Automation with Ansible

utian@btech.id

References

– – –

- Ansible Documentation: http://docs.ansible.com
- Red Hat Training Ansible Essentials: https://www.redhat.com/en/services/training/do007-ansible-essentials-simplicity-automation-technical-overview
- Mastering Ansible, Jesse Keating, 2015 Pack Publishing

Keywords

– – –

- Ansible
- Ansible Galaxy
- Ansible Tower
- Ansible Vault
- Jinja2
- Tasks
- Modules
- Ad-Hoc Commands
- Playbooks
- Roles
- Inventories

Automation with Ansible

Ansible

What is Ansible?

– – –

- An automation language that can describe an IT application infrastructure in Ansible Playbooks.
- An automation engine that runs Ansible Playbooks.
- Ansible Tower is an enterprise framework for controlling, securing and managing your Ansible automation with a UI and RESTful API.

Ansible is Simple

– – –

- Human readable automation
- No special coding skills needed
- Tasks executed in order
- Get productive quickly

Ansible is Powerful

– – –

- Application deployment
- Configuration management
- Workflow orchestration
- Orchestrate the application lifecycle

Ansible is Agentless

– – –

- Agentless architecture
- Uses OpenSSH & WinRM
- No agents to exploit or update
- More efficient & more secure

Ansible is Cross Platform

– – –

Agentless support for all major OS variants, physical, virtual, cloud and network.

Ansible Works With Existing Toolkits

Homogenize existing environments by leveraging current toolsets and update mechanisms.

Ansible Modules

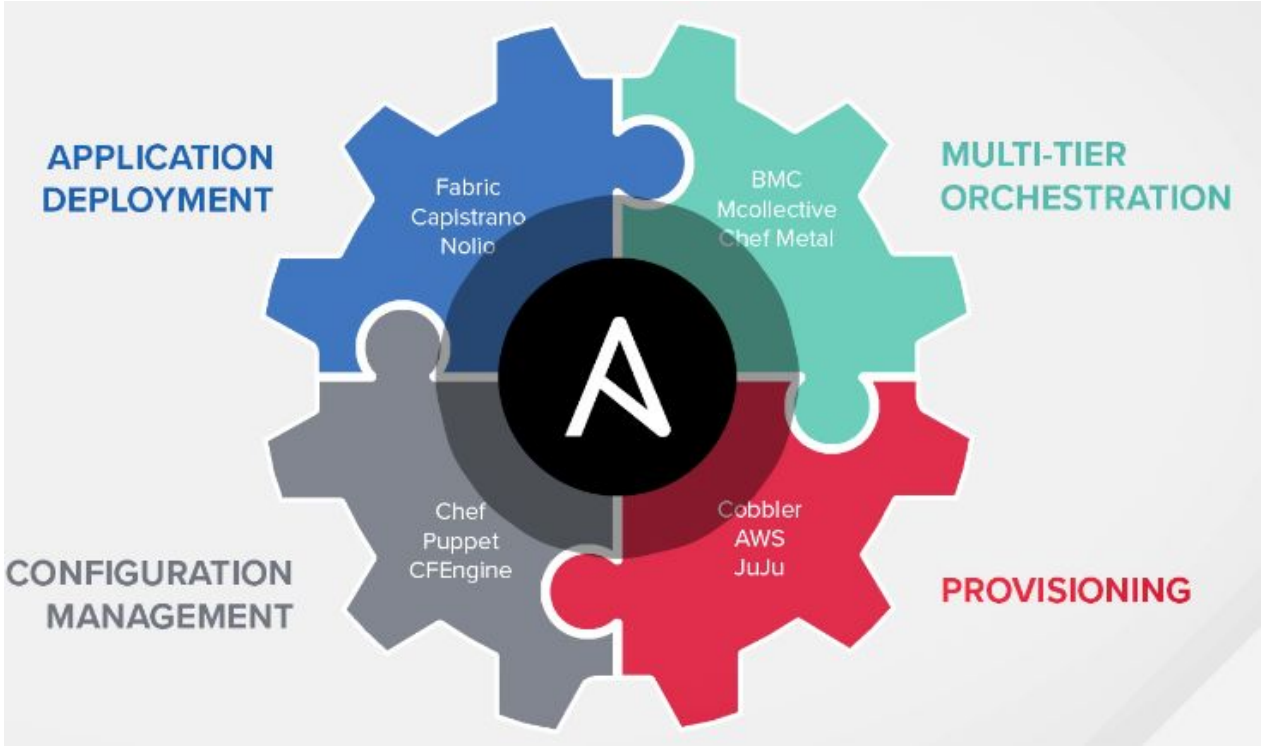
– – –

- Cloud
- Containers
- Database
- Files
- Messaging
- Monitoring
- Network
- Notifications
- Packaging
- Source Control
- System
- Testing
- Utilities
- Web Infastructure

http://docs.ansible.com/ansible/modules_by_category.html

Ansible is Complete Package

– – –

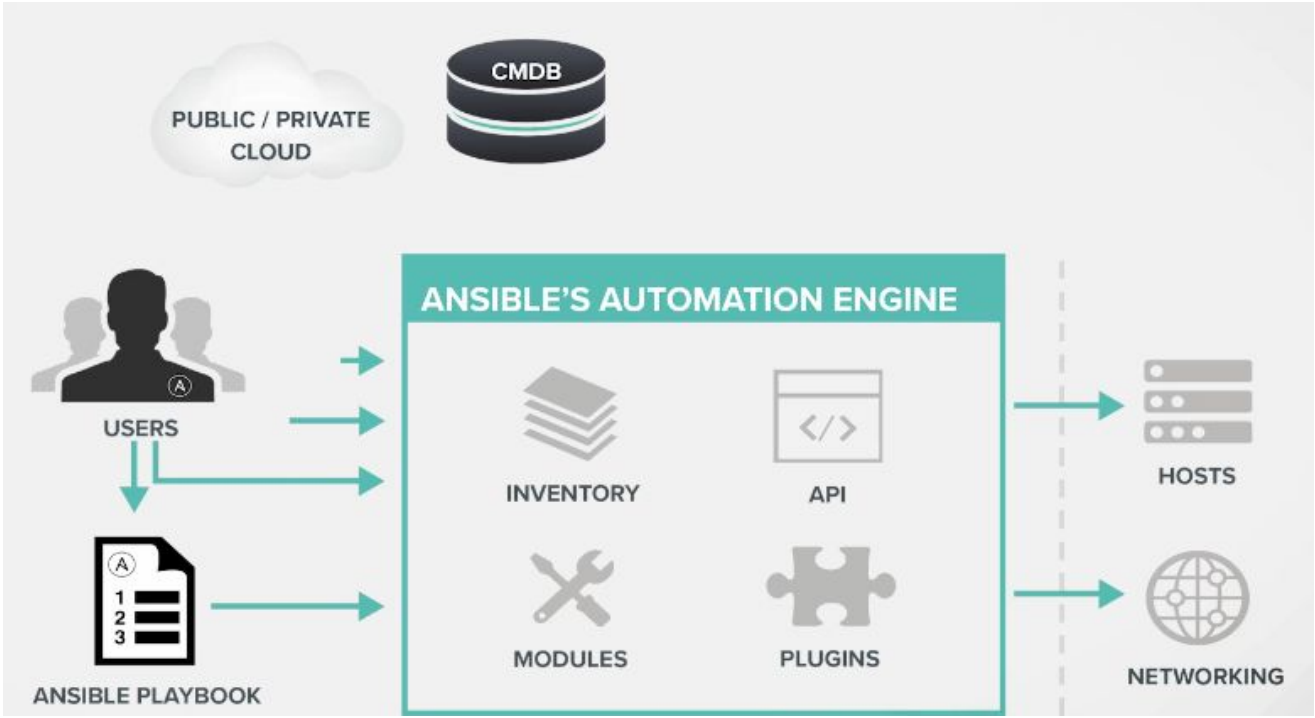


Use Cases

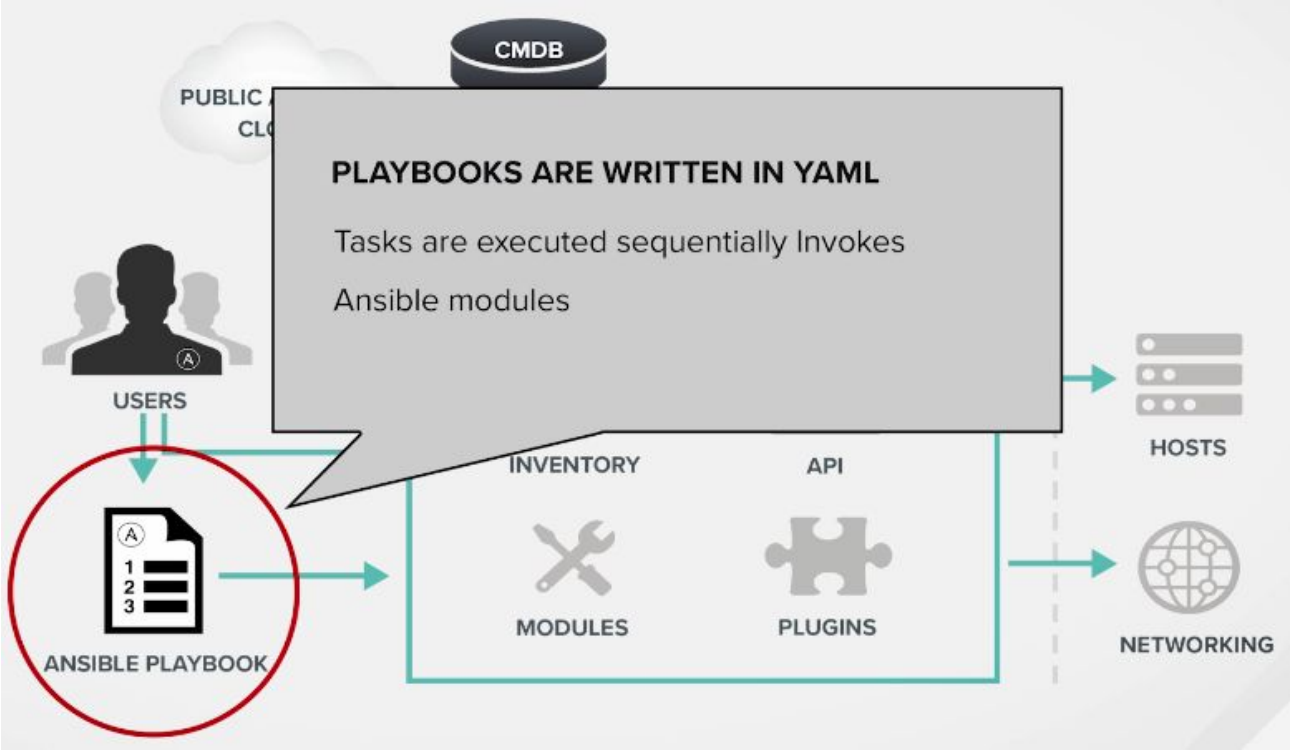
– – –

- Configuration Management
- Security and Compliance
- Application Deployment
- Orchestration
- Continuous Delivery
- Provisioning

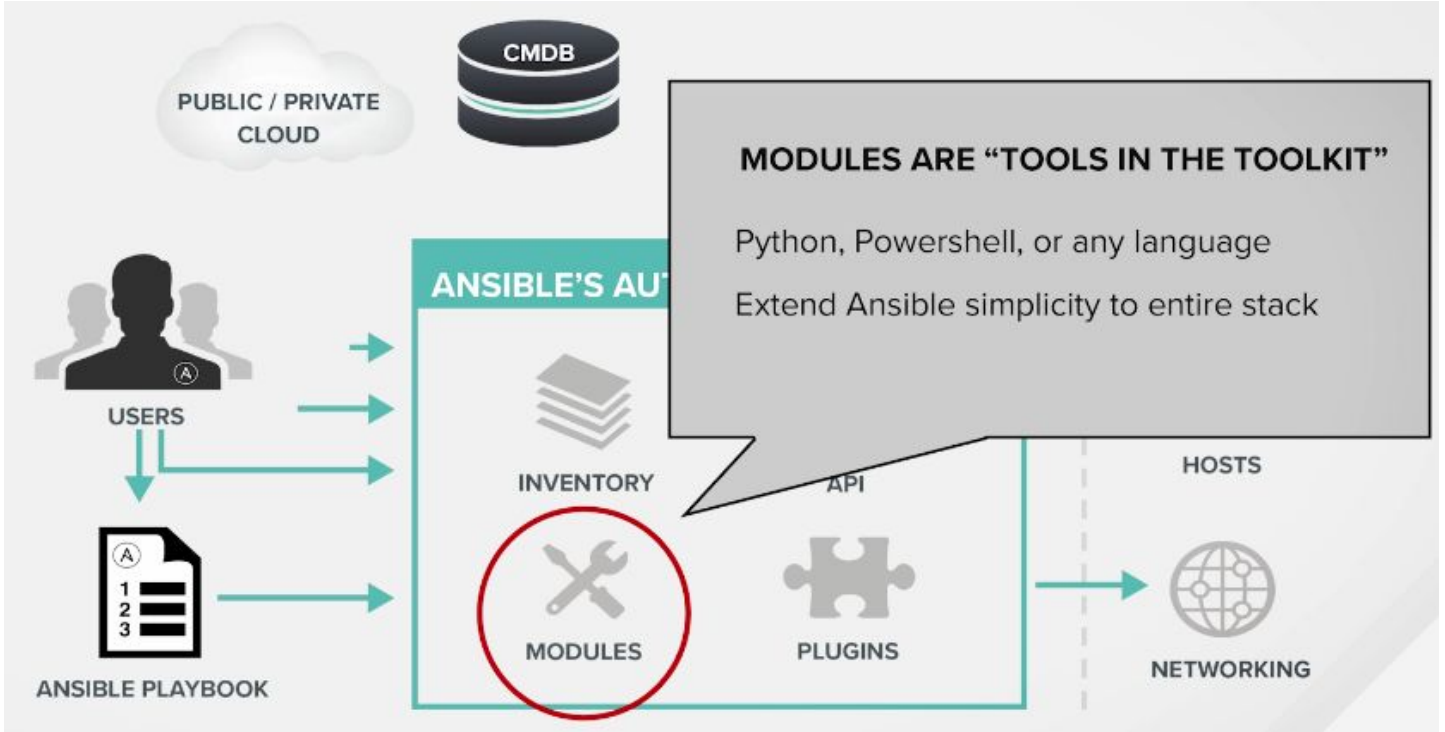
How Ansible Works (0)



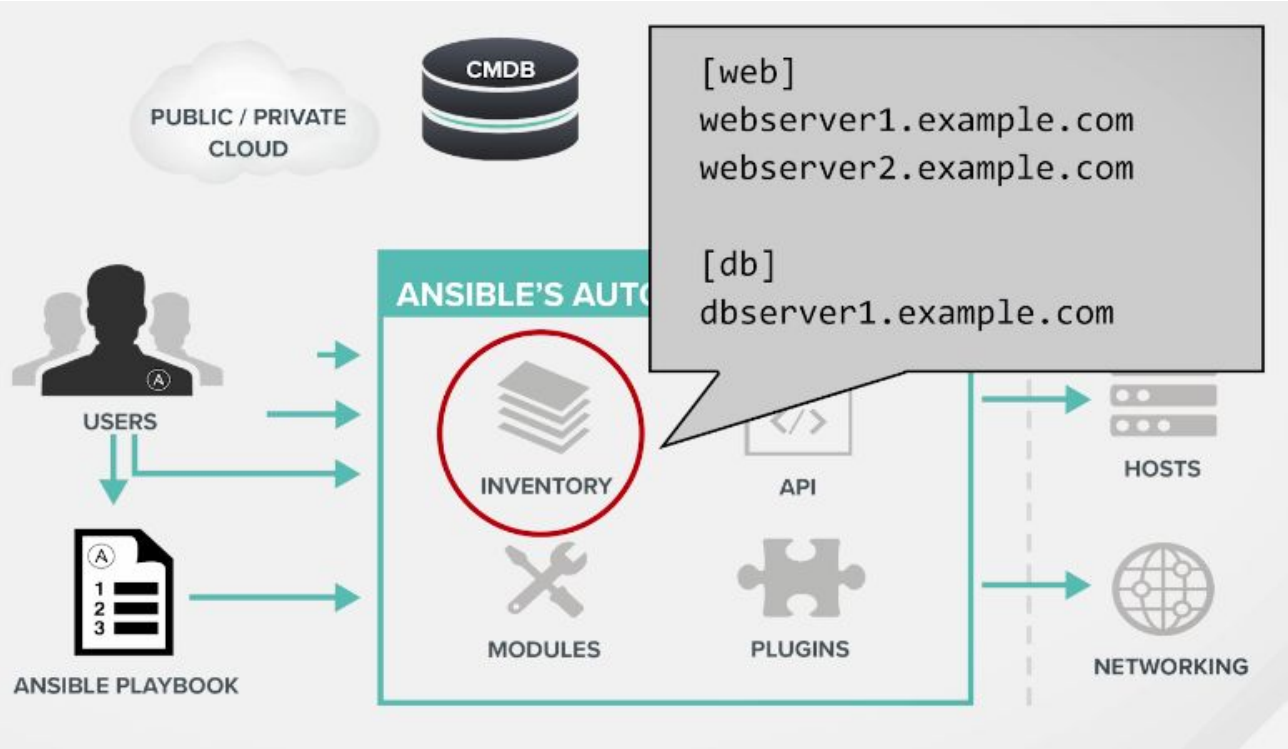
How Ansible Works (1)



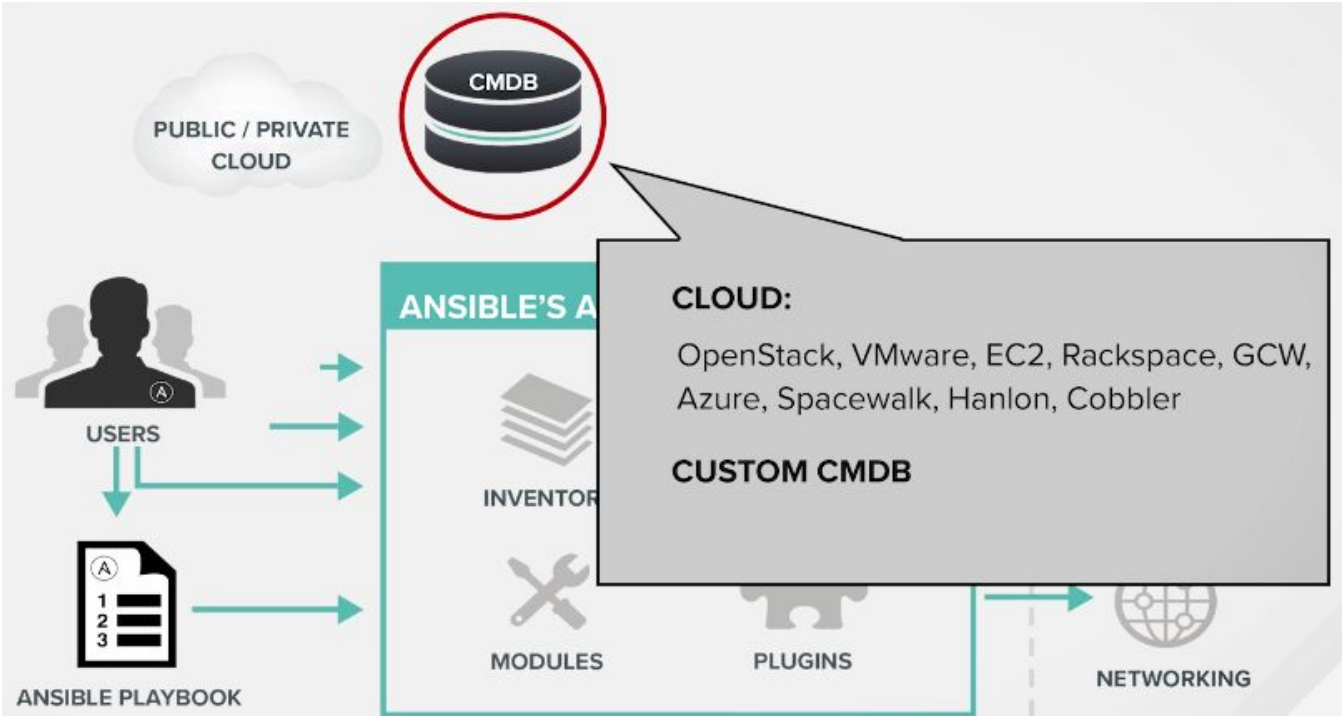
How Ansible Works (2)



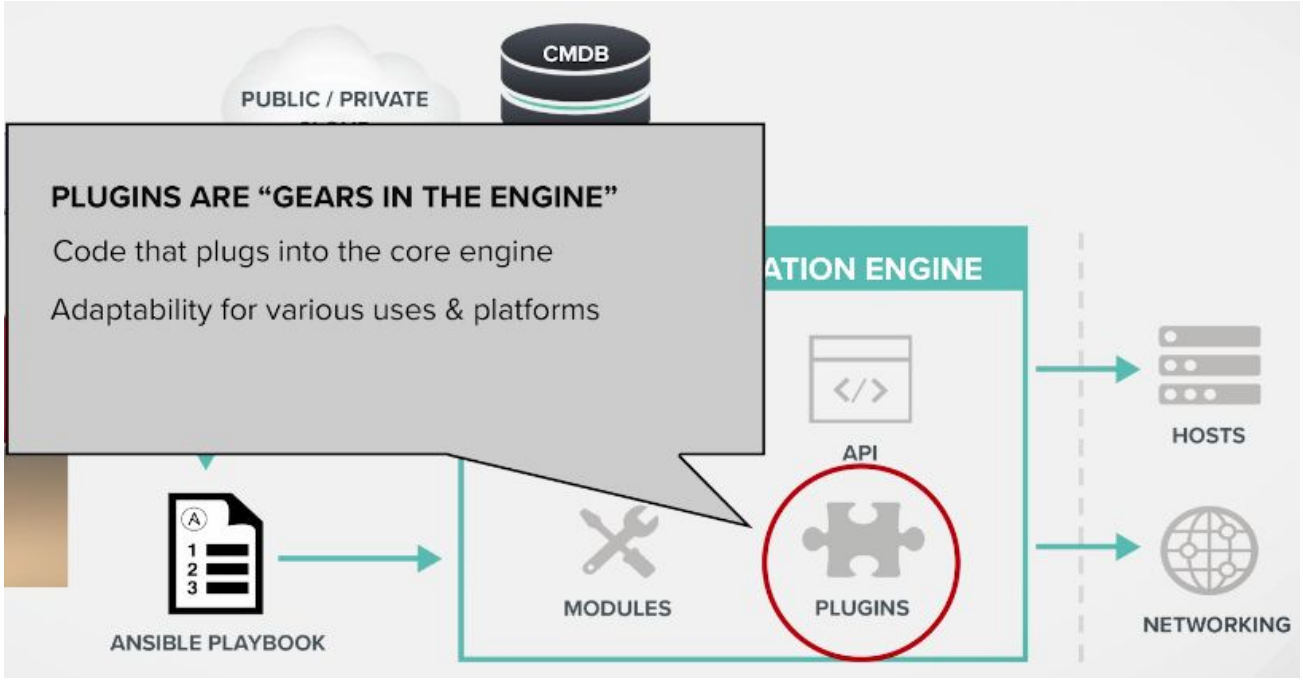
How Ansible Works (3)



How Ansible Works (4)



How Ansible Works (5)



Automation with Ansible

Modules

Modules

- apt/yum
- copy
- file
- get_url
- git
- ping
- debug
- service
- synchronize
- template
- url
- user
- wait_for
- assert

Modules

– – –

http://docs.ansible.com/ansible/modules_by_category.html

Modules: Run Commands

– – –

- **command**: Takes the command and executes it. The most secure and predictable.
- **shell**: executes through a shell like /bin/sh so you can use pipes etc
- **script**: Runs a local script on a remote node after transferring it.
- **raw**: Executes a command without going through the Ansible module susystem.

Ad-Hoc Commands

```
# check all my inventory hosts are ready to be
# managed by Ansible
$ ansible all -m ping

# run the uptime command on all hosts in the
# web group
$ ansible web -m command -a “uptime”

# collect and display the discovered for the
# localhost
$ ansible localhost -m setup
```

Discovered Facts

```
$ ansible localhost -m setup
localhost | success >> {
    "ansible_facts": {
        "ansible_default_ipv4": {
            "address": "192.168.1.37",
            "alias": "wlan0",
            "gateway": "192.168.1.1",
            "interface": "wlan0",
            "macaddress": "c4:85:08:3b:a9:16",
            "mtu": 1500,
            "netmask": "255.255.255.0",
            "network": "192.168.1.0",
            "type": "ether"
        },
```

Automation with Ansible

Inventory

Inventory

Inventory is a collection of hosts (nodes) against which Ansible can work with.

- Hosts
- Groups sources
- Inventory-specific data
- Static or dynamic

Satic Inventory Example

```
10.42.0.2
10.42.0.6
10.42.0.7
10.42.0.8
10.42.0.100
```

```
[control]
control ansible_host=10.42.0.2

[web]
node-1 ansible_host=10.42.0.6
node-2 ansible_host=10.42.0.7
node-3 ansible_host=10.42.0.8

[haproxy]
haproxy ansible_host=10.42.0.100

[all:vars]
ansible_user=vagrant
ansible_ssh_private_key_file=~/.vagrant.d/insecure_private_key
```

Automation with Ansible

Playbooks

Variables

Ansible can work with metadata from various sources and manage their context in the form of variables.

Variable Precedence

1. Extra vars
2. Task vars (only for the task)
3. Block vars (only for tasks in the block)
4. Role and include vars
5. Play vars_files
6. Play vars_prompt
7. Play vars
8. Set_facts
9. Registered vars
10. Host facts
11. Playbook host_vars
12. Playbook group_vars
13. Inventory host_vars
14. Inventory group_vars
15. Inventory vars
16. Role defaults

Variables

– – –

- **file**: A directory should exist
- **yum**: A package should be installed
- **service**: A service should be running
- **template**: Render a config file from a template
- **get_url**: fetch an archive file from a URL
- **git**: Clone a source code repository

Tasks Example

```
tasks:
  - name: add cache dir
    file:
      path: /opt/cache
      state: directory

  - name: install nginx
    yum:
      name: nginx
      state: latest

  - name: restart nginx
    service:
      name: nginx
      state: restarted
```

Handler Tasks

Handlers are special tasks that run at the end of a play if notified by another task.

If a configuration file gets changed notify a service restart task it needs to run.

Handler Example

```
tasks:
  - name: add cache dir
       file:
      path: /opt/cache
      state: directory

  - name: install nginx
    yum:
      name: nginx
      state: latest
    notify: restart nginx

handlers:
  - name: restart nginx
    service:
      name: nginx
      state: restarted
```

Plays and Playbooks

Plays are ordered sets of tasks to execute against host selections from your inventory.

A playbook is a file containing one or more plays.

Playbook Example

```
---
- name: install and start apache
  hosts: web
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root

  tasks:
  - name: install httpd
    yum: pkg=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
  - name: start httpd
    service: name=httpd state=started
```

Automation with Ansible

Roles

Roles

Roles are a packages of closely related Ansible content that can be shared more easily than plays alone.

Project with Embedded Roles

```
site.yml
roles/
   common/
     files/
     templates/
     tasks/
     handlers/
     vars/
     defaults/
     meta/
   webservers/
     files/
     templates/
     tasks/
     handlers/
     vars/
     defaults/
     meta/
```

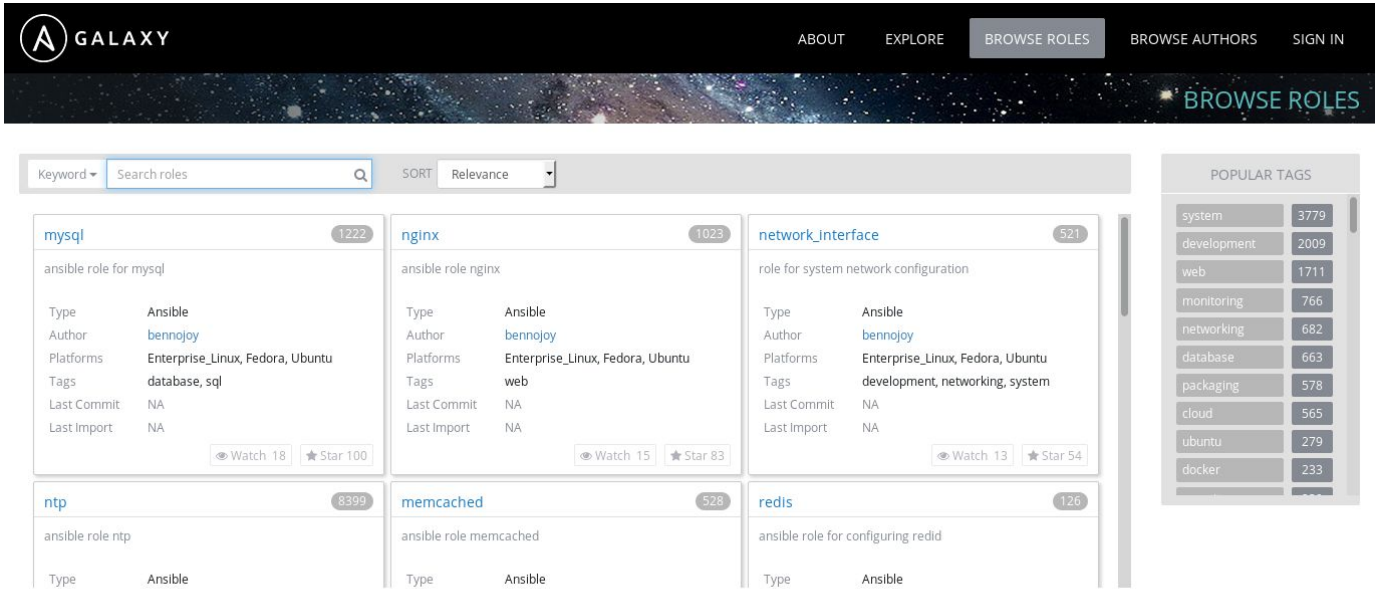
Playbook with Roles

```
# site.yml
---
- hosts: web
  roles:
     - common
     - webservers
```

Ansible Galaxy

– – –

https://galaxy.ansible.com



Automation with Ansible

Vault

Vault to Encrypt

_ _ _

- group_vars/files
- host_vars/files
- include_vars targets
- vars_files targets
- --extra-vars targets
- role variables
- Role defaults
- Task files
- Handler files