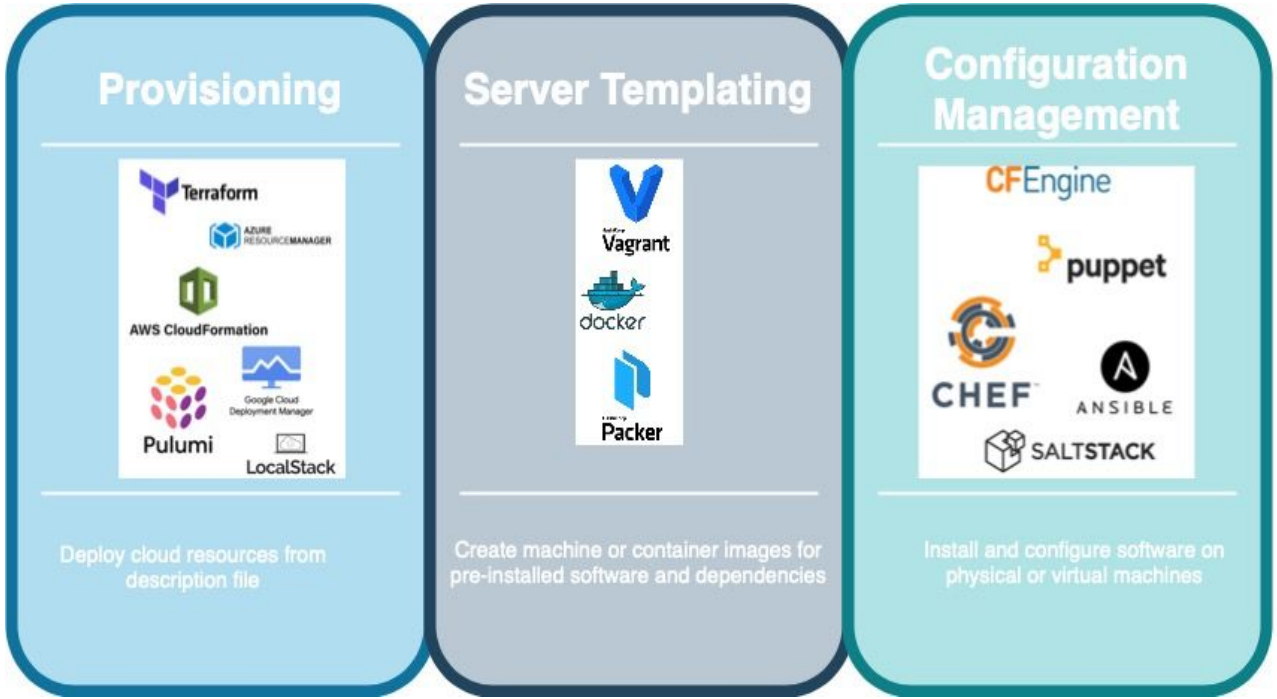
Ansible & CI/CD

Dr Assane Wade
Abir Chebbi
2023-2024

hepia
Haute école du paysage, d'ingénierie
et d'architecture de Genève

Hes·so GENÈVE
Haute Ecole Spécialisée
de Suisse occidentale

Configuration Management



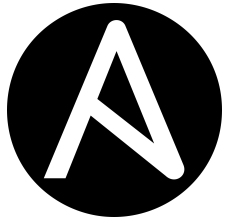
Ansible Characteristics

Agent-less architecture
Low maintenance overhead by avoiding the installation of additional software across IT infrastructure.

Simplicity
Automation playbooks use straightforward YAML syntax for code that reads like documentation. Ansible is also decentralized, using SSH existing OS credentials to access to remote machines.

Scalability and flexibility
Easily and quickly scale the systems you automate through a modular design that supports a large range of operating systems, cloud platforms, and network devices.

Idempotence and predictability
When the system is in the state your playbook describes Ansible does not change anything, even if the playbook runs multiple times.



Ansible Modules Providers

ARISTA	ATLASSIAN	aws	Microsoft Azure	Google Cloud Platform	CISCO
MORE INFO	MORE INFO	MORE INFO	MORE INFO	MORE INFO	MORE INFO
CUMULUS	JUNIPER NETWORKS	NetApp	vmware	Windows	IBM
MORE INFO	MORE INFO	MORE INFO	MORE INFO	MORE INFO	MORE INFO

Ansible uses cases

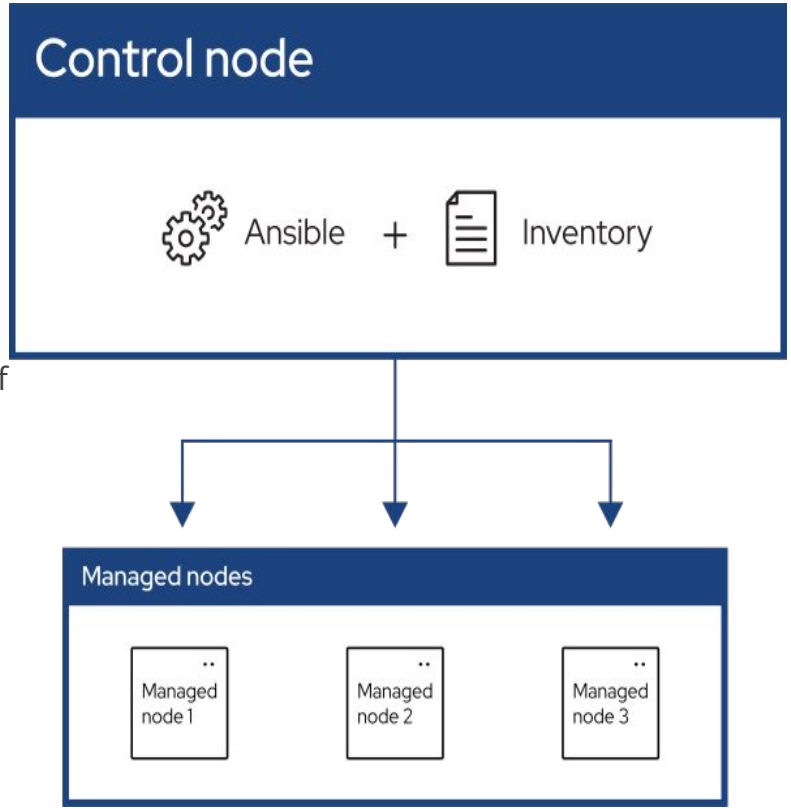
Ansible provides open-source automation that reduces complexity and runs everywhere. Using Ansible lets you automate virtually any task. Here are some common use cases for Ansible:

- Eliminate repetition and simplify workflows
- Manage and maintain system configuration
- Continuously deploy complex software
- Perform zero-downtime rolling updates

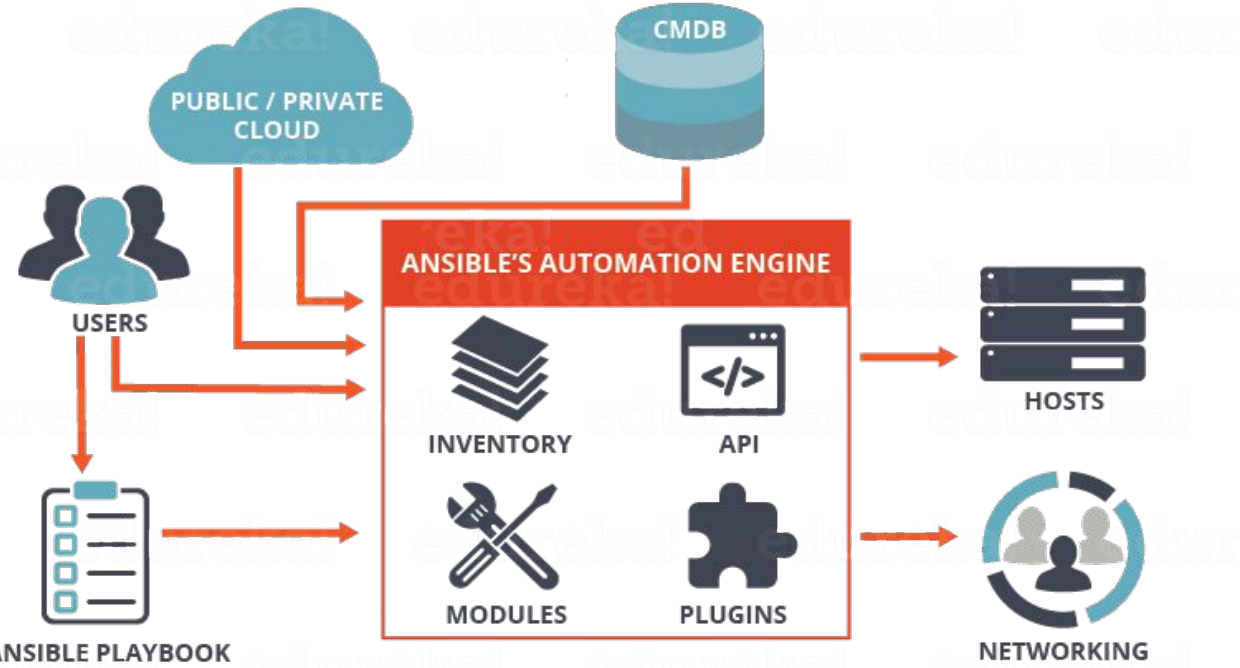
Ansible Architecture

Control node : The machine from which you run the Ansible CLI tools (ansible-playbook , ansible, ansible-vault and others). You can use any computer that meets the software requirements as a control node - laptops, shared desktops, and servers can all run Ansible. You can also run Ansible in containers known as Execution Environments. Multiple control nodes are possible, but Ansible itself does not coordinate across them,

Managed nodes : Also referred to as 'hosts', these are the target devices (servers, network appliances or any computer) you aim to manage with Ansible. Ansible is not normally installed on managed nodes, unless you are using ansible-pull, but this is rare and not the recommended setup.



Ansible Architecture



Inventory (Hostfile)

A list of managed nodes provided by one or more ‘inventory sources’. Your inventory can specify information specific to each node, like IP address. It is also used for assigning groups, that both allow for node selection in the Play and bulk variable assignment.

```
[my_servers]
server1 ansible_host=192.168.1.1
server2 ansible_host=192.168.1.2
```

Playbooks

Playbooks: They contain Plays (which are the basic unit of Ansible execution). This is both an ‘execution concept’ and how we describe the files on which ansible-playbook operates.

Playbooks are written in YAML and are easy to read, write, share and understand.

Playbook

Play: The main context for Ansible execution, this playbook object maps managed nodes (hosts) to tasks. The Play contains variables, roles and an ordered lists of tasks and can be run repeatedly. It basically consists of an implicit loop over the mapped hosts and tasks and defines how to iterate over them.

```
- name: My First Playbook
  hosts: my_servers
  tasks:
    - name: Ensure Apache is installed
      yum:
        name: httpd
        state: present
    - name: Ensure Apache is running
      service:
        name: httpd
        state: started
```

Roles & tasks

Roles: A limited distribution of reusable Ansible content (tasks, handlers, variables, plugins, templates and files) for use inside of a Play.

To use any Role resource, the Role itself must be imported into the Play.

Tasks: The definition of an ‘action’ to be applied to the managed host. You can execute a single task once with an ad hoc command using ansible or ansible-console (both create a virtual Play).

```
tasks:
  - name: Ensure
Apache is installed
  yum:
    name: httpd
    state: present
```

Variables

Playbooks can make use of variables to make them more flexible and reusable. Variables can be defined at various levels, including at the playbook level, in tasks, or in separate variable files.

```
- name: Ensure Apache is installed
  yum:
    name: "{{ apache_package_name }}"
    state: present
```

Modules

The code or binaries that Ansible copies to and executes on each managed node (when needed) to accomplish the action defined in each Task.

Each module has a particular use, from administering users on a specific type of database to managing VLAN interfaces on a specific type of network device.

You can invoke a single module with a task, or invoke several different modules in a playbook. Ansible modules are grouped in collections.

Example: Module copy

```
- name: Copy configuration file
  copy:
    src: /path/to/config.conf
    dest: /etc/myapp/config.conf
```

Handlers & Modules

Handlers: A special form of a Task, that only executes when notified by a previous task which resulted in a ‘changed’ status.

```
- name: Restart Apache if configuration changes
  service:
    name: httpd
    state: restarted
    become: true
    notify: Restart Apache
```

Plugins & Collections

Plugins : Pieces of code that expand Ansible’s core capabilities. Plugins can control how you connect to a managed node (connection plugins), manipulate data (filter plugins) and even control what is displayed in the console (callback plugins).

Collections : A format in which Ansible content is distributed that can contain playbooks, roles, modules, and plugins. You can install and use collections through Ansible Galaxy.

Collection resources can be used independently and discretely from each other.

Ansible Common Commands

Task	Command	Example
Ad-Hoc Commands	ansible <group_name> -a "command_to_run"	ansible web_servers -a "uptime"
Png Test	ansible <group_name> -m ping	ansible all -m ping
Running Playbooks	ansible-playbook <playbook.yml>	ansible-playbook site.yml
Gathering Facts	ansible <group_name> -m setup	ansible web_servers -m setup
Copying Files	ansible <group_name> -m copy -a "src=<source_path> dest=<destination_path>"	ansible web_servers -m copy -a "src=/local/path/file.txt dest=/remote/path/"
Installing Packages	ansible <group_name> -m yum -a "name=<package_name> state=present"	ansible web_servers -m yum -a "name=nginxstate=present"
Checking Playbook Syntax	ansible-playbook --syntax-check <playbook.yml>	ansible-playbook --syntax-check site.yml
Encrypting Strings	ansible-vault encrypt_string "your_secret_data"	ansible-vault encrypt_string "your_secret_data"

Continuous Integration and Continuous Deployment (CI/CD)

Definition : A continuous integration and continuous deployment (CI/CD) pipeline is a series of steps that must be performed in order to deliver a new version of software. CI/CD pipelines are a practice focused on improving software delivery throughout the software development life cycle via automation.

Continuous Integration and Continuous Deployment (CI/CD)

- Develop higher quality code, faster and more securely.
- Even it’s possible to manually execute each of the steps of a CI/CD pipeline, the true value of CI/CD pipelines is realized through automation.

Continuous Integration and Continuous Deployment (CI/CD)

CI/CD Tools

	Jenkins	TeamCity	circleci	Bamboo	GitLab
Open source	Yes	No	No	No	No
Ease of use & setup	Medium	Medium	Medium	Medium	Medium
Built-in features	3/5	4/5	4/5	4/5	4/5
Integration	★★★★★	★★★	★★★★	★★★	★★★★
Hosting	On premise & Cloud	On premise & Cloud	On premise	On premise & Bitbucket as Cloud	On premise & Cloud
Free version	Yes	Yes	Yes	Yes	Yes
Build agent license pricing	Free	From $59 per month	From $15 per month	From $10 one-off payment	From $19 per month per user
Supported OSs	Windows, Linux, macOS, Unix-like OS	Linux or MacoS	Windows, Linux, macOS, Solaris, FreeBSD and more	Windows, Linux, macOS, Solaris	Linux distributions: Ubuntu, Debian, CentOS, Oracle Linux

https://katalon.com/resources-center/blog/ci-cd-tools

CI/CD Workflow Management