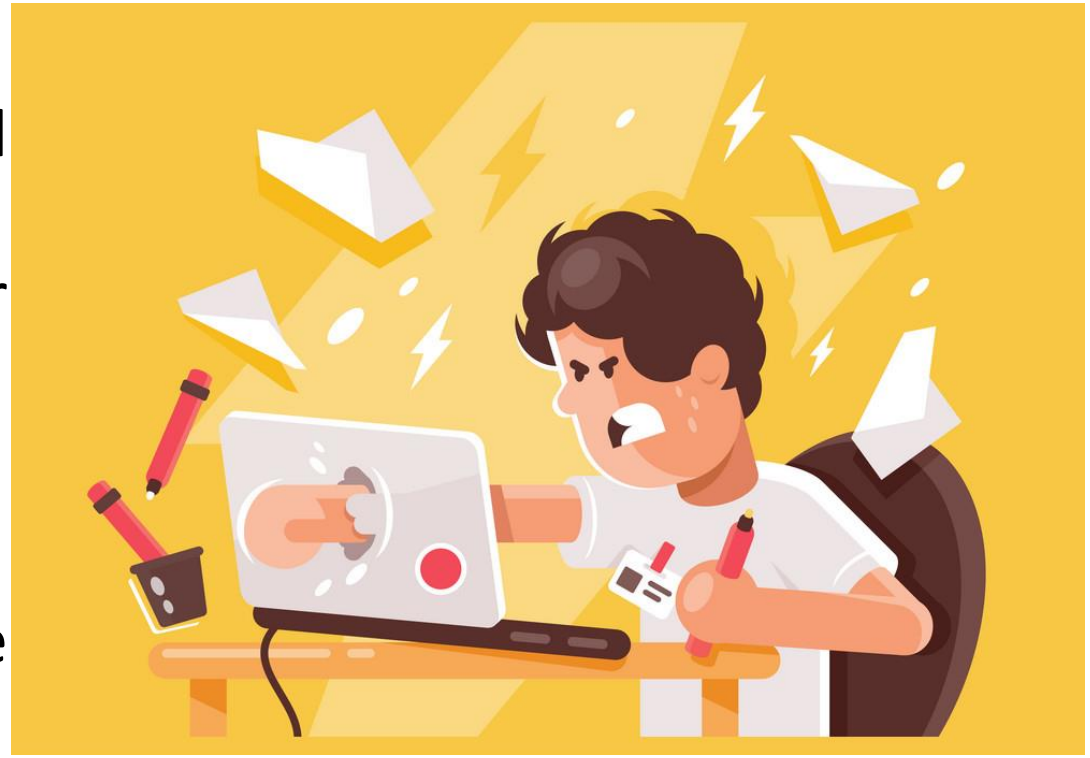

Presented by

Manhal Mohammed

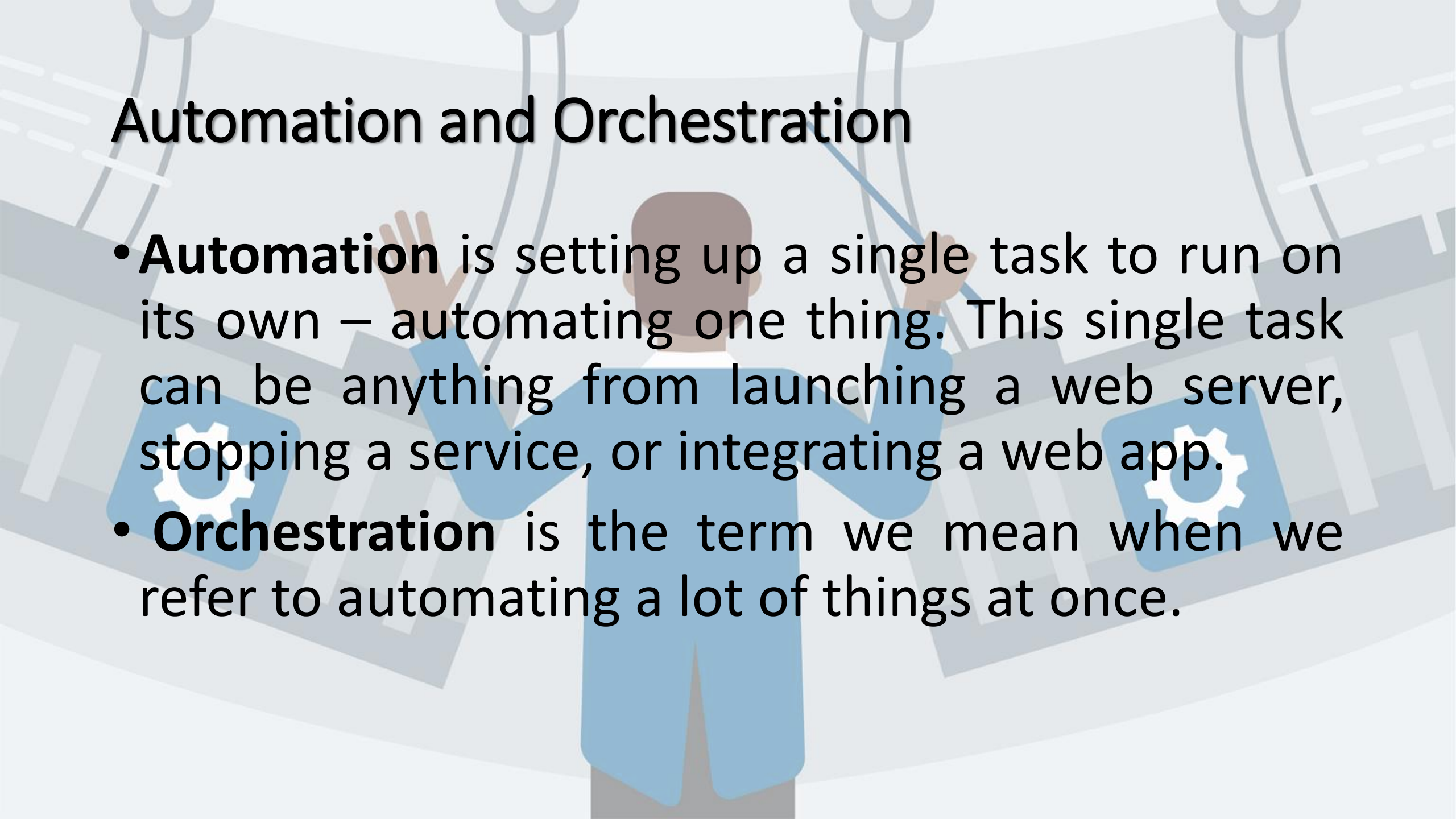
AFNOG 2019

# Manual Process

- Slower process
- Struggles in efficiently scale as demand changes
- Inconsistency in data entry, room for errors.
- Large ongoing staff training cost.
- is dependent on good individuals.
- Time consuming and costly to produce reports.
- Duplication of data entry.



# Automation and Orchestration



- **Automation** is setting up a single task to run on its own – automating one thing. This single task can be anything from launching a web server, stopping a service, or integrating a web app.
- **Orchestration** is the term we mean when we refer to automating a lot of things at once.

# Why Automation?

- ✓**Tasks in code**
- ✓**Collaboration**
- ✓**Eliminate errors**
- ✓**Write once**
- ✓**Laziness**
- ✓**Etc….**

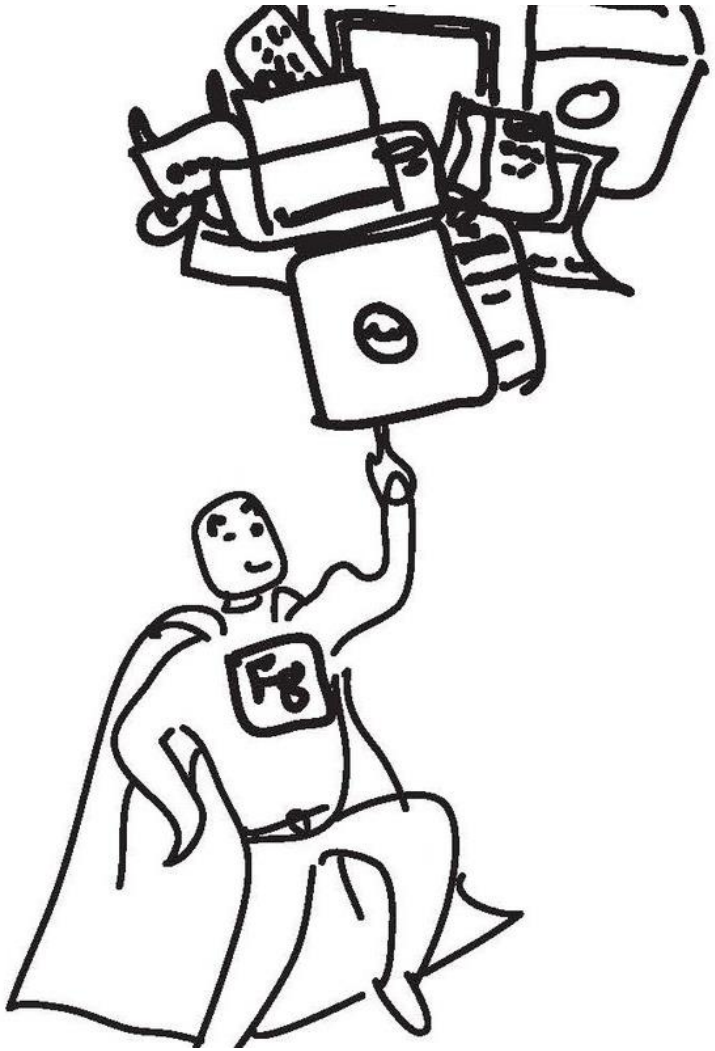

**Automation tools** :
Bamboo ,Docker, Kubernetes, Puppet , **Ansible**.

# Introduction to Ansible

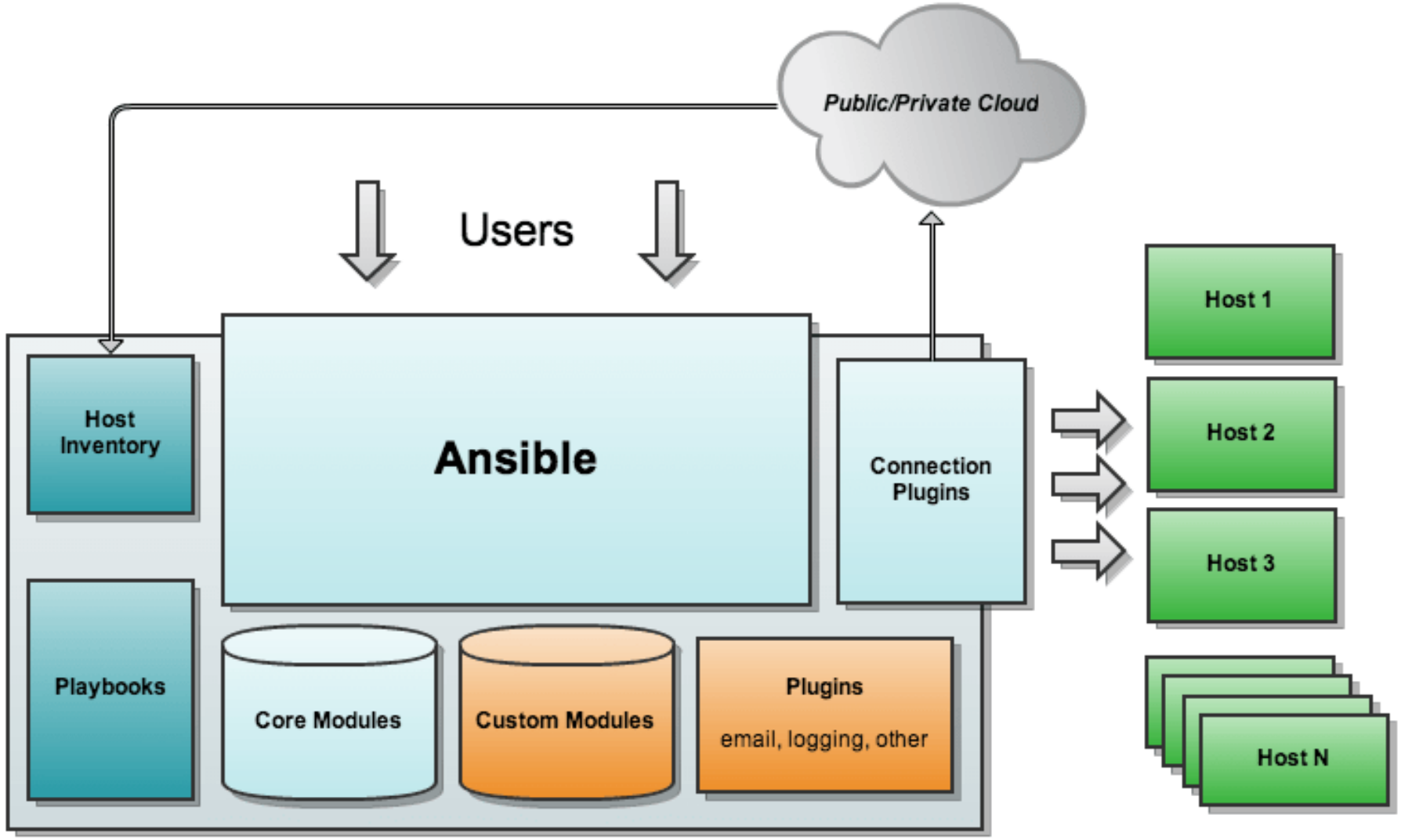
- Ansible is a radically simple IT automation platform that makes your applications and systems easier to deploy.
- It support configuration management with examples as below.
    - Configuration of servers
    - Application deployment
    - Continuous testing of already install application
    - Provisioning
    - Orchestration
    - Automation of tasks

# Why Ansible

- It is a free open source application
- Agent-less – No need for agent installation and management
- Phython/yaml based
- Highly flexible and configuration management of systems.
- Large number of ready to use modules for system management
- Custom modules can be added if needed
- Configuration roll-back in case of error
- Simple and human readable
- Self documenting



# Ansible Architecture



# Installation of Ansible

- Install packages below on the Server  Machine

  # sudo apt-get install ansible ; for Debian

  # sudo yum install ansible ; for REHL

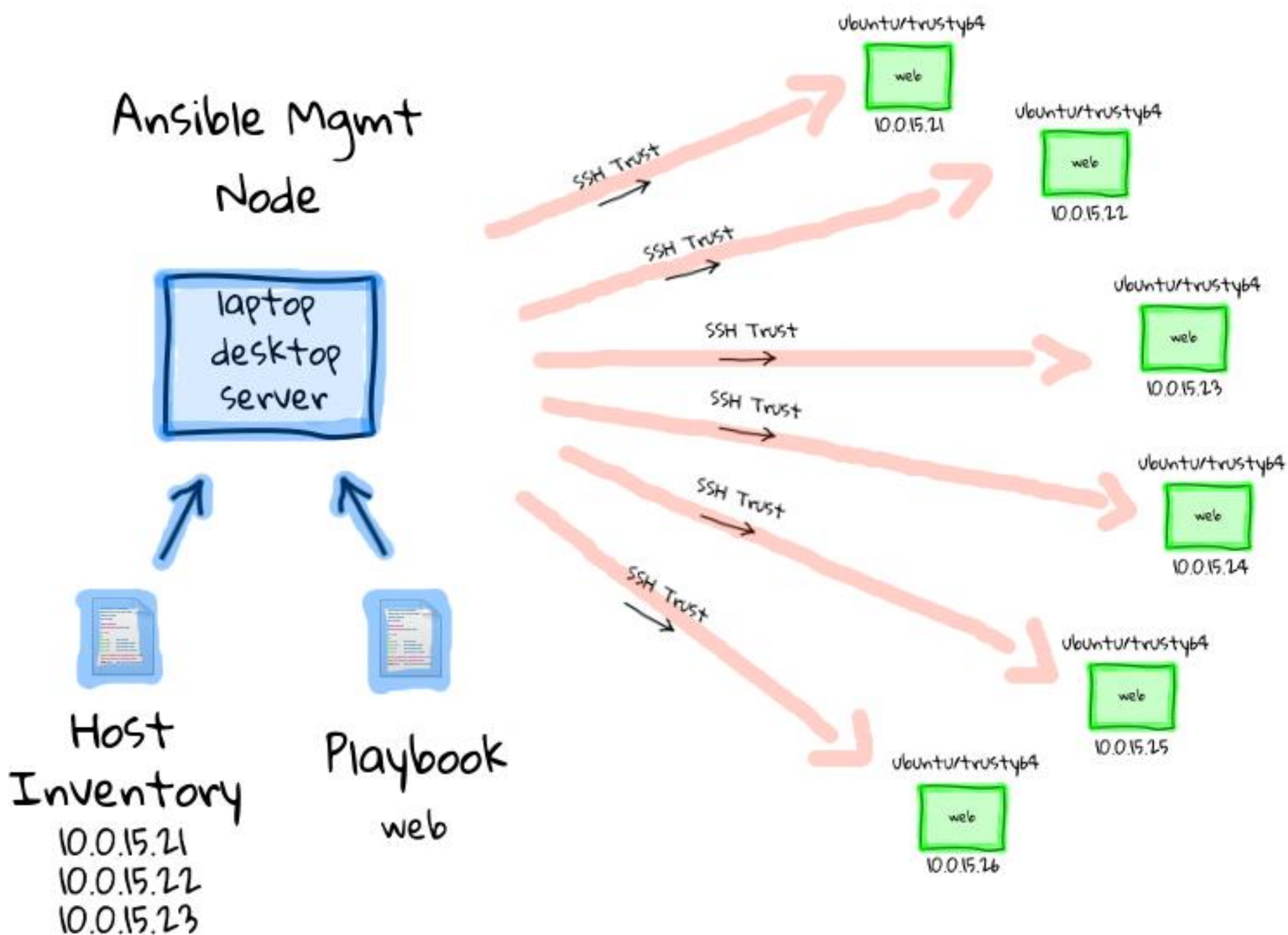
  # sudo pkg install py27-ansible ; for FreeBSD


# How Ansible Works

Ansible works by connecting to your nodes and pushing out small programs, called "Ansible modules" to them.

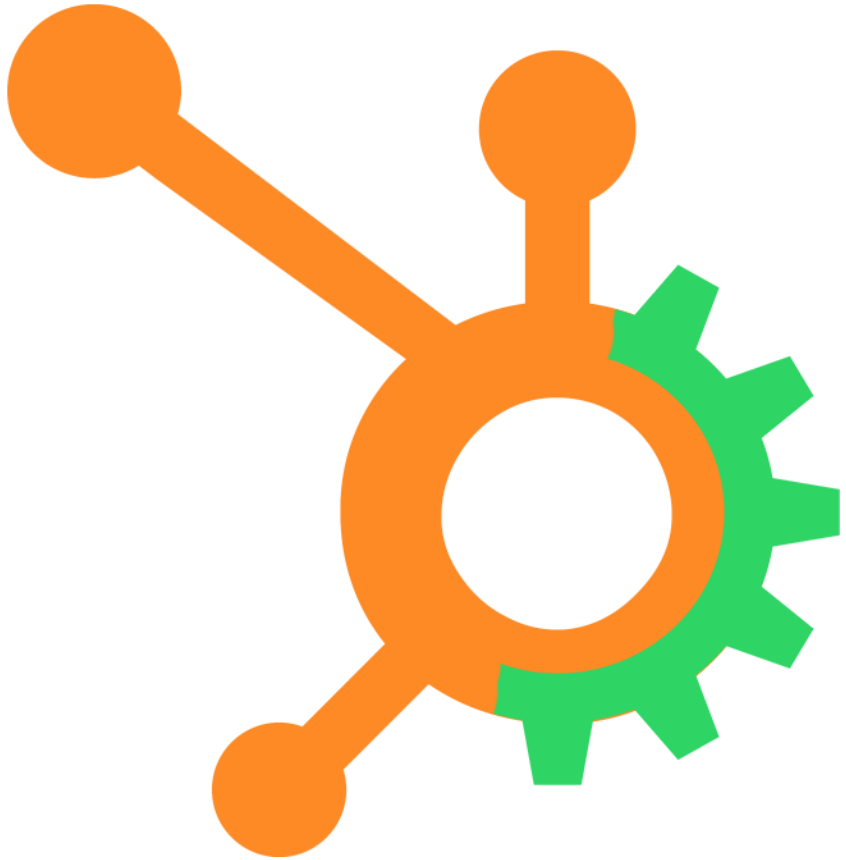
These programs are written to be resource models of the desired state of the system.

Ansible then executes these modules (over SSH by default), and removes them when finished.


Ansible Mgmt Node
laptop
desktop
server
Host Inventory
10.0.15.21
10.0.15.22
10.0.15.23
Playbook
web
SSH Trust
SSH Trust
SSH Trust
SSH Trust
SSH Trust
SSH Trust
ubuntu/trusty64
web
10.0.15.21
ubuntu/trusty64
web
10.0.15.22
ubuntu/trusty64
web
10.0.15.23
ubuntu/trusty64
web
10.0.15.24
ubuntu/trusty64
web
10.0.15.25
ubuntu/trusty64
web
10.0.15.26

# Modules

- Modules are bits of code transferred to the target system and executed to satisfy the task declaration.
- Ansible ships with several hundred today!
  - apt/yum/package
  - Copy/file
  - git
  - ping
  - debug
  - service
  - Template



**List out all modules installed**

```
# ansible-doc -l
```

**Read documentation for installed module**

```
# ansible-doc
```

**Modules Documentation :** https://docs.ansible.com/

# Modules: Run Commands

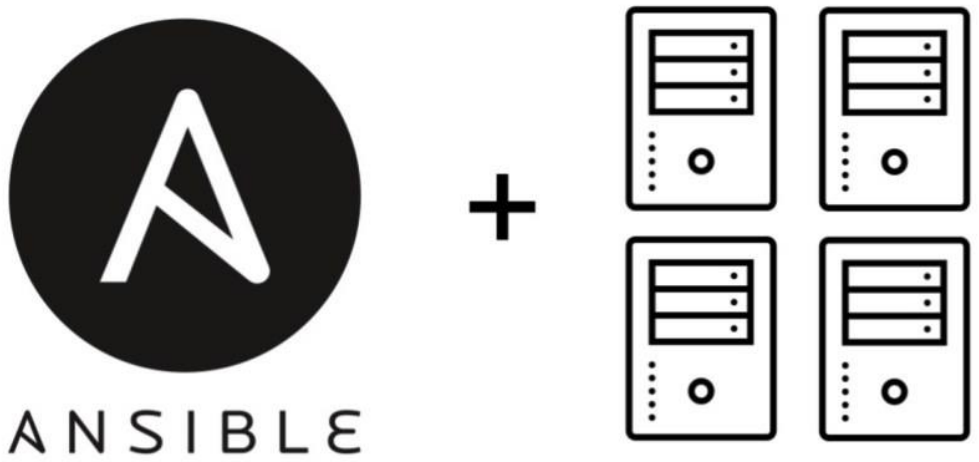
If Ansible doesn’t have a module that suits your needs there are the “run command” modules:

- **command**: Takes the command and executes it on the host. The most secure and predictable.
- **shell**: Executes through a shell like /bin/sh so you can use pipes etc. Be careful.
- **script**: Runs a local script on a remote node after transferring it.
- **raw**: Executes a command without going through the Ansible module subsystem “**clients with no pyhton installed** ”.

# Inventory

Inventory is a collection of hosts (nodes) with associated data and groupings that Ansible can connect and manage

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

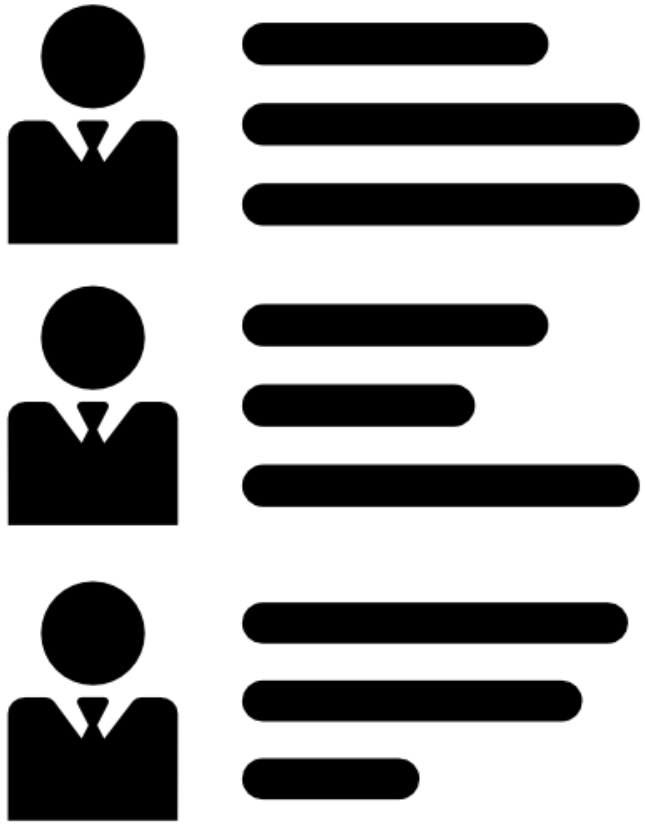



which defaults to be saved in the location /etc/ansible/hosts.

# Inventory example:

- Open the /etc/ansible/hosts and edit the following :

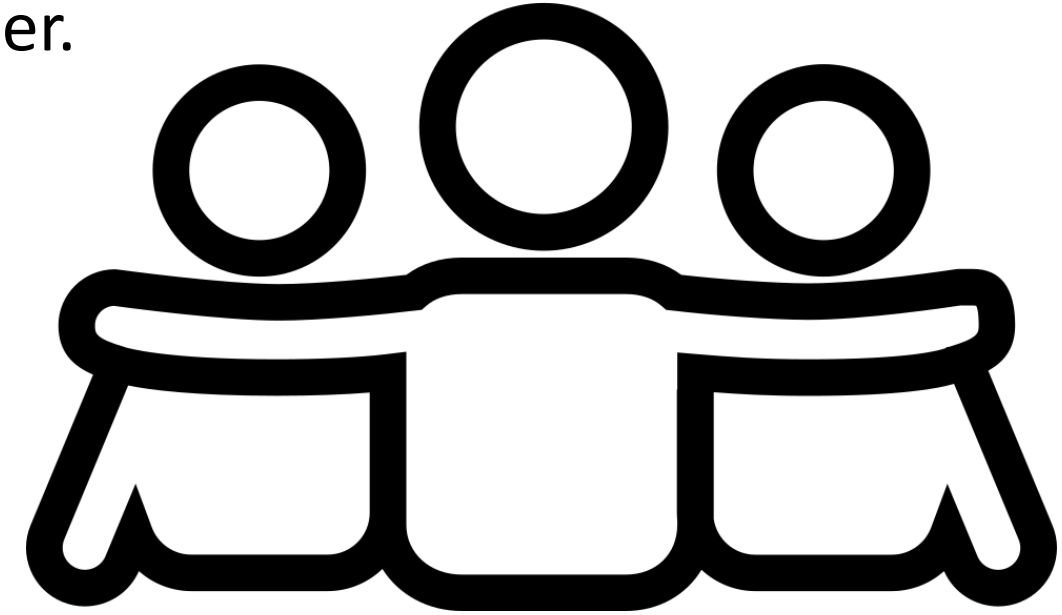
```
[me]
localhost
[group1]                                -> group of hosts name
pc1.sse.ws.afnog.org                    -> defining host
pc2.sse.ws.afnog.org
pc3.sse.ws.afnog.org    ansible_ssh_port=22
pc4.sse.ws.afnog.org    ansible_user=afnog
[group2]
pc[5:8].sse.ws.afnog.org
[all:vars]                              -> setting global variable for all groups
ansible_user=afnog                      -> specifying user to be connected with
ansible_sudo_pass=PassWord              -> passing the root password for sudo users
```



# SSH Keys Are Your Friends ☺

Passwords are supported by ansible , but SSH keys with ssh-agent are one of the best ways to use Ansible.

Though if you want to use Kerberos, that's good too. Lots of options! Root logins are not required, you can login as any user, and then su or sudo to any user.



# Creating KEY pairs:

- The first step is to create the key pair on your machine

  # ssh-keygen -t rsa
- Once you have entered the Gen-Key command, you will get a few more questions:
- Enter file in which to save the key (/home/test/.ssh/id_rsa):
- Enter no password for the next prompt
- Copy the Public Key

  # ssh-copy-id afnog@pcX.sse.ws.afnog.org
- Repeat the same process for other machines you wish to login automatically with.
- Ensure the **afnog** username has sudo access to the remote clients

# Ad-Hoc Commands

An ad-hoc command is a single Ansible task to perform quickly, but don't want to save for later.

- check all my inventory hosts are ready to be managed by Ansible

  # ansible all -m ping

- collect and display the discovered *facts for the localhost

  # ansible localhost -m setup

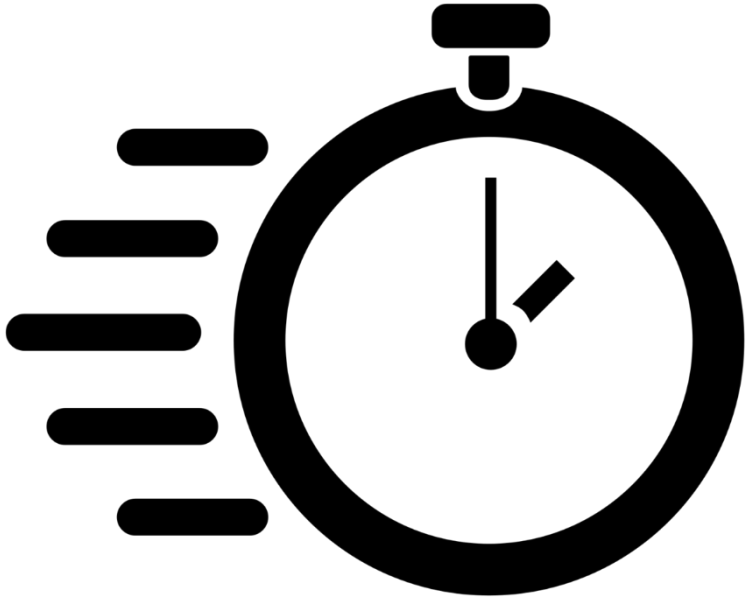
  # ansible  pcX.sse.ws.afnog.org -m setup -a "filter=*ipv4 "

- run the uptime command on all hosts in the group 2 group

  # ansible group2 -m command -a "uptime "

- fetch hard drives utilization

  # ansible -m command -a 'df -h' pcX.sse.ws.afnog.org



*** Ansible facts** are system properties that are collected by **Ansible** when it executes on a remote system. The **facts** contain useful details such as storage and network configuration about a target system.

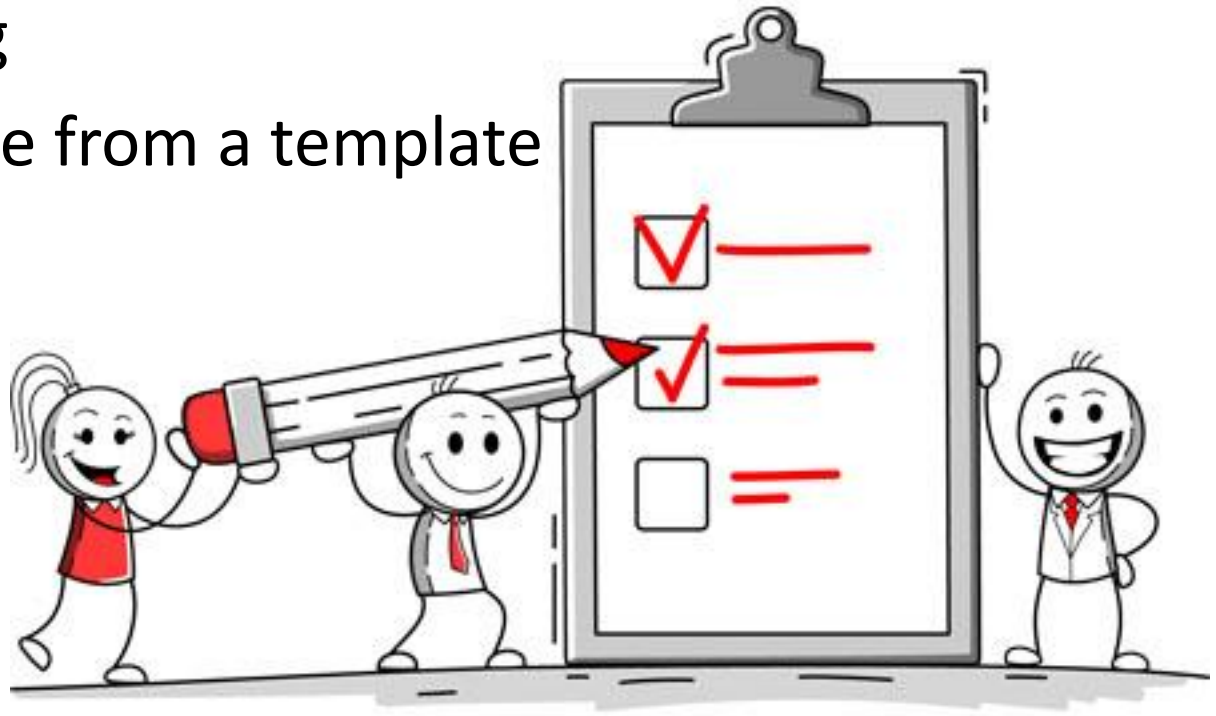
# Plays & Playbooks

- Plays are ordered sets of tasks to execute against host selections from your inventory.
- A playbook is a file containing one or more plays.
- Every playbook configuration begins with triple dash ( ---)
- The hosts, tasks, name, action are various instructions commands to help automate your play installation process , **using yaml and jinja2**
- **Jinja2** is a modern and designer-friendly templating language for Python.
- **YAML** is a human friendly data serialization standard for all programming languages. It is less and essentially allows you to provide powerful configuration settings, without having to learn a more complex code type.

# Tasks

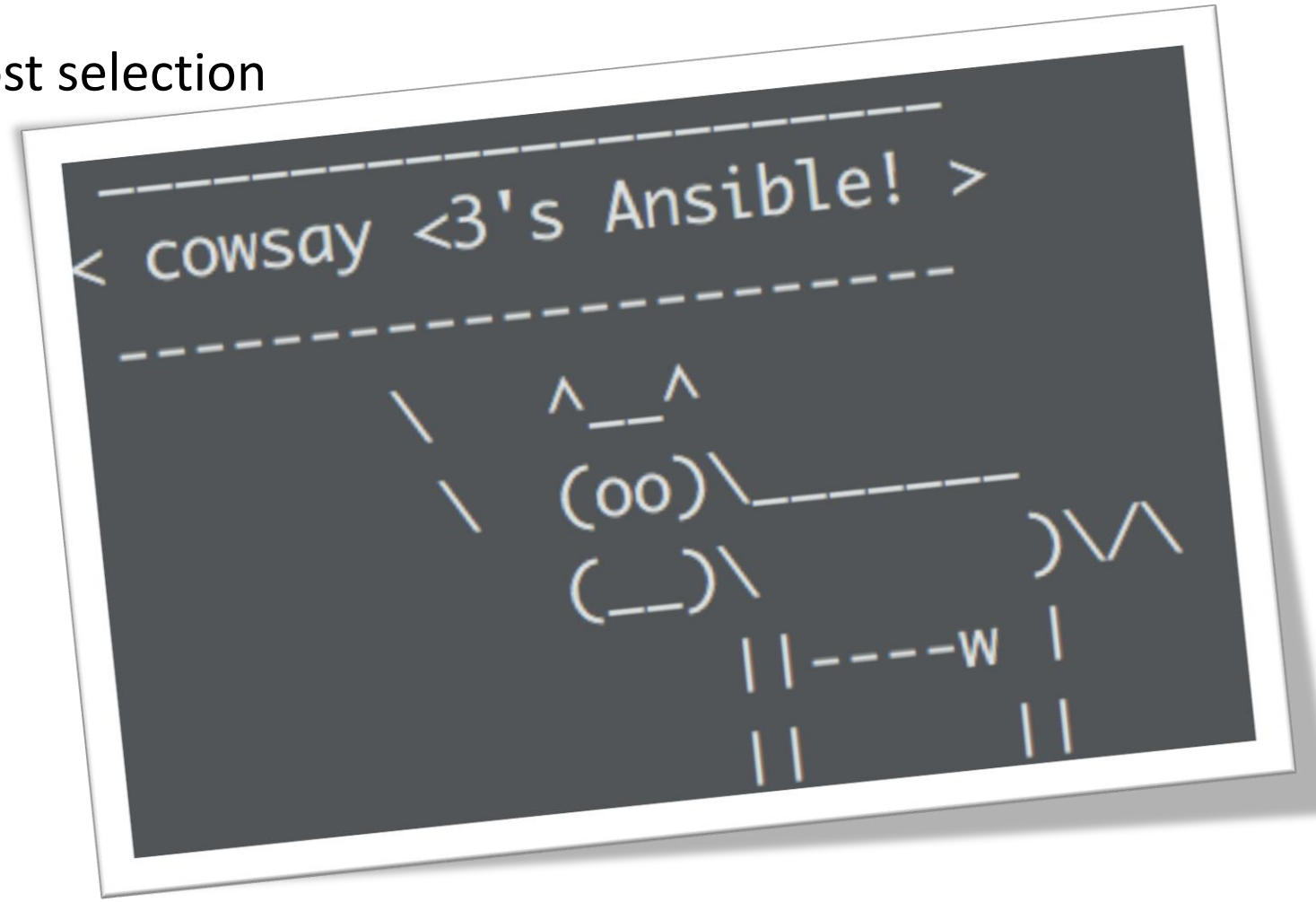
Tasks are the application of a module to perform a specific unit of work.

- **file**: A directory should exist
- **apt**: A package should be installed
- **service**: A service should be running
- **template**: Render a configuration file from a template



# Example Tasks in a Play

```
---
- hosts: me                          ; host selection
  tasks:
- name: INSTALLING COWSAY
  apt:
      name: cowsay
      state: latest
 - name: restart DNS
   service:
      name: bind9
      state: restarted
```



# Handler Tasks

- Handlers are special tasks that run at the end of a play if notified by another task when a change occurs.
- If a configuration file gets changed notify a service restart task that it needs to run.

```
tasks:
- name: httpd package is present
  yum:
     name: httpd
     state: latest
     notify: restart httpd
- name: latest index.html file is present
   copy:
        src: files/index.html
        dest: /var/www/html/
handlers:
- name: restart httpd
   service:
        name: httpd
        state: restarted
```

**References :**

- Ansible Hands-on Introduction : Jon Jozwiak, Minneapolis RHUG, 2017
- Introduction to Ansible : Frank Kuse, Afnog 2017