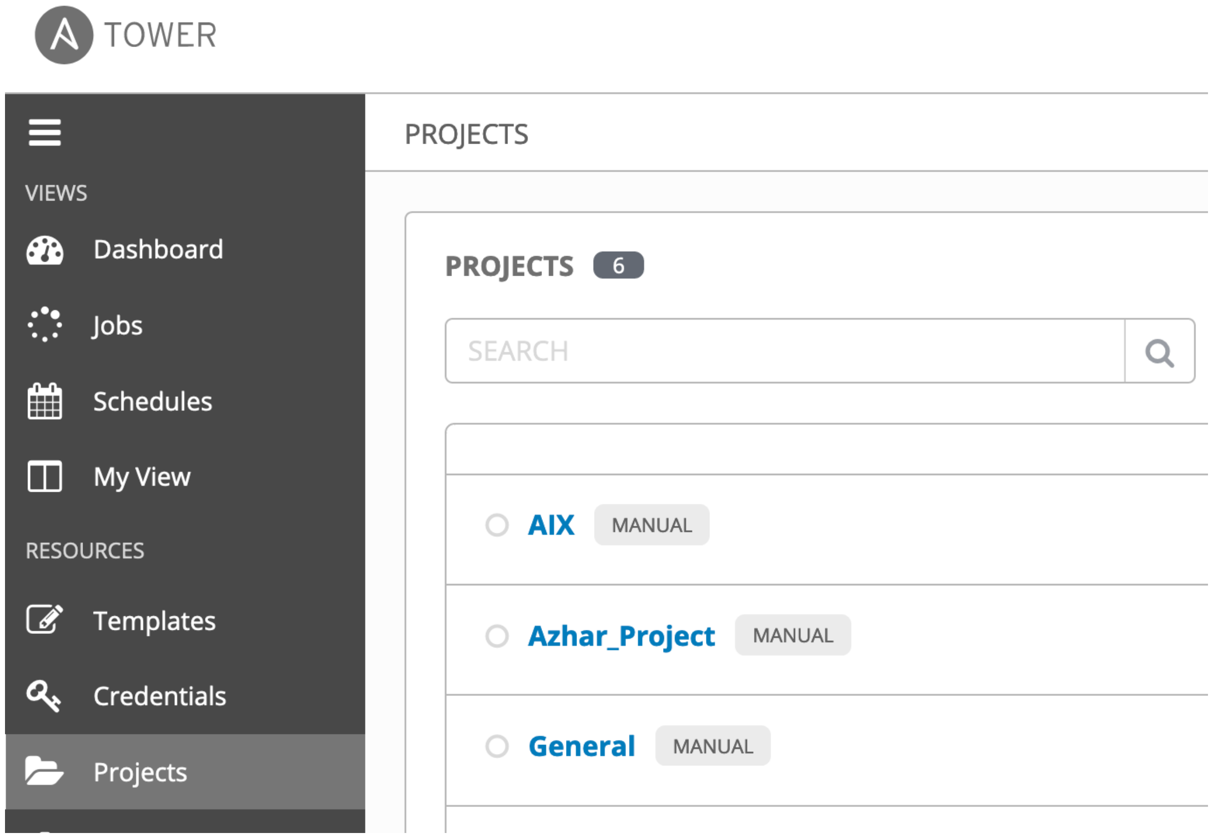
NAME	TIME
Deployment pipeline	1/23/2019 4:38:36 PM
Rollback deployment	1/23/2019 4:38:35 PM
Deploy to development	1/23/2019 4:38:19 PM
Test application	1/23/2019 4:38:18 PM

Ansible Tower - Projects

Project

A project is a logical collection of Ansible Playbooks, represented in Ansible Tower.

You can manage Ansible Playbooks and playbook directories by placing them in a source code management system supported by Ansible Tower, including Git, Subversion, and Mercurial.



○ AIX MANUAL

○ Azhar_Project MANUAL

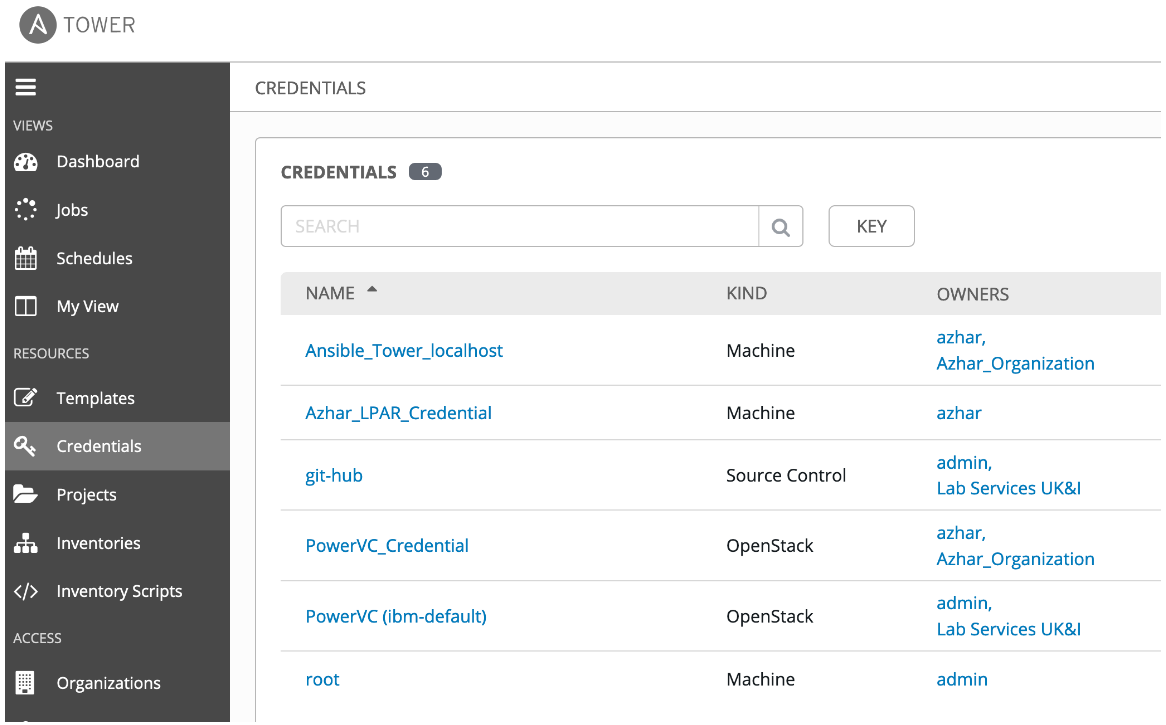
○ General MANUAL

Ansible Tower - Credentials

Credentials

Credentials are utilized by Ansible Tower for authentication with various external resources:

- Connecting to remote machines to run jobs
- Syncing with inventory sources
- Importing project content from version control systems
- Connecting to and managing devices



CREDENTIALS 6

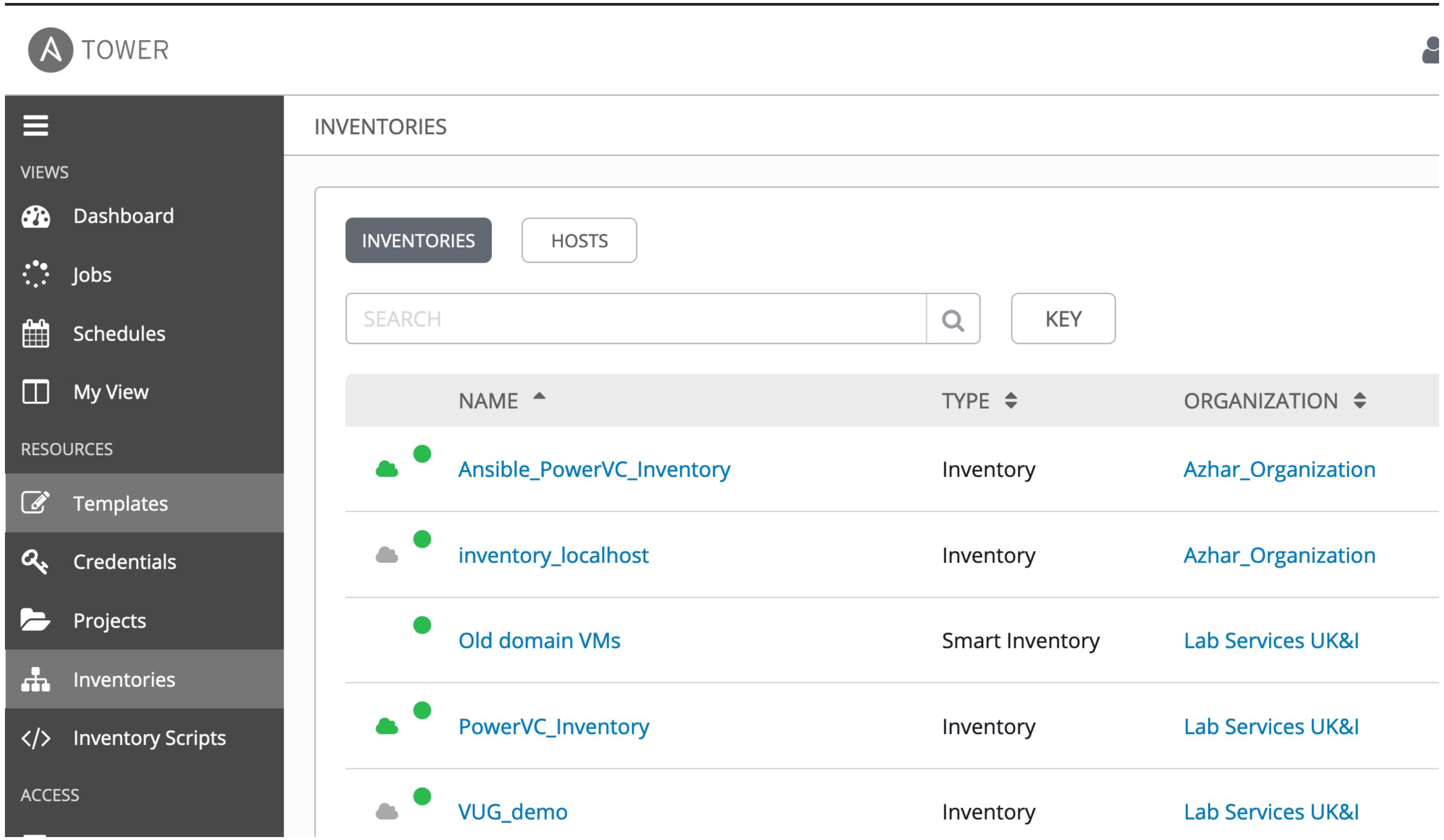
NAME	KIND	OWNERS
Ansible_Tower_localhost	Machine	azhar, Azhar_Organization
Azhar_LPAR_Credential	Machine	azhar
git-hub	Source Control	admin, Lab Services UK&I
PowerVC_Credential	OpenStack	azhar, Azhar_Organization
PowerVC (ibm-default)	OpenStack	admin, Lab Services UK&I
root	Machine	admin

Ansible Tower - Inventory

Inventory

Inventory is a collection of hosts clients (just like the with the engine) with associated data and groupings that Ansible Tower can connect to and manage.

- Hosts (nodes)
- Groups
- Inventory-specific data (variables)
- Static or dynamic sources



NAME	TYPE	ORGANIZATION
Ansible_PowerVC_Inventory	Inventory	Azhar_Organization
inventory_localhost	Inventory	Azhar_Organization
Old domain VMs	Smart Inventory	Lab Services UK&I
PowerVC_Inventory	Inventory	Lab Services UK&I
VUG_demo	Inventory	Lab Services UK&I

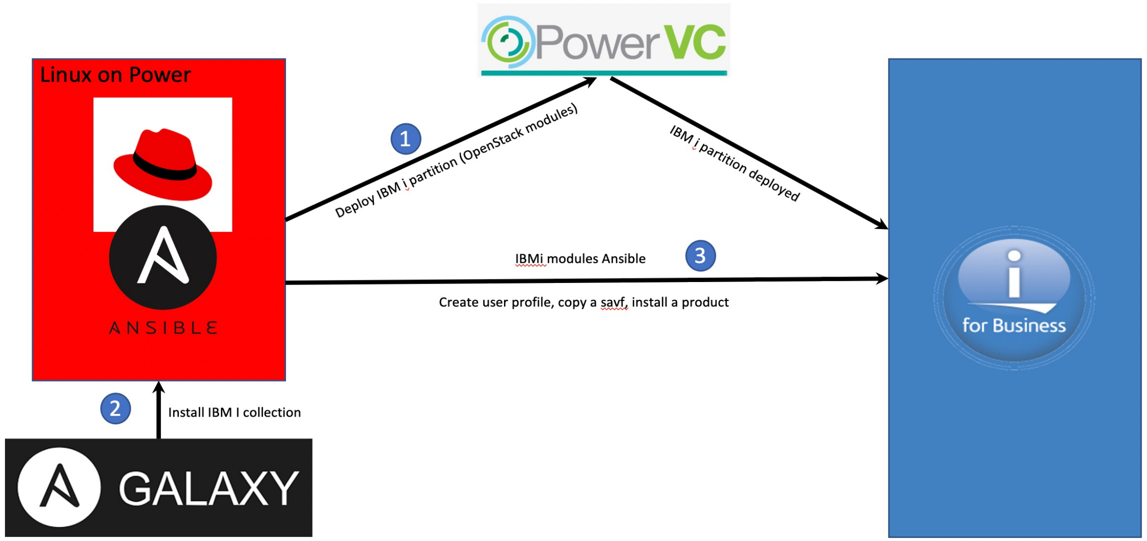
ANSIBLE

Ansible for i Labs

LAB 1

Ansible for i 101

https://ibm.box.com/v/ansible-for-i-lab1

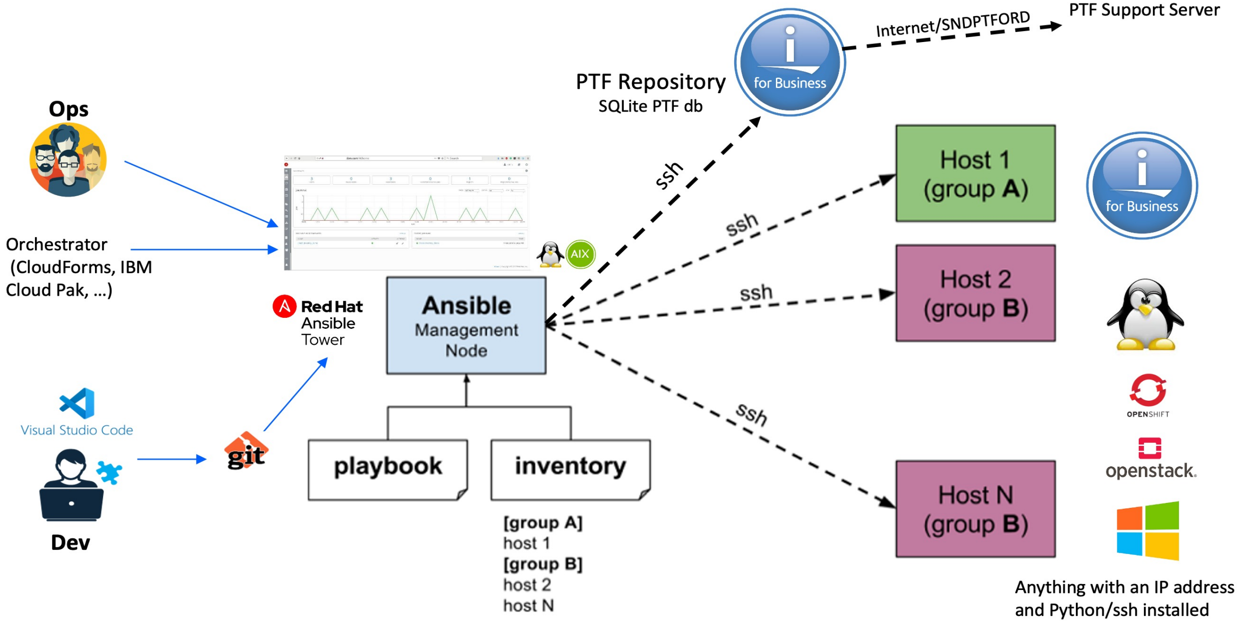


LAB 2

PTF Management advanced

Ansible AWX / Redhat Ansible Tower

https://ibm.box.com/v/ansible-for-i-lab2



Ansible for i - Example

Q: How do I automate a backup on multiple systems with a single tape drive?

Answer:
ansible-playbook playbooks/**ibmi-savelib.yml**

```
[myibmisystems]
10.7.19.71 ansible_ssh_user=benoit
10.7.19.72 ansible_ssh_user=benoit
10.7.19.73 ansible_ssh_user=benoit[ibmi:vars]
ansible_python_interpreter="/QOpensys/pkgs/bin/python3"
ansible_ssh_common_args='-o StrictHostKeyChecking=no'
```

ibmi-savelib.yml

```
---
## Sequential save on all IBM i systems in the myibmisystems group
## serial :1 for sequential execution (single tape drive)
- hosts: myibmisystems
 serial: 1
 collections:
  - ibm.power_ibmi
 tasks:
  - name: Vary on TAPE
   ibmi_device_vary:
    device_list: ['TAPVRT01']
    status: '*ON'
  - name: LODIMGCLG
   ibm.power_ibmi.ibmi_cl_command:
    cmd: 'LODIMGCLG IMGCLG(VIRTUALTAP) DEV(TAPVRT01)'
    become_user: '<userprofile>'
    become_user_password: '<userprofilepwd>'
  - name: SAVLIB
   ibm.power_ibmi.ibmi_cl_command:
    cmd: 'SAVLIB LIB(TOTO) ACCPTH(*YES) DEV(TAPVRT01)'
    become_user: '<userprofile>'
    become_user_password: '<userprofilepwd>'
  - name: Vary off TAPE
   ibm.power_ibmi.ibmi_device_vary:
    device_list: ['TAPVRT01']
    status: '*OFF'
```

Ansible for i Demo

https://github.com/bmarolleau/ansible-for-i

demo0-list-inventory.sh
demo1-ptfgroup-check.sh
demo2-disable-usrprf-CL.sh
demo3-fix-imgclg.sh
demo4-sync-apply-ptfgrp.sh