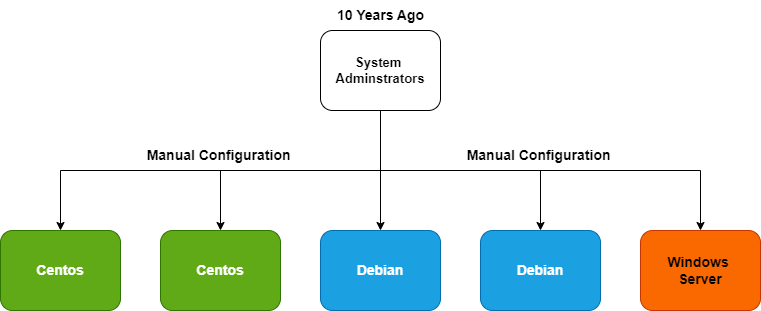
**ANSIBLE**

**Configuration Management**

Configuration management is a process for maintaining computer systems, servers, applications, network devices, and other IT components in a desired state. It’s a way to help ensure that a system performs as expected, even after many changes are made over time.



Previously 10 Years ago, configuration management will be handled by the system administrators. The major roles of system administrators are:

* **Distribution Versions are Up-to Date:** They need to check whether the versions of Linux are up-to date or not & also the supported files are installed in the linux systems.
* **System Level Dependencies are Up-to Date:** They need to check packages are up-to date (OpenSSH/wget/Curl/Shared Libraries etc.,) Suppose if Java is installed, then they need to check the latest version installed or not, they need to installed web servers, app servers and also right version with secure installation.
* **Application Dependencies are Up-to Date:** They also need to check application dependencies are installed or not.
* **Maintenance:** They need to do maintenance like CPU, Memory resources etc.,
* They need to write shell scripts which may get fails because some scripts may work on centos & some scripts may work on Debian and some on windows servers respectively.

Suppose the system admin has got the task to update the version of java, then he has to login to linux with centos and run a command, similarly login to linux with Debian and has to run a command. It was a very tedious activity. Here, the life of System Admins became like hell.

So, the concept of configuration management came into place. Some of the Configuration Management tools are Puppet, Chef, Salt, Ansible etc.,

Puppet & Chef both acts as an interface between system admin & physical servers. They both uses pull-based architecture.

**Architecture of Puppet & Chef:**

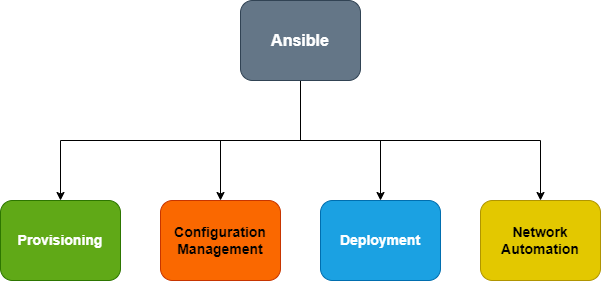


**Challenges using puppet & chef:**

* The Learning curve, because we need to learn ruby and also complex style of puppet and chef which was becoming difficult for system administrators.
* Additionally, both puppet and chef are not agentless in the architecture, because system admins need to go to each of the physical servers and they have to install agent (software).
* For every 30 minutes, manage nodes will connect to the control node and check for any new configuration.

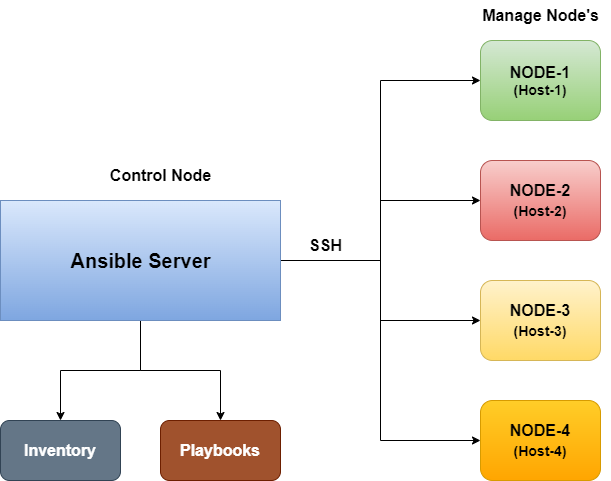
**Ansible (Configuration Management Tool)**

Ansible is an open-source IT automation engine that automates provisioning, configuration management, application deployment, orchestration, and many other IT processes. It is free to use, and the project benefits from the experience and intelligence of its thousands of contributors.



Ansible started as a configuration management tool, but now it is a powerful automation platform where **configuration management** is one of its tasks. Apart from that ansible can also do **provisioning** (used for creating infra using the resources like EC2/S3/VM etc.,) & it is widely used for **deployment** in CI/CD (deploying artifacts to the target servers) & also ansible is used for **network automation**.

**Ansible Architecture:**



Ansible follows **push-based** model. The control node pushes configurations and the desired state to the managed nodes. This model allows for rapid configuration changes across multiple servers.

* **Control Node:** Ansible is installed on the control node, which runs all tasks and playbooks. There can be multiple control nodes. The Control Node makes an **SSH** connection and executes the small modules on the host's machine and install the software.

**Example:** ssh ec2-user@IPAddress -C “echo ‘Hello’ > /tmp/hello.txt”

* **Managed Nodes:** These are the servers that Ansible manages. Ansible doesn't need to be installed on these nodes, as the control node communicates with them using SSH or PowerShell.
* **Inventory:** The inventory file provides the list of hosts where the Ansible modules need to be run.
* **Playbooks:** An Ansible playbook is a list of plays which contains modules that are automatically executed on a specified group of hosts or inventory. Playbooks are used for configuration management and automation, and can help with tasks like deploying applications, provisioning infrastructure, and managing configurations.
  + **Structure:** Playbooks are written in YAML format and define a set of tasks and configurations in a structured format.
  + **Reusability:** Playbooks can be reused and put under source control, allowing you to push out new configurations or confirm the configuration of remote systems.
  + **Order:** Tasks are executed in the order they are written.
  + **Scope:** Playbooks can orchestrate multiple roles and plays across different groups and hosts.
  + **Idempotency:** If all the modules a playbook uses are idempotent, then the playbook itself is likely to be idempotent.

**Ad-hoc Commands**

It is the command issued from ansible server targeting node manually, basically on some emergency/ad-hoc purpose. An Ansible ad-hoc command uses the /usr/bin/ansible command-line tool to automate a single task on one or more managed nodes. ad hoc commands are quick and easy, but they are not reusable.

**Syntax:** ansible **[**pattern**]** -m **[**module**]** -a "[module options]"

**Example:**

1. ansible -i 172.31.41.249, all -e ansilbe\_user=ec2-user -e ansible\_password=DevOps321 -b -m dnf -a "name=nginx state=installed"
2. ansible -i 172.31.41.249, all -e ansilbe\_user=ec2-user -e ansible\_password=DevOps321 -b -m service -a "name=nginx state=started"

**Ansible Modules**

An Ansible module is a small program that performs actions on a local machine, application programming interface (API), or remote host. Modules are expressed as code, usually in Python, and contain metadata that defines when and where a specific automation task is executed and which users can execute it. In Ansible modules, module name is mandatory and we can pass arguments to it which are optional/mandatory.

The following 3 example modules are included with Ansible Core and a part of all Ansible installations:

* **dnf module** ([ansible.builtin.dnf](https://docs.ansible.com/ansible/latest/collections/ansible/builtin/dnf_module.html" \l "ansible-collections-ansible-builtin-dnf-module))- Installs, upgrades, and removes packages and groups with the dnf package manager—the default software package manager for Fedora.
* **service module** ([ansible.builtin.service](https://docs.ansible.com/ansible/latest/collections/ansible/builtin/service_module.html" \l "ansible-collections-ansible-builtin-service-module))- Manages services on remote hosts. Example actions include, but are not limited to, starting, stopping, and restarting a service.
* **command module** ([ansible.builtin.command](https://docs.ansible.com/ansible/latest/collections/ansible/builtin/command_module.html" \l "ansible-collections-ansible-builtin-command-module))- Executes commands on targets.

**Now we will start writing our First Playbook**

**1.** We will check the ping module with control node. Name of the Playbook is **01.play.yml**

**Playbook**

- name: ping the control node

  hosts: web **#To which hosts our ansible server wants to connect**

  tasks: **#List of tasks/modules/collections**

  - name: ping the server

    ansible.builtin.ping:

**Inventory File**

[web]

172.31.91.31 **#Private IPAddress of Control Node**

* Command to run the playbook

**ansible-playbook -i inventory.ini -e ansible\_user=ec2-user -e ansible\_password=DevOps321 01-play.yml**

**2.** We will run multiple plays in single playbook. Name of the Playbook is **02-multi-play.yml**

Writing multiple plays in single playbook is not recommended in ansible.

**Playbook**

#PLAY-01

- name: PLAY-01

hosts: web

tasks:

- name: PLAY-01 TASK-01

ansible.builtin.debug: **#printing statements during execution**

msg: "This is from PLAY-01 and TASK-01"

#PLAY-02

- name: PLAY-02

hosts: web

tasks:

- name: PLAY-02 TASK-01

ansible.builtin.debug:

msg: "This is from PLAY-02 and TASK-01"

**3.** We will write playbook for installing nginx on the control node. Name of the Playbook is **03-nginx.yml**