Ansible – Configuration Management System

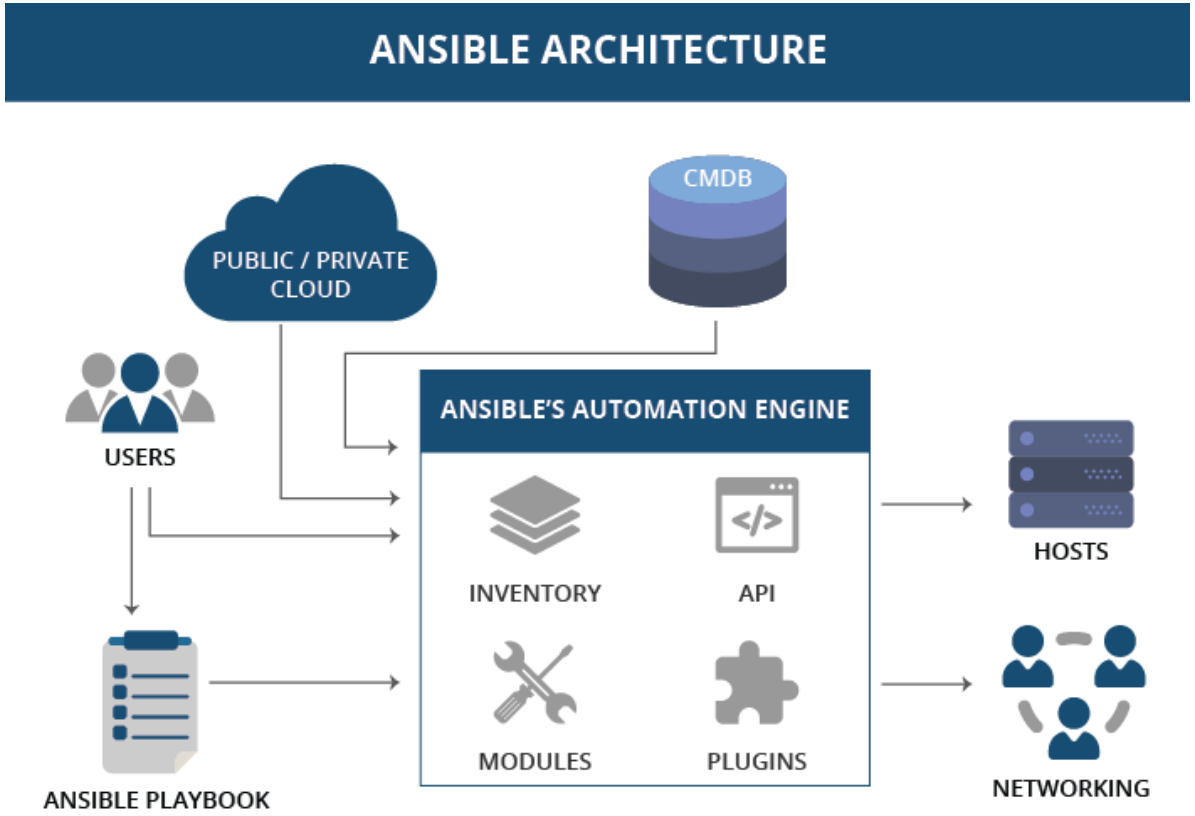
What is Ansible?.

- Ansible is a lightweight, open source IT automation tool
- Has an Agentless architecture
- Nodes do not require to install and run background daemons
- Reduces the pressure on the network

Why Ansible?.

- Provisioning
- Configuration management
- Continuous delivery
- Application deployment
- Security compliance
- Can be done through scripts(**Time consuming,Coding skills,Maintenance**)
- Powerful
- Agentless
- Simple

Ansible Architecture



Ansible Foundation

- Provisioning
- Configuration management
- Continuous delivery
- Application deployment
- Security compliance
- Can be done through scripts(**Time consuming,Coding skills,Maintenance**)
- Powerful
- Agentless
- Simple

Why Ansible?.

- Yaml
- Ansible Inventory
- Ansible Playbook
- Ansible Modules
- Ansible Variables
- Conditionals
- Loops
- Ansible Role

Yaml

- Yaml is a text based format
- Configuration file in Ansible written in Yaml

Xml

```
<Servers>
    <Server>
        <name>Server1</name>
        <owner>John</owner>
        <created>12232012</created>
        <status>active</status>
    </Server>
</Servers>
```

Json

```
{
    Servers: [
        {
        name: Server1,
        owner: John,
        created: 12232012,
        status: active,
        }
    ]
}
```

Yaml

```
Servers:
    -   name: Server1
        owner: John
        created: 12232012
        status: active
```

Yaml

Key value pair

```
Fruit: Apple
Vegetable: Carrot
Liquid: Water
Meat: Chicken
```

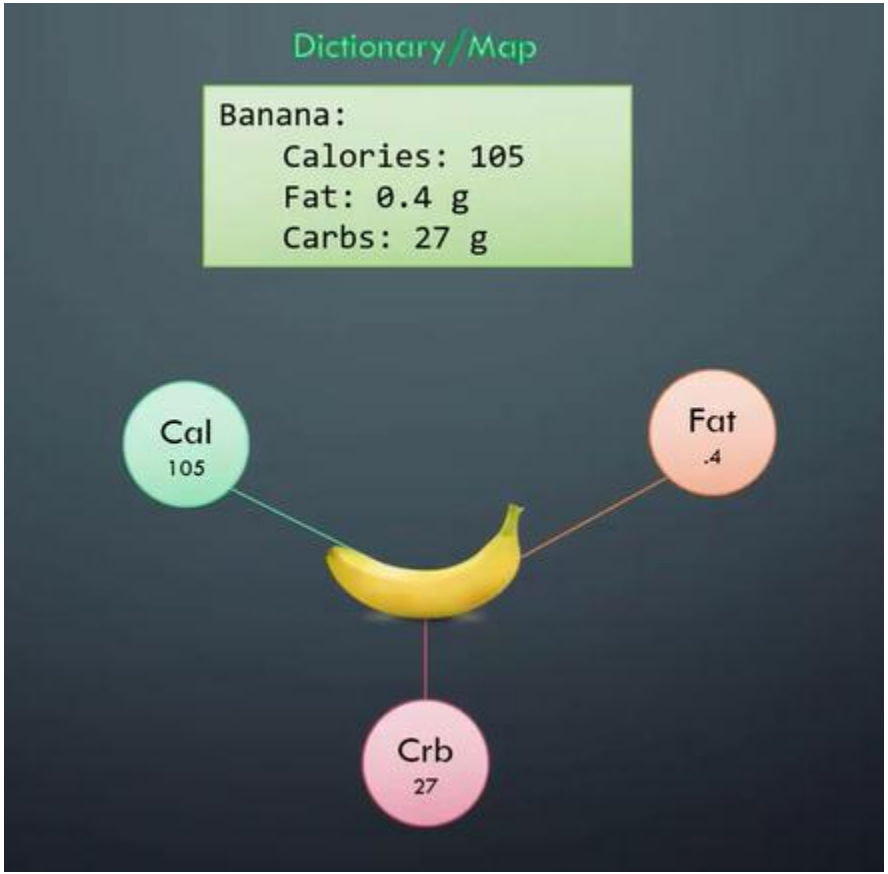
Array/List

```
Fruits:
-   Orange
-   Apple
-   Banana

Vegetables:
-   Carrot
-   Cauliflower
-   Tomato
```

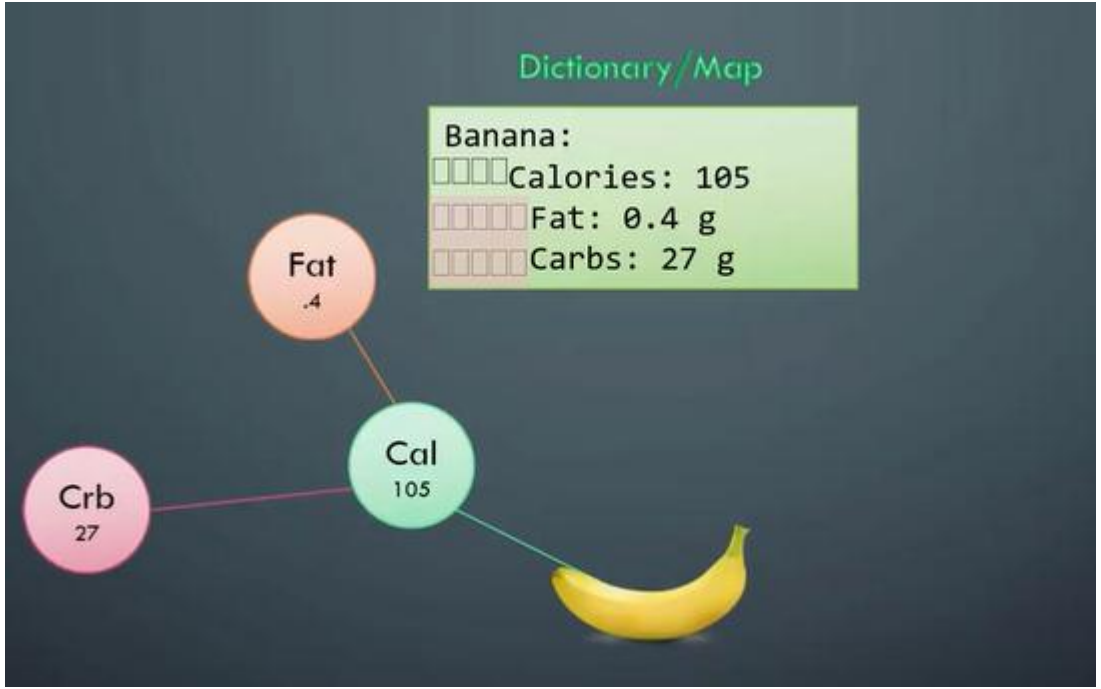
Dictionary/Map

```
Banana:
    Calories: 105
    Fat: 0.4 g
    Carbs: 27 g

Grapes:
    Calories: 62
    Fat: 0.3 g
    Carbs: 16 g
```

Yaml - Spaces


Dictionary/Map
Banana:
Calories: 105
Fat: 0.4 g
Carbs: 27 g
Cal
105
Fat
.4
Crb
27

Yaml - Spaces



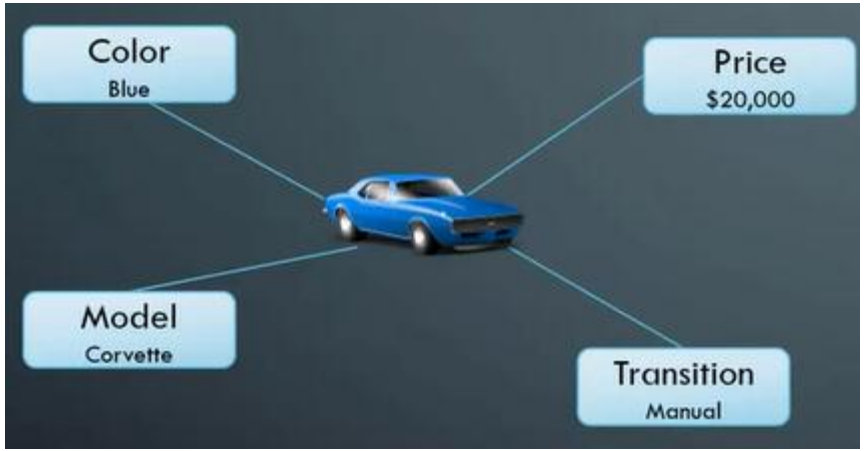
Equal no. of spaces

Yaml - Advanced

Key Value / Dictionary / Lists

```
Fruits:
    -   Banana:
            Calories: 105
            Fat: 0.4 g
            Carbs: 27 g

    -   Grape:
            Calories: 62
            Fat: 0.3 g
            Carbs: 16 g
```

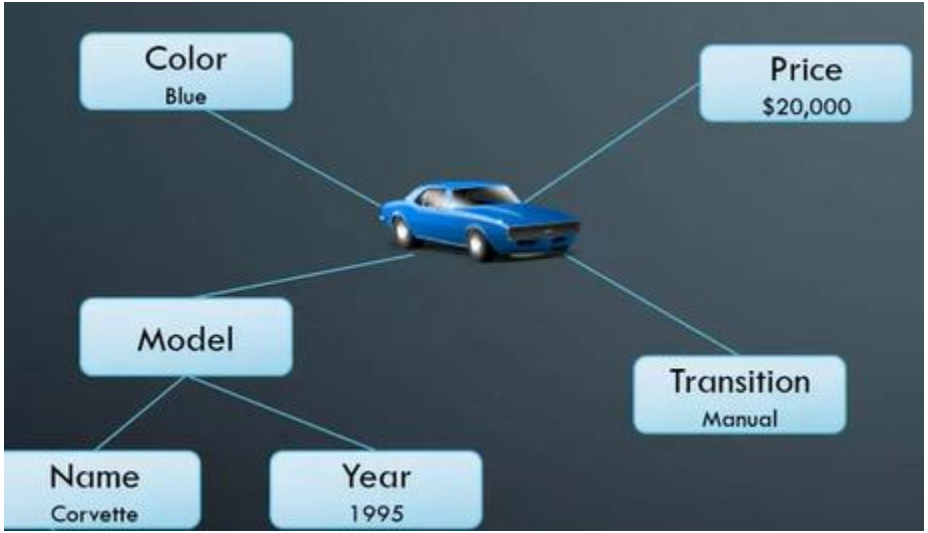
Dictionary vs List vs List of Dictionaries



Dictionary

```
Color: Blue
Model: Corvette
Transition: Manual
Price: $20,000
```

Dictionary vs List vs List of Dictionaries



Dictionary In Dictionary

```
Color: Blue
Model:
   Name: Corvette
   Year: 1995
Transition: Manual
Price: $20,000
```

Dictionary vs List vs List of Dictionaries

List Of String

```
- Blue Corvette
- Grey Corvette
- Red Corvette
- Green Corvette
- Blue Corvette
- Black Corvette
```

Dictionary vs List vs List of Dictionaries

List Of Dictionary

```
- Color: Blue
  Model:
     Name: Corvette
     Model: 1995
  Transition: Manual
  Price: $20,000
- Color: Grey
  Model:
     Name: Corvette
     Model: 1995
  Transition: Manual
  Price: $22,000
- Color: Red
  Model:
     Name: Corvette
     Model: 1995
  Transition: Automatic
  Price: $20,000
- Color: Green
  Model:
     Name: Corvette
     Model: 1995
  Transition: Manual
  Price: $23,000
- Color: Blue
  Model:
     Name: Corvette
     Model: 1995
  Transition: Manual
  Price: $20,000
```

Yaml – Notes

Dictionary/Map

```
Banana:
    Calories: 105
    Fat: 0.4 g
    Carbs: 27 g
```

=

```
Banana:
    Calories: 105
    Carbs: 27 g
    Fat: 0.4 g
```

Array/List

```
Fruits:
-   Orange
-   Apple
-   Banana
```

≠

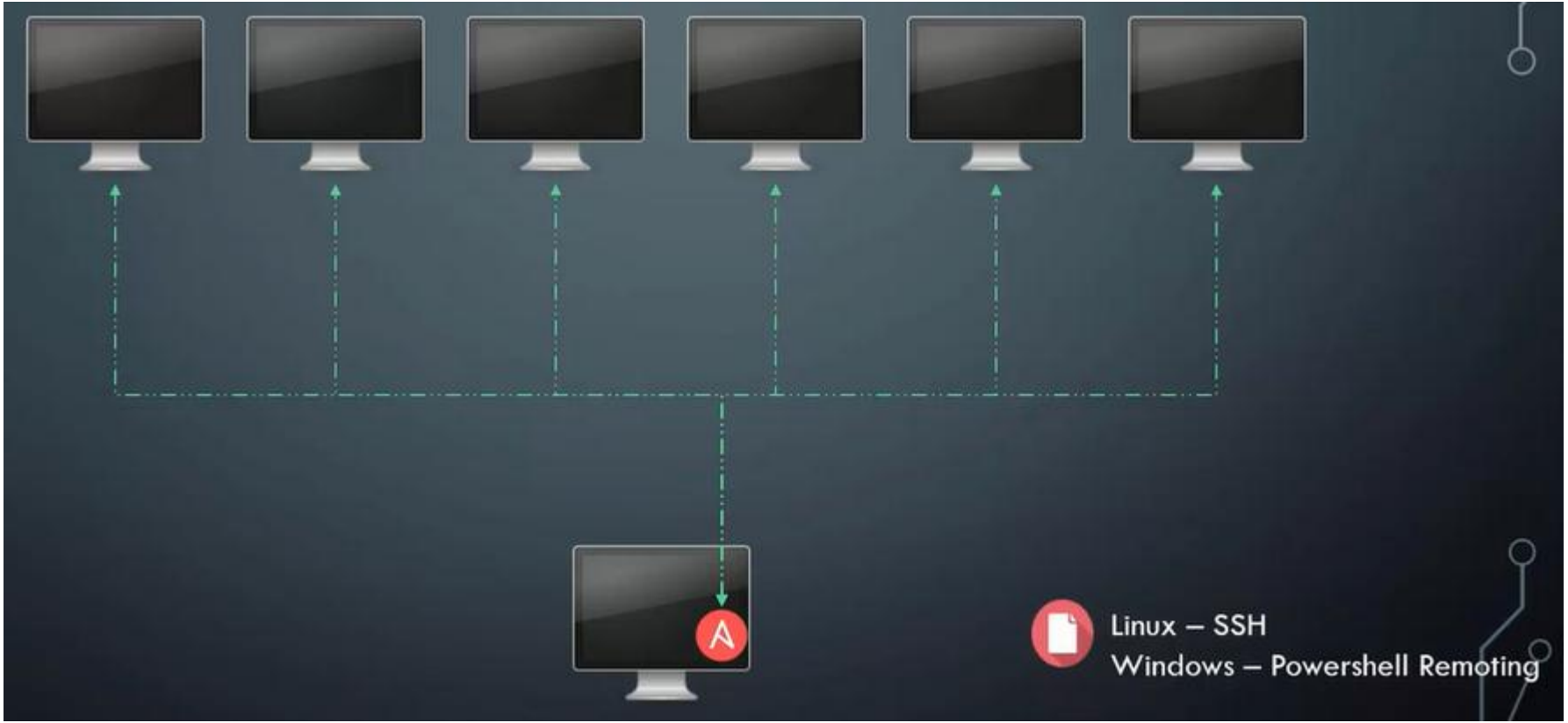
```
Fruits:
-   Orange
-   Banana
-   Apple
```

```
# List of Fruits
Fruits:
-   Orange
-   Apple
-   Banana
```

Dictionary – Unordered
List – Ordered

Hash # – Comments

Inventory- Server Info File



Inventory - Structure


```
#Sample Inventory File

Server1.company.com
Server2.company.com
```

More on Inventory

```
web     ansible_host=server1.company.com
db      ansible_host=server2.company.com
mail    ansible_host=server3.company.com
web2    ansible_host=server4.company.com
```

Inventory Parameters:

- ansible_connection – ssh/winrm/localhost
- ansible_port – 22/5986
- ansible_user – root/administrator
- ansible_ssh_pass - Password

Ansible Playbooks

We define what we want Ansible to do

Set of instruction we provide to Ansible to do its magic

Can be simple of can be complex as per requirement

```
#Simple Ansible Playbook

-  Run command1 on server1
-  Run command2 on server2
-  Run command3 on server3
-  Run command4 on server4
-  Run command5 on server5
-  Run command6 on server6
-  Run command7 on server7
-  Run command8 on server8
-  Run command9 on server9
-  Restarting Server1
-  Restarting Server2
-  Restarting Server3
-  Restarting Server4
-  Restarting Server5
-  Restarting Server6
-  Restarting Server7
```

```
#Complex Ansible Playbook

-  Deploy 50 VMs on Public Cloud
-  Deploy 50 VMs on Private Cloud
-  Provision Storage to all VMs
-  Setup Network Configuration on Private VMs
-  Setup Cluster Configuration
-  Configure Web server on 20 Public VMs
-  Configure DB server on 20 Private VMs
-  Setup Loadbalancing between web server VMs
-  Setup Monitoring components
-  Install and Configure backup clients on VMs
-  Update CMDB database with new VM Information
```

- Playbook – A single YAML file
  - Play – Defines a set of activities (tasks) to be run on hosts
    - Task – An action to be performed on the host
      - Execute a command
      - Run a script
      - Install a package
      - Shutdown/Restart

YAML format

```
#Simple Ansible Playbook1.yml
-
  name: Play 1
  hosts: localhost
  tasks:
    - name: Execute command 'date'
      command: date

    - name: Execute script on server
      script: test_script.sh

    - name: Install httpd service
      yum:
        name: httpd
        state: present

    - name: Start web server
      service:
        name: httpd
        state: started
```

Ansible Playbooks

```
#Simple Ansible Playbook1.yml
-
  name: Play 1
  hosts: localhost
  tasks:
    - name: Execute command 'date'
      command: date

    - name: Execute script on server
      script: test_script.sh

-
  name: Play 2
  hosts: localhost
  tasks:
    - name: Install web service
      yum:
        name: httpd
        state: present

    - name: Start web server
      service:
        name: httpd
        state: started
```

Ansible Playbooks - Host

```
#Simple Ansible Playbook1.yml
-
  name: Play 1
  hosts: localhost
  tasks:
     - name: Execute command 'date'
       command: date

     - name: Execute script on server
       script: test_script.sh

     - name: Install httpd service
       yum:
          name: httpd
          state: present

     - name: Start web server
       service:
          name: httpd
          state: started
```

```
#Sample Inventory File

localhost

Server1.company.com
Server2.company.com

[mail]
Server3.company.com
Server4.company.com

[db]
Server5.company.com
Server6.company.com

[web]
Server7.company.com
Server8.company.com
```

Ansible Playbooks - Module

Different action run by a task is called Module

```
#Simple Ansible Playbook1.yml
-
  name: Play 1
  hosts: localhost
  tasks:
     - name: Execute command 'date'
       command: date

     - name: Execute script on server
       script: test_script.sh

     - name: Install httpd service
       yum:
          name: httpd
          state: present

     - name: Start web server
       service:
          name: httpd
          state: started
```

Ansible Playbooks - Run

- Execute Ansible Playbook
- Syntax: ansible-playbook <playbook file name>

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

```
ansible-playbook --help
```

Modules

- System
- Commands
- Files
- Database
- Cloud
- Windows
- More..

Modules

- System
- Commands
- Files
- Database
- Cloud
- Windows
- More..

- User
- Group
- Hostname
- Iptables
- Lvg
- Lvol
- Make
- Mount
- Ping
- Timezone
- Systemd
- Service

Modules

- System
- Commands
- Files
- Database
- Cloud
- Windows
- More..

- Command
- Expect
- Raw
- Script
- Shell

Modules

- System
- Commands
- Files
- Database
- Cloud
- Windows
- More..

- Acl
- Archive
- Copy
- File
- Find
- Lineinfile
- Replace
- Stat
- Template
- Unarchive

Modules

- System
- Commands
- Files
- Database
- Cloud
- Windows
- More..

- Mongodb
- Mssql
- Mysql
- Postgresql
- Proxysql
- vertica

Modules

- System
- Commands
- Files
- Database
- Cloud
- Windows
- More..

- Amazon
- Atomic
- Azure
- Centrylink
- Cloudscale
- Cloudstack
- Digital Ocean
- Docker
- Google
- Linode
- Openstack
- Rackspace
- Smartos
- Softlayer
- VMware

Modules

- System
- Commands
- Files
- Database
- Cloud
- Windows
- More..

- Win_copy
- Win_command
- Win_domain
- Win_file
- Win_iis_website
- Win_msg
- Win_msi
- Win_package
- Win_ping
- Win_path
- Win_robocopy
- Win_regedit
- Win_shell
- Win_service
- Win_user
- And more

Modules - Command

parameter	comments
chdir	cd into this directory before running the command
creates	a filename or (since 2.0) glob pattern, when it already exists, this step will **not** be run.
executable	change the shell used to execute the command. Should be an absolute path to the executable.
free_form	the command module takes a free form command to run. There is no parameter actually named 'free form'. See the examples!
removes	a filename or (since 2.0) glob pattern, when it does not exist, this step will **not** be run.
warn (added in 1.8)	if command warnings are on in ansible.cfg, do not warn about this particular line if set to no/false.

```
#Simple Ansible Playbook1.yml
-
  name: Play 1
  hosts: localhost
  tasks:
    - name: Execute command 'date'
      command: date

    - name: Display resolv.conf contents
      command: cat /etc/resolv.conf
    - name: Display resolv.conf contents
      command: cat resolv.conf chdir=/etc

    - name: Display resolv.conf contents
      command: mkdir /folder creates=/folder
```

Modules - Service

- Manage Services – Start, Stop, Restart

```
#Sample Ansible Playbook1.yml
-
  name: Start Services in order
  hosts: localhost
  tasks:
    - name: Start the database service
      service: name=postgresql state=started

    - name: Start the httpd service
      service: name=httpd state=started
```

```
#Sample Ansible Playbook1.yml
-
  name: Start Services in order
  hosts: localhost
  tasks:
    - name: Start the database service
      service:
        name: postgresql
        state: started
```

IDEMPOTENCY

- Why “started” and not “start”

“Start” the service httpd

"Started" the service httpd

Ensure service httpd is started

If httpd is not already started => start it
If httpd is already started, =>do nothing

Idempotency
An operation is idempotent if the result of performing it once is exactly the same as the result of performing it repeatedly without any intervening actions.

Modules - LineInFile

- Search for a line in a file and replace it or add it if it doesn't exist.

```
#Sample /etc/resolv.conf

nameserver 10.1.250.1
nameserver 10.1.250.2
```

```
nameserver 10.1.250.10
```

```
#Sample Ansible Playbook1.yml
-
  name: Add DNS server to resolv.conf
  hosts: localhost
  tasks:
        - lineinfile:
            path: /etc/resolv.conf
            line: 'nameserver 10.1.250.10'
```

```
#Sample script

echo "nameserver 10.1.250.10" >> /etc/resolv.conf
```

```
#Sample /etc/resolv.conf

nameserver 10.1.250.1
nameserver 10.1.250.2
nameserver 10.1.250.10
```

```
#Sample /etc/resolv.conf

nameserver 10.1.250.1
nameserver 10.1.250.2
nameserver 10.1.250.10
nameserver 10.1.250.10
nameserver 10.1.250.10
```

Variables

- Stores information that varies with each host
- Inventory

```
#Sample Inventory File

Web1 ansible_host=server1.company.com ansible_connection=ssh ansible_shh_pass=P@ssW
db   ansible_host=server2.company.com ansible_connection=winrm ansible_shh_pass=P@s
Web2 ansible_host=server3.company.com ansible_connection=ssh ansible_shh_pass=P@ssW
```

- Playbook

```
#Sample Ansible Playbook1.yml
-
  name: Add DNS server to resolv.conf
  hosts: localhost
  vars:
      dns_server: 10.1.250.10
  tasks:
          - lineinfile:
              path: /etc/resolv.conf
              line: 'nameserver 10.1.250.10'
```

- Variable files

```
#Sample Variable_file.yml
variable1: value1
variable2: value2
```

Variables – In Use

```
#Sample Ansible Playbook1.yml
-
  name: Set Firewall Configurations
  hosts: web
  tasks:
     -  firewalld:
          service: https
          permanent: true
          state: enabled
     -  firewalld:
          port:8081/tcp
          permanent: true
          state: disabled

     -  firewalld:
          port: 161-162/udp
          permanent: true
          state: disabled

     -  firewalld:
          source: 192.0.2.0/24
          Zone: internal
          state: enabled
```

```
#Sample Inventory File

Web http_port=     snmp_port=        inter_ip_range=
```

```
#Sample Ansible Playbook1.yml
-
  name: Set Firewall Configurations
  hosts: web
  tasks:
    -  firewalld:
         service: https
         permanent: true
         state: enabled
    -  firewalld:
         port: '{{ http_port }}'/tcp
         permanent: true
         state: disabled

    -  firewalld:
         port: '{{ snmp_port }}'/udp
         permanent: true
         state: disabled

    -  firewalld:
         source: '{{ inter_ip_range }}'/24
         Zone: internal
         state: enabled
```

```
#Sample Inventory File

Web http_port=8081 snmp_port=161-162 inter_ip_range=192.0.2.0
```

```
#Sample variable File – web.yml

http_port: 8081
snmp_port: 161-162
inter_ip_range: 192.0.2.0
```

{{ }}

Jinja2 Templating

```
source: {{ inter_ip_range }}
```

```
source: '{{ inter_ip_range }}'
```

```
source: SomeThing{{ inter_ip_range }}SomeThing
```

Conditional

```
#Sample Inventory File

web1 ansible_host=web1.company.com ansible_connection=ssh ansible_shh_pass=P@ssW
db   ansible_host=db.company.com ansible_connection=winrm ansible_shh_pass=P@s
web2 ansible_host=web3.company.com ansible_connection=ssh ansible_shh_pass=P@ssW

[all_servers] # Group
web1
db
web2
```

```
#Sample Ansible Playbook1.yml
-
  name: Start services
  hosts: all_servers
  tasks:
    -    service: name=mysql state=started


    -    service: name=httpd state=started
```

Conditional

```
#Sample Inventory File

web1 ansible_host=web1.company.com ansible_connection=ssh ansible_shh_pass=P@ssW
db   ansible_host=db.company.com ansible_connection=winrm ansible_shh_pass=P@s
web2 ansible_host=web3.company.com ansible_connection=ssh ansible_shh_pass=P@ssW

[all_servers] # Group
web1
db
web2
```

```
#Sample Inventory File
…
…

[db_servers] # Group
db

[web_servers] # Group
web1
web2
```

```
#Sample Ansible Playbook1.yml
-
  name: Start services
  hosts: all_servers
  tasks:
     -    service: name=mysql state=started
          when: ansible_host == “db.company.com”

     -    service: name=httpd state=started
          when: ansible_connection == “web1.company.com” or
                ansible_connection == “web2.company.com”
```

```
#Sample Ansible Playbook1.yml
-
  name: Start db services
  hosts: db_servers
  tasks:
     -    service: name=mysql state=started

-
  name: Start web services
  hosts: web_servers
  tasks:
     -    service: name=httpd state=started
```

Register And When

```
#Sample Ansible Playbook1.yml
-
  name: Check status of service and email if its down
  hosts: localhost
  tasks:
     -    command: service httpd status



     -    mail:
              to: Admins <system.admins@company.com>
              subject: Service Alert
              body: 'Service {{ ansible_hostname }} is down.'
          when:
```

Register And When

```
#Sample Ansible Playbook1.yml
-
  name: Check status of service and email if its down
  hosts: localhost
  tasks:
     -    command: service httpd status
          register: command_output


     -    mail:
               to: Admins <system.admins@company.com>
               subject: Service Alert
               body: 'Service {{ ansible_hostname }} is down.'
          when: command_output.stdout.find('down') != -1
```

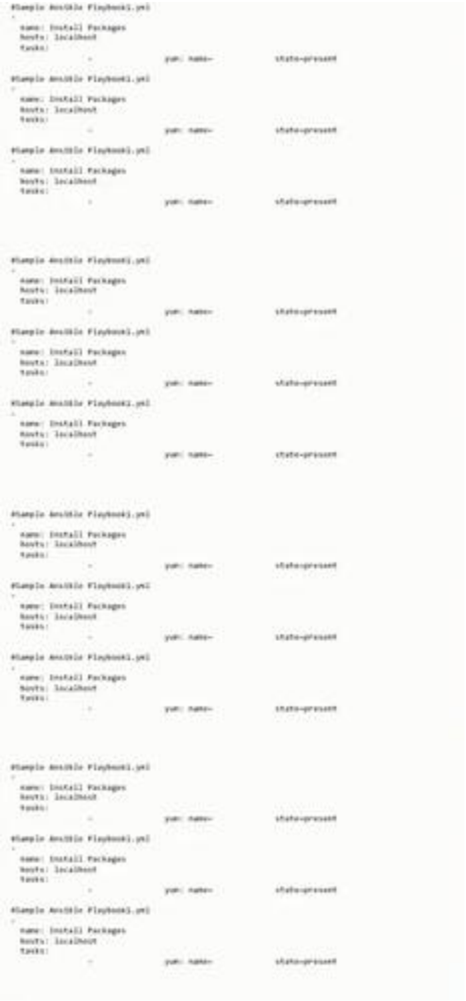
Loops

```
#Sample Ansible Playbook1.yml
-
  name: Install Packages
  hosts: localhost
  tasks:
     -    yum: name=httpd        state=present
     -    yum: name=binutils state=present
     -    yum: name=glibc state=present
     -    yum: name=ksh state=present
     -    yum: name=libaio state=present
     -    yum: name=libXext state=present
     -    yum: name=gcc state=present
     -    yum: name=make state=present
     -    yum: name=sysstat state=present
     -    yum: name=unixODBC state=present
     -    yum: name=mongodb state=present
     -    yum: name=nodejs state=present
     -    yum: name=grunt state=present
```

Loops

```
#Sample Ansible Playbook1.yml
-
  name: Install Packages
  hosts: localhost
  tasks:
     -    yum: name='{{ item }}' state=present
          with_items:
               -    httpd
               -    binutils
               -    glibc
               -    ksh
               -    libaio
               -    libXext
               -    gcc
               -    make
               -    sysstat
               -    unixODBC
               -    mongodb
               -    nodejs
               -    grunt
```

Ansible - Roles



Packages

Modules

```
#Sample Ansible Playbook1.yml
-
  name: Install Packages
  hosts: localhost
  tasks:
               -               yum: name=           state=present
#Sample Ansible Playbook1.yml
-
  name: Install Packages
  hosts: localhost
  tasks:
               -               yum: name=           state=present
#Sample Ansible Playbook1.yml
-
  name: Install Packages
  hosts: localhost
  tasks:
               -               yum: name=           state=present
```

```
#Sample Ansible Playbook1.yml
-
  name: Install Packages
  hosts: localhost
  tasks:
               -               yum: name=           state=present
#Sample Ansible Playbook1.yml
-
  name: Install Packages
  hosts: localhost
  tasks:
               -               yum: name=           state=present
#Sample Ansible Playbook1.yml
-
  name: Install Packages
  hosts: localhost
  tasks:
               -               yum: name=           state=present
```

```
#Sample Ansible Playbook1.yml
-
  name: Install Packages
  hosts: localhost
  tasks:
               -               yum: name=           state=present
#Sample Ansible Playbook1.yml
-
  name: Install Packages
  hosts: localhost
  tasks:
               -               yum: name=           state=present
#Sample Ansible Playbook1.yml
-
  name: Install Packages
  hosts: localhost
  tasks:
               -               yum: name=           state=present
```

```
#Sample Ansible Playbook1.yml
-
  name: Install Packages
  hosts: localhost
  tasks:
               -               yum: name=           state=present
#Sample Ansible Playbook1.yml
-
  name: Install Packages
  hosts: localhost
  tasks:
               -               yum: name=           state=present
#Sample Ansible Playbook1.yml
-
  name: Install Packages
  hosts: localhost
  tasks:
               -               yum: name=           state=present
```

Packages

Modules

Classes

Functions

Include

setup_application.yml (1500 Lines)

```
# Provision VMs
-
  name: Provision VMs on Cloud
  hosts: localhost
  tasks:
        -        vmware:
-----------------------------------------
-----------------------------------------
-----------------------------------------
-----------------------------------------

# Install Required Dependencies
-
  name: Install Required dependencies
  hosts: localhost
  tasks:
        -        yum:
-----------------------------------------
-----------------------------------------
-----------------------------------------
-----------------------------------------

# Configure Web Server
-
  name: Configure Web Server
  hosts: localhost
  tasks:
        -        apache:
-----------------------------------------
-----------------------------------------
-----------------------------------------
-----------------------------------------

# Setup and Start Application
-
  name: Setup and Start Application
  hosts: localhost
  tasks:
        -        service:
-----------------------------------------
-----------------------------------------
-----------------------------------------
-----------------------------------------
```

provision_vms.yml (500 Lines)

```
# Provision VMs
-
  name: Provision VMs on Cloud
  hosts: localhost
  tasks:
        -        vmware:
-----------------------------------------
-----------------------------------------
-----------------------------------------
-----------------------------------------
```

install_dependencies.yml (500 Lines)

```
# Install Required Dependencies
-
  name: Install Required dependencies
  hosts: localhost
  tasks:
        -        yum:
-----------------------------------------
-----------------------------------------
-----------------------------------------
-----------------------------------------
```

configure_web_server.yml (500 Lines)

```
# Configure Web Server
-
  name: Configure Web Server
  hosts: localhost
  tasks:
        -        apache:
-----------------------------------------
-----------------------------------------
-----------------------------------------
-----------------------------------------
```

setup_start_application.yml (500 Lines)

```
# Setup and Start Application
-
  name: Setup and Start Application
  hosts: localhost
  tasks:
        -        service:
-----------------------------------------
-----------------------------------------
-----------------------------------------
```

setup_application.yml (5 Lines)

```
# Setup Application
- include provision_vms.yml
- include install_dependencies.yml
- include configure_web_server.yml
- include setup_start_application.yml
```

```
- include <playbook name>
```

Include Task And Vars

```
#Sample Ansible Playbook1.yml
-
  name: Set Firewall Configurations
  hosts: web
  vars
      http_port: 8081
      snmp_port: 160-161
      inter_ip_range: 192.0.2.0
  tasks:
   -  firewalld:
        service: https
        permanent: true
        state: enabled
   -  firewalld:
        port: '{{ http_port }}'/tcp
        permanent: true
        state: disabled

   -  firewalld:
        port: '{{ snmp_port }}'/udp
        permanent: true
        state: disabled

   -  firewalld:
        source: '{{ inter_ip_range }}'/24
        Zone: internal
        state: enabled
```

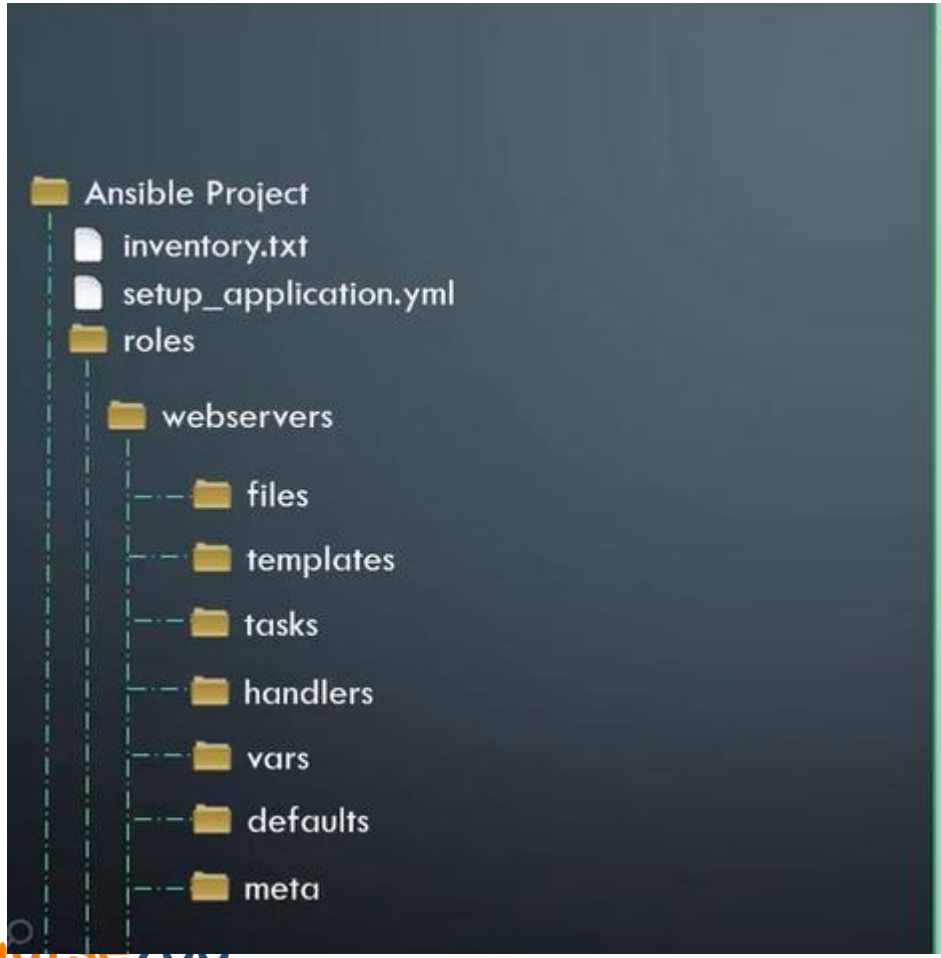
Include Task And Vars

```
#Sample Ansible Playbook1.yml
-
  name: Set Firewall Configurations
  hosts: web
  vars_files:
          - variables.yml


  tasks:
   - include: tasks.yml
```

```
    # Sample Tasks File tasks.yml
  - firewalld:
      service: https
      permanent: true
      state: enabled
  - firewalld:
      port: '{{ http_port }}'/tcp
      permanent: true
      state: disabled

  - firewalld:
      port: '{{ snmp_port }}'/udp
      permanent: true
      state: disabled

  - firewalld:
      source: '{{ inter_ip_range }}'/24
      Zone: internal
      state: enabled
```

```
    # Sample Vars File variables.yml


http_port: 8081
snmp_port: 160-161
inter_ip_range: 192.0.2.0
```

Roles – Organizing the files




```yaml
#Sample Ansible setup_application.yml
-
  name: Set Firewall Configurations
  hosts: web
  vars:
      http_port: 8081
      snmp_port: 160-161
      inter_ip_range: 192.0.2.0

  tasks:
  -  firewalld:
        service: https
        permanent: true
        state: enabled
  -  firewalld:
        port: '{{ http_port }}'/tcp
        permanent: true
        state: disabled

  -  firewalld:
        port: '{{ snmp_port }}'/udp
        permanent: true
        state: disabled

  -  firewalld:
        source: '{{ inter_ip_range }}'/24
        Zone: internal
        state: enabled
```

Roles – Organizing the files

- Ansible Project
  - inventory.txt
  - setup_application.yml
  - roles
    - webservers
      - files
      - templates
      - tasks
      - handlers
      - vars
      - defaults
      - meta

```
tasks:
-  firewalld:
      service: https
      permanent: true
      state: enabled
-  firewalld:
      port: '{{ http_port }}'/tcp
      permanent: true
      state: disabled

-  firewalld:
      port: '{{ snmp_port }}'/udp
      permanent: true
      state: disabled

-  firewalld:
      source: '{{ inter_ip_range }}'/24
      Zone: internal
      state: enabled
```

```
#Sample Ansible setup_application.yml
-
  name: Set Firewall Configurations
  hosts: web
```

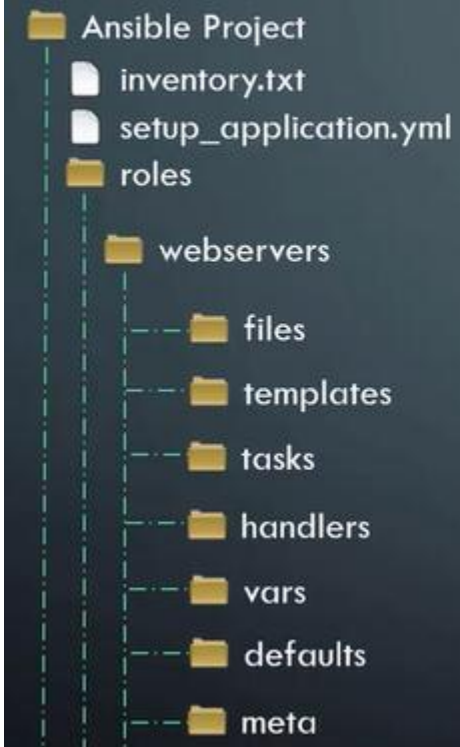
Roles – Organizing the files

- Ansible Project
  - inventory.txt
  - setup_application.yml
  - roles
    - webservers
      - files
      - templates
      - tasks
      - handlers
      - vars
      - defaults
      - meta

```
tasks:
-  firewalld:
      service: https
      permanent: true
      state: enabled
-  firewalld:
      port: '{{ http_port }}'/tcp
      permanent: true
      state: disabled

-  firewalld:
      port: '{{ snmp_port }}'/udp
      permanent: true
      state: disabled

-  firewalld:
      source: '{{ inter_ip_range }}'/24
      Zone: internal
      state: enabled
```

```
#Sample Ansible setup_application.yml
-
  name: Set Firewall Configurations
  hosts: web
```

Roles – Organizing the files




```
#Sample Ansible setup_application.yml
-
  name: Set Firewall Configurations
  hosts: web
  roles:
      - webservers
```

Temples - filters

The name is {{ my_name }} => The name is Bond

The name is {{ my_name | upper }} => The name is BOND

The name is {{ my_name | lower }} => The name is bond

The name is {{ my_name | title }} => The name is Bond

The name is {{ my_name | replace ("Bond", "Bourne") }} => The name is Bourne

The name is {{ first_name | default("James") }} {{ my_name }} => The name is James Bond

- Substitute
- Upper
- Lower
- Title
- replace
- default

Jinja2 – filters List and Set

{{ [1, 2, 3] | min }}	=> 1
{{ [1, 2, 3] | max }}	=> 3
{{ [1, 2, 3, 2] | unique }}	=> 1, 2, 3
{{ [1, 2, 3, 4] | union([4, 5]) }}	=> 1, 2, 3, 4, 5
{{ [1, 2, 3, 4] | intersect([4, 5]) }}	=> 4
{{ 100 | random }}	=> Random number

- min
- max
- unique
- union
- intersect
- random

Jinja2 – filters For File

{{ "/etc/hosts" | basename }}	=> hosts
{{ "c:\windows\hosts" | win_basename }}	=> hosts
{{ "c:\windows\hosts" | win_splitdrive }}	=> ["c:", "\windows\hosts"]
{{ "c:\windows\hosts" | win_splitdrive | first }}	=> "c:"
{{ "c:\windows\hosts" | win_splitdrive | last }}	=> "\windows\hosts"

Ansible Vault

- A feature of ansible that allows us to keep sensitive data
- Such as passwords or keys in encrypted files, rather than as plaintext
- Ansible Vault can encrypt any structured data file used by Ansible
- Reduces the pressure on the network

Ansible Vault

```
# Inventory File - inventory.txt

db_server  ansible_ssh_pass=Passw0rd ansible_host=192.168.1.1
web_server ansible_ssh_pass=Passw0rd ansible_host=192.168.1.2
```

```
ansible-vault encrypt inventory.txt
```


```
$ANSIBLE_VAULT;1.1;AES256
6138346438393963323838323935623966643231356533346363643532646236386332326363626 1
6432623864313032636434613931316262646534633165340a323664333661323961666361326430
6263656233373863663837663132623364613038613364643863373962336264623862643835626 5
6534663335386138370a6231336533393561386238313066383838383638393038663030316430 38
3337306165386330366438393531666262306531613734336131343531376130333263363733393 2
6462336262356539666539323735643065396661639643666393832346333636632663136306633
6134386537636264316635646665383661393766623662623564613063323839336139663361316 2
6563303338666338363832326564636536346536653331316131316632313363383030626366303 9
6663323963383236633933613733656464643434383132313432303735626538643164323334663 1
6263613365353039386666663864313363656436653036666363356538636336623632376336383 7
3638356566383562396664373966623762626435333336346434666533373132326562353035373 6
6234326638613833656335616433303061623830613266653762396339336136333631313863323 8
6137
```

Ansible Vault

```
ansible-playbook playbook.yml -i inventory.txt
```

```
ERROR! Attempted to read "inventory.txt" as ini file: Decryption failed on inventory.txt
```

```
ansible-playbook playbook.yml -i inventory.txt -ask-vault-pass
```

```
root@controller:/opt/first_project # ansible-playbook /tmp/temp_playbook.yml -i inventory.txt --ask-vault-pass
Vault password:

PLAY [Test Template playbook] ***********************************************************************

TASK [Gathering Facts] ******************************************************************************
ok: [target1]
```

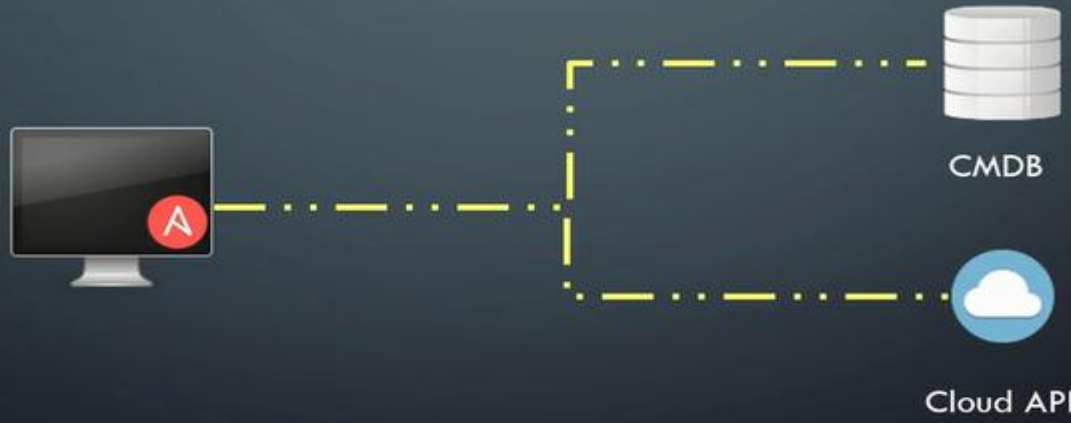
```
ansible-playbook playbook.yml -i inventory.txt -vault-password-file ~./vault_pass.txt
```

```
ansible-playbook playbook.yml -i inventory.txt -vault-password-file ~./vault_pass.py
```

Dynamic Inventory

- Inventory information that Ansible retrieves programmatically when ansible playbook is run as oppose to us defining in static text file

```
# Inventory File - inventory.txt

db_server    ansible_host=192.168.1.1
web_server   ansible_host=192.168.1.2
```



Dynamic Inventory

```
# Inventory File - inventory.txt

db_server    ansible_host=192.168.1.1
web_server   ansible_host=192.168.1.2
```

```
ansible-playbook playbook.yml -i inventory.txt
```

```
ansible-playbook playbook.yml -i inventory.py
```

```python
#!/usr/bin/env python

import json

# Get inventory data from source - CMDB or any other API
def get_inventory_data():
    return {
        "databases": {
            "hosts": ["db_server"],
            "vars": {
                "ansible_ssh_pass": "Passw0rd",
                "ansible_ssh_host": "192.168.1.1"
            }
        },
        "web": {
            "hosts": ["web_server"],
            "vars": {
                "ansible_ssh_pass": "Passw0rd",
                "ansible_ssh_host": "192.168.1.2"
            }
        }
    }


# Default main function
if __name__ == "__main__":
    inventory_data = get_inventory_data()
    print(json.dumps(inventory_data))
```

Dynamic Inventory

What environment to run in?

Retrieve Inventory Information

Main function to print data

```python
#!/usr/bin/env python

import json

# Get inventory data from source - CMDB or any other API
def get_inventory_data():
    return {
        "databases": {
            "hosts": ["db_server"],
            "vars": {
                "ansible_ssh_pass": "Passw0rd",
                "ansible_ssh_host": "192.168.1.1"
            }
        },
        "web": {
            "hosts": ["web_server"],
            "vars": {
                "ansible_ssh_pass": "Passw0rd",
                "ansible_ssh_host": "192.168.1.2"
            }
        }
    }


# Default main function
if __name__ == "__main__":
    inventory_data = get_inventory_data()
    print(json.dumps(inventory_data))
```

Custom Modules

```
#Sample Ansible Playbook1.yml
-
  name: Debug Something
  hosts: target1
  tasks:
     -  debug: msg='This is test message'
```

```
TASK [debug] ****************************************
ok: [target1] => {
    "msg": ""This is test message""
}
```

```
#Sample Ansible Playbook1.yml
-
  name: Debug Something
  hosts: target1
  tasks:
     -  custom_debug: msg='This is test message'
```

```
TASK [debug] ****************************************************
ok: [target1] => {
    "msg": ""Sat 29 Jul 17:13:33 BST 2017 - This is test message""
}
```

CODE

Import JSON

Import AnsibleModule

Instantiate AnsibleModule

```
#Sample Ansible Playbook1.yml
-
  name: Debug Something
  hosts: target1
  tasks:
     -  custom_debug:
          msg: 'This is test message'
```

```python
#!/usr/bin/python

try:
    import json
except ImportError:
    import simplejson as json

from ansible.module_utils.basic import AnsibleModule
import time
import sys

def main():
    module = AnsibleModule(
        argument_spec = dict(
            msg=dict(required=True,  type='str')
        )
    )

    msg = module.params['msg']

    # Successfull Exit
    try:
        print(json.dumps({
            msg: '%s - %s' % (time.strftime("%c"), msg),
            "changed": True
        }))
        sys.exit(0)
    except:
    # Fail Exit
        print(json.dumps({
            "failed": True,
            "msg": "failed debugging"
        }))


if __name__ == '__main__':
    main()
```