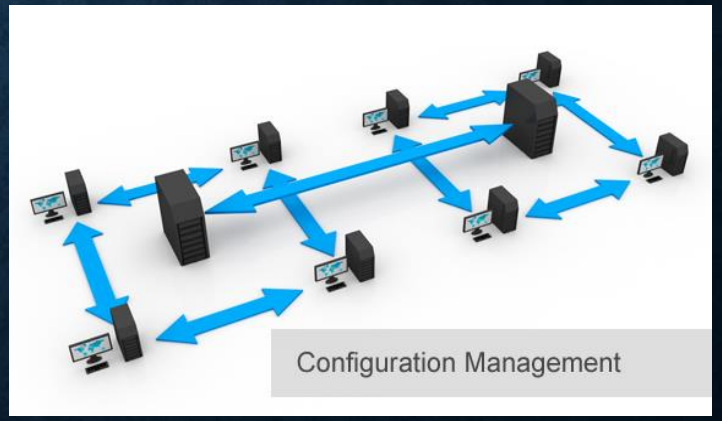

ANSIBLE

Configuration Management

- **Configuration Management**
  - It's a method through which we automate admin tasks.
  - Configuration Management tool turns your code into infrastructure
  - So your code would be Testable, Repeatable & Versionable.
- IT infrastructure refers to the composite of :
  - Software
  - Network
  - People
  - Process



Configuration Management

- Pain points :
  - Managing user & group accounts
  - Dealing with packages
  - Taking backup
  - Deploying all kinds of applications
  - Configure services

Why Configuration Management Tool?

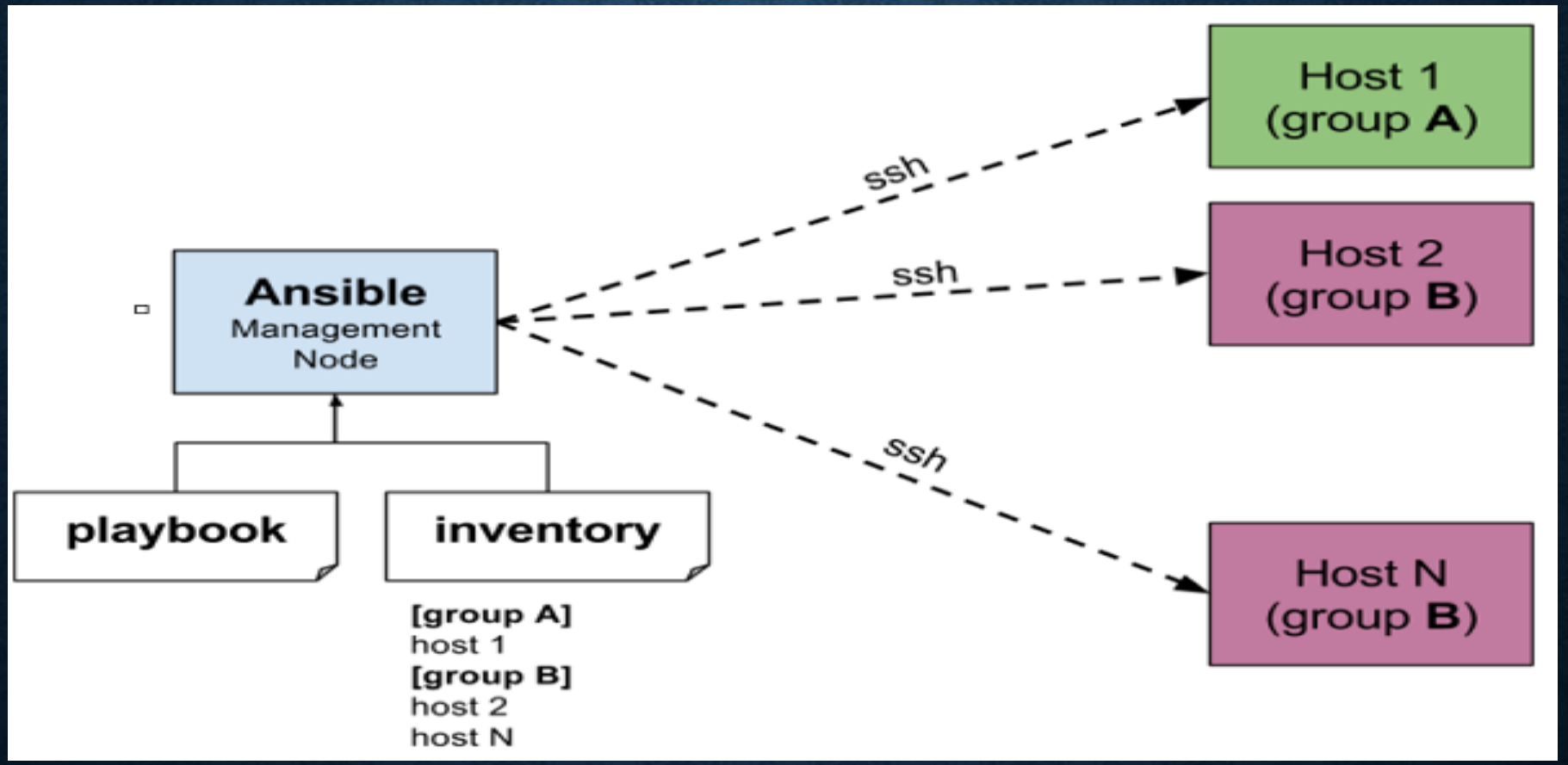
- Why Configuration Management ?
  - Complete Automation
  - Increase Uptime
  - Improve Performance
  - Ensure Compliance
  - Prevent Errors
  - Reduces Cost

Why Ansible?

- Ansible is an administration tool. What ever system admins (Linux/windows) used to do manually, now we are automating all those tasks by using Ansible (Any CM Tool)
- Can use this tool whether your servers are in on-premises or in the cloud.
- It turns your code into infrastructure i.e your computing environment has some of the same attributes as your application:
    - Your code is versionable.
    - Your code is repeatable.
    - Your code is testable.
- You only need to tell what the desired configuration should be, not how to achieve it
- Through automation, get desired state of server.

Why Ansible?

- Other tools in the market can be really complicated ..
  - huge overhead of Infrastructure setup
  - complicated setup
  - Pull mechanism
  - Lot of learning required
- Pros of Ansible :
  - ✓ Agentless
  - ✓ Relies on ssh
  - ✓ Uses python
  - ✓ Push mechanism


Host 1
(group A)
ssh
Host 2
(group B)
ssh
Ansible
Management
Node
ssh
playbook
inventory
Host N
(group B)
[group A]
host 1
[group B]
host 2
host N

Install & Configure Ansible(Server)

- Launch Amazon Linux (need not to install ansible in nodes)
- yum install wget -y
- wget http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
- yum install epel-release-latest-7.noarch.rpm -y
- sudo yum update -y
- sudo yum install git python python-devel python-pip openssl ansible -y
- ansible --version
- sudo vi /etc/ansible/ansible.cfg & enable the below lines

```
inventory = /etc/ansible/hosts
sudo_user = root
```

YAML(Yet Ain't Markup Language) Basics

- For Ansible, nearly every YAML file starts with a list
- Each item in the list is a list of key/value pairs, commonly called a "dictionary"
- All YAML files have to begin with "---" and end with "..."
- All members of a list lines must begin with same indentation level starting with "- "

```
--- # A list of tasty fruits
fruits:
  - Apple
  - Orange
  - Strawberry
  - Mango
...
```

- A dictionary is represented in a simple key: value form (the colon must be followed by a space)

```
--- # An employee record
Employee:
   name: SAI
   job: DevOps Engineer
   skill: Elite
...
```

Playbooks

- Each playbook is composed of one or more ‘modules’ in a list

- Playbooks are divided into many sections like….
  - **Target Section** - Defines the hosts against which playbooks tasks has to be executed
  - **Variable Section** - Defines variables
  - **Tasks Section** - List of all modules that we need to run, in an order

Our First Playbook

- All sections begin with "-" & its attributes & parameters beneath it
- Indentation is imp, use only spaces & not tabs
- Create a folder(playbooks) & go inside that(vi test.yml)
- test.yml

```
--- # My First YAML playbook
- hosts: demo
  tasks:
    - name: Install httpd on server
      action: yum pkg=httpd state=installed
```

- Run ansible-playbook to call the playbook

```
ansible-playbook test.yml
```

Target Section

- Create a file (vi first.yml)
- Example:

```
--- # My First YAML playbook
- hosts: demo
  user: ansible
  become: yes            # yes or no
  connection: ssh        # ssh or paramico
  gather_facts: yes      # yes or no
```

- Run ansible-playbook to call the playbook

ansible-playbook first.yml

Task Section

- Example:

```
--- # My First YAML playbook
- hosts: demo
  user: ansible
  become: yes
  connection: ssh
  tasks:
   - name: Install HTTPD on centos 7
     action: yum name=httpd state=installed
   - name: Install MYSQL on centos 7
     action: yum name=mysql state=installed
```

- Run ansible-playbook to call the playbook

ansible-playbook first.yml

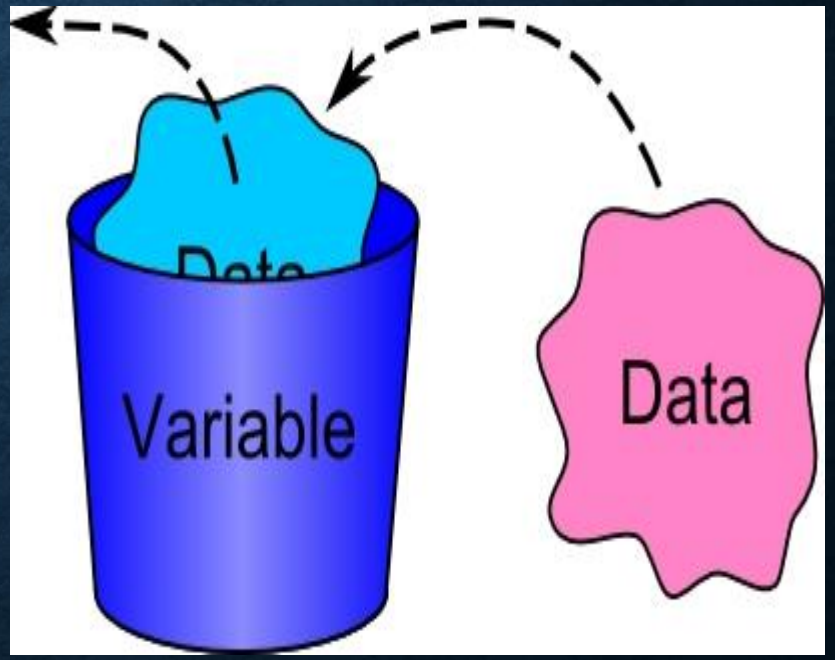
(Remove httpd package)

Variables: Inclusion Types

- Create a section called vars within a playbook
- Put vars above tasks so that we define it first & use it later

```
--- # My First YAML playbook
- hosts: demo
  user: ansible
  become: yes
  connection: ssh
  vars:
    pkgname: httpd
  tasks:
    - name: Install HTTPD server on centos 7
      action: yum name='{{pkgname}}' state=installed
```

(remove httpd package)



Handler Section

- Consists the ability to notify when something happens
- Also call another set of tasks

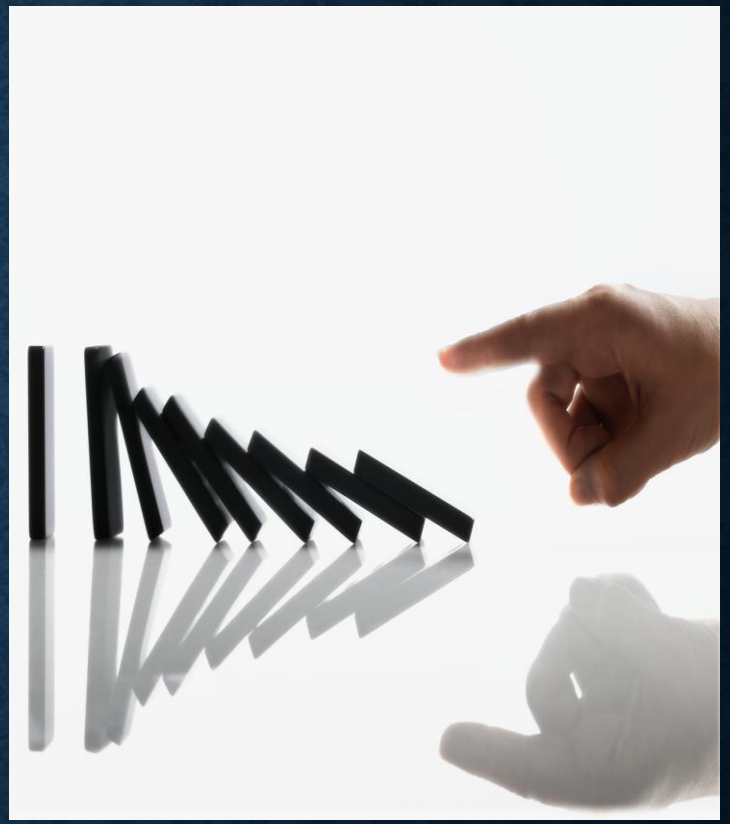
  (Remove apache from node)
- Example

```
--- # My First YAML playbook
- hosts: demo
  user: ansible
  become: yes
  connection: ssh
  tasks:
    - name: Install HTTPD server on centos 7
      action: yum name=httpd state=installed
      notify: restart HTTPD    #is called only if the action is ran & successful #
  handlers:
    - name: restart HTTPD    #has to match the notify name. Otherwise throws error #
      action: service name=httpd state=restarted
```

(Install httpd in both nodes & don't start service. Then try running above script. It won't start service)

- Run ansible-playbook to call the playbook

  ansible-playbook first.yml (Remove httpd from node)



Dry Run

- Check whether the playbook is formatted correctly
- Test how the playbook is going to behave without running the tasks

```
ansible-playbook webserver.yml --check
```

Loops

- Often you'll want to do many things in one task, such as create a lot of users, install a lot of packages, or repeat a polling step until a certain result is reached

- Example

```
--- # Loop Playbook
- hosts: demo
  user: ansible
  become: yes
  connection: ssh
  tasks:
    - name: add a list of users
      user: name='{{ item }}' state=present
      with_items:
        - Raj
        - Sai
        - Hari
```

Conditionals

- Few tasks might be needed to execute only on specific scenario

- **When statement**

  Sometimes you will want to skip a particular step on a particular host

- Example 1

```
--- # When playbook example
- hosts: demo
  user: ansible
  become: yes
  connection: ssh
  tasks:
    - name: Install apache for debian
      command: apt-get -y install apache2
      when: ansible_os_family == "Debian"
    - name: Install apache for redhat
      command: yum -y install httpd
      when: ansible_os_family == "RedHat“
```

cat /etc/os-rele*

Vault

- Ansible allows keeping sensitive data such as passwords or keys in encrypted files, rather than as plaintext in your playbooks

- Creating a new Encrypted playbook (Put ---) (Verify file permissions) (Open playbook now)

  ansible-vault create playbook.yml
- Edit the Encrypted playbook

  ansible-vault edit playbook.yml
- Change the password

  ansible-vault rekey playbook.yml
- Decrypt the playbook & verify by opening playbook

  ansible-vault decrypt playbook.yml
- Encrypt an existing playbook (verify by opening playbook)

  ansible-vault encrypt playbook.yml

Roles (Theory)

- Adding more & more functionality to the playbooks will make it difficult to maintain in a single file
- We can organize playbooks into a directory structure called roles

- **Creating Role Framework (eg:** playbook/roles/webserver)
  /master.yml

/master.yml

playbook/roles/webserver/tasks/main.yml

/vars/main.yml

/handlers/main.yml

Roles (Lab)

- master.yml (master.yml & roles folder must be in same directory level)
- vi roles/webserver/tasks/main.yml

```
- name: Install Apache on CentOS
  yum: pkg=httpd state=latest
```

playbook/roles/webserver/tasks/main.yml

/master.yml

/ansible-playbook master.yml (command run place)

Roles(Lab)

- **This is how we mention Role in play-books**

- **Example**

✓ Create master.yml (inside playbook folder)

```
--- # master playbook for web servers
- hosts: all
  user: ansible
  become: yes
  connection: ssh
  roles:
    - webserver
```

ansible-playbook master.yml (run being present in playbook folder)

easy_install pip
pip install ansible
ansible --version

ansible demo –b –m file -a "name=myfile state=touch"

yum update –y (Take RHEL machine)
rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
yum install ansible –y
ansibel --version

sudo rpm -ivh http://dl.fedoraproject.org/pub/epel/6/i386/epel-release-6-8.noarch.rpm
yum install ansible –y
ansible –version