KodeKloud

Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Ansible
For the
Absolute Beginner

The Curriculum

Red Hat Ansible for Beginners

- Ansible Introduction for Beginners
- Understanding YAML
- Ansible Inventory
- Inventory Formats
- Grouping and Parent-Child Relationships
- Ansible Variables and Facts
- Ansible Playbooks
- Ansible Modules and Plugins
- Ansible Handles
- Ansible Templates
- Ansible Roles and Collections

Ansible

Introduction

Why Ansible?

Provisioning

Configuration Management

Continuous Delivery

Application Deployment

Security Compliance

Scripts

- Time
- Coding Skills
- Maintenanc e

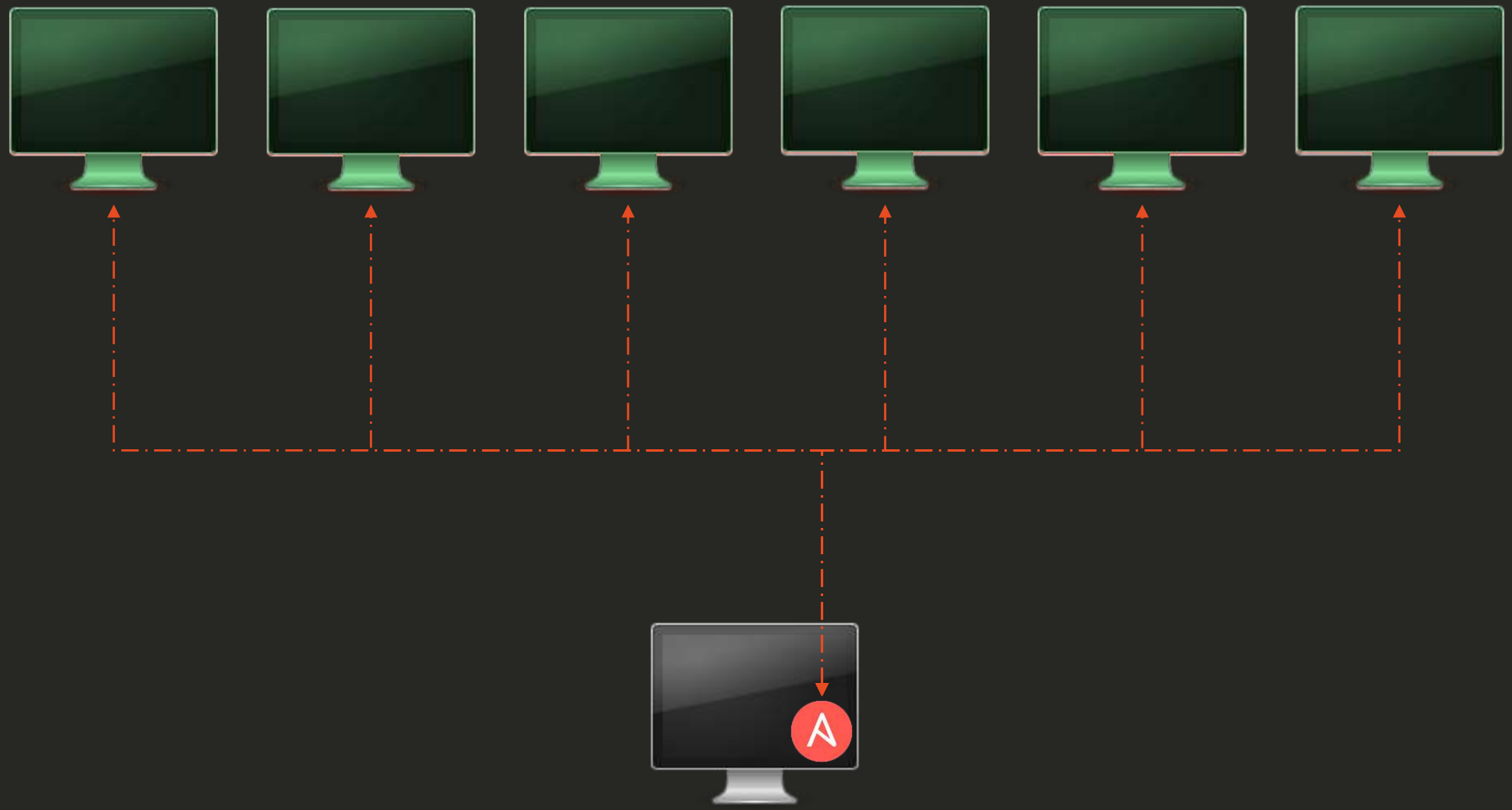
- Simple
- Powerful
- Agentless

Scripts vs Ansible Playbook

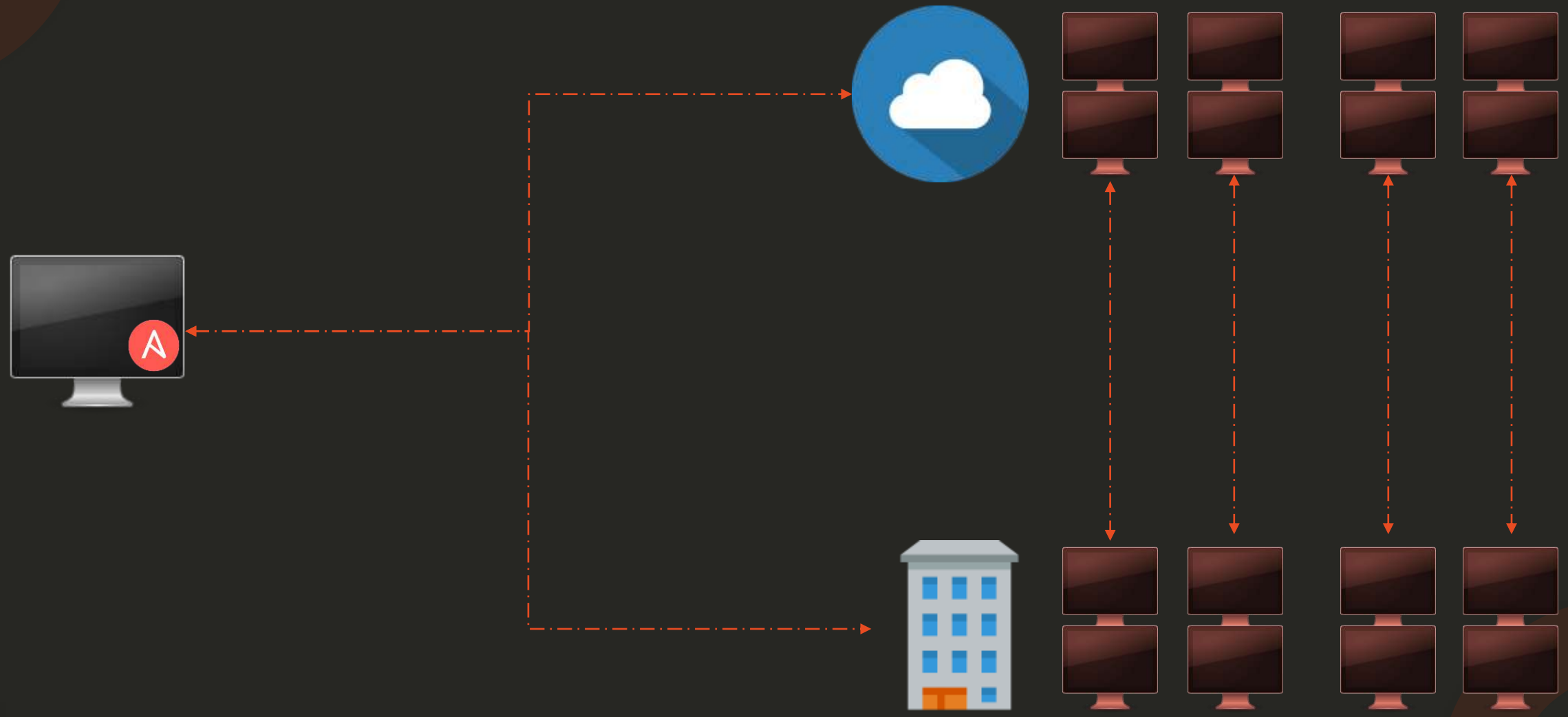
```
#!/bin/bash
# Script to add a user to Linux system
if [ $(id -u) -eq 0 ]; then
   $username=johndoe
   read -s -p "Enter password : " password
   egrep "^$username" /etc/passwd >/dev/null
   if [ $? -eq 0 ]; then
      echo "$username exists!"
      exit 1
   else
      useradd -m -p $password $username
      [ $? -eq 0 ] && echo "User has been added
to system!" || echo "Failed to add a user!"
   fi
fi
```

```
- hosts: all_my_web_servers_in_DR
  tasks:
    - user:
        name: johndoe
```

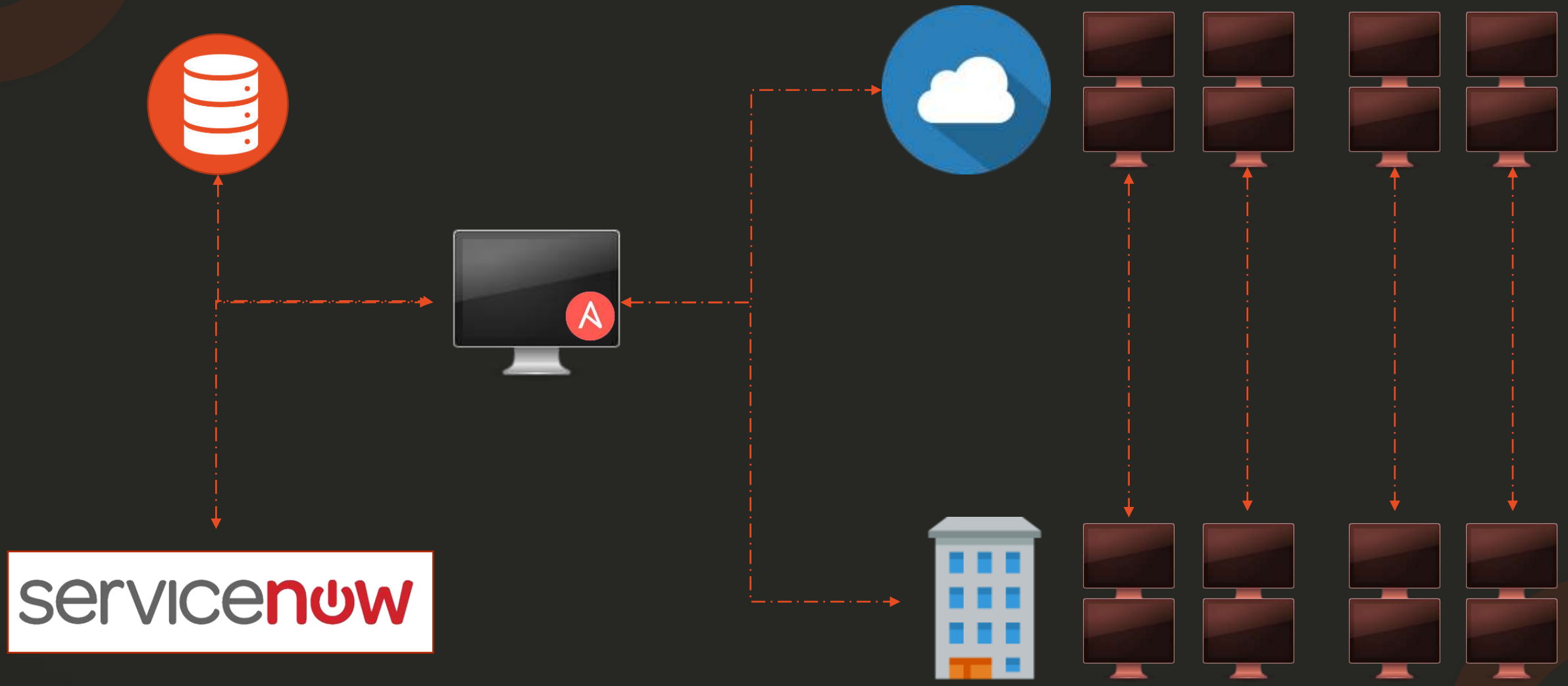
Use case example - Simple



Use case example - complex



Use case example - complex



KodeKloud

Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Ansible Configuration Files

Ansible Configuration Files

/etc/ansible/ansible.cfg

```
[defaults]

[inventory]

[privilege_escalation]

[paramiko_connection]

[ssh_connection]

[persistent_connection]

[colors]
```

Ansible Configuration Files

/etc/ansible/ansible.cfg

```
[defaults]

inventory              = /etc/ansible/hosts
log_path               = /var/log/ansible.log

library                = /usr/share/my_modules/
roles_path             = /etc/ansible/roles
action_plugins         = /usr/share/ansible/plugins/action

gathering              = implicit

# SSH timeout
timeout                = 10
forks                  = 5




[inventory]

enable_plugins         = host_list, virtualbox, yaml, constructed
```

Ansible Configuration Files

/etc/ansible/ansible.cfg

/opt/web-playbooks

/opt/web-playbooks/ansible.cfg

/opt/db-playbooks

/opt/db-playbooks/ansible.cfg

/opt/network-playbooks

/opt/network-playbooks/ansible.cfg

Ansible Configuration Files

/etc/ansible/ansible.cfg

/opt/ansible-web.cfg

/opt/web-playbooks

/opt/db-playbooks

/opt/db-playbooks/ansible.cfg

/opt/network-playbooks

/opt/network-playbooks/ansible.cfg

```
$ANSIBLE_CONFIG=/opt/ansible-web.cfg ansible-playbook playbook.yml
```

Ansible Configuration Files

1. /opt/ansible-web.cfg

/opt/web-playbooks

2. /opt/web-playbooks/ansible.cfg

4. /etc/ansible/ansible.cfg

/opt/db-playbooks

/opt/db-playbooks/ansible.cfg

3. .ansible.cfg

/opt/network-playbooks

/opt/network-playbooks/ansible.cfg

1.
```
$ANSIBLE_CONFIG=/opt/ansible-web.cfg
```

Ansible Configuration Files

/etc/ansible/ansible.cfg

/opt/web-playbooks

/opt/db-playbooks

/opt/network-playbooks

/opt/storage-playbooks

/etc/ansible/ansible.cfg

```
gathering                = implicit
```

```
ANSIBLE_   GATHERING=explicit
```

Ansible Configuration Variables

```
$  ANSIBLE_GATHERING=explicit        e-playbook playbook.yml
```

```
$ export ANSIBLE_GATHERING=explicit
$ ansible-playbook playbook.yml
```

```
/opt/web-playbooks/ansible.cfg
gathering              = explicit
```

View Configuration

```
$ ansible-config list          # Lists all configurations
```

```
$ ansible-config view          # Shows the current config file
```

```
$ ansible-config dump          # Shows the current settings
```

```
$ export ANSIBLE_GATHERING=explicit
$ ansible-config dump | grep GATHERING
  DEFAULT_GATHERING(env: ANSIBLE_GATHERING) = explicit
```

KodeKloud

Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

WHAT IS 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
```

WHAT IS 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 - NOTES

Dictionary/Map

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

=

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

Dictionary – Unordered
List – Ordered

Array/List

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

≠

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

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

Hash # – Comments

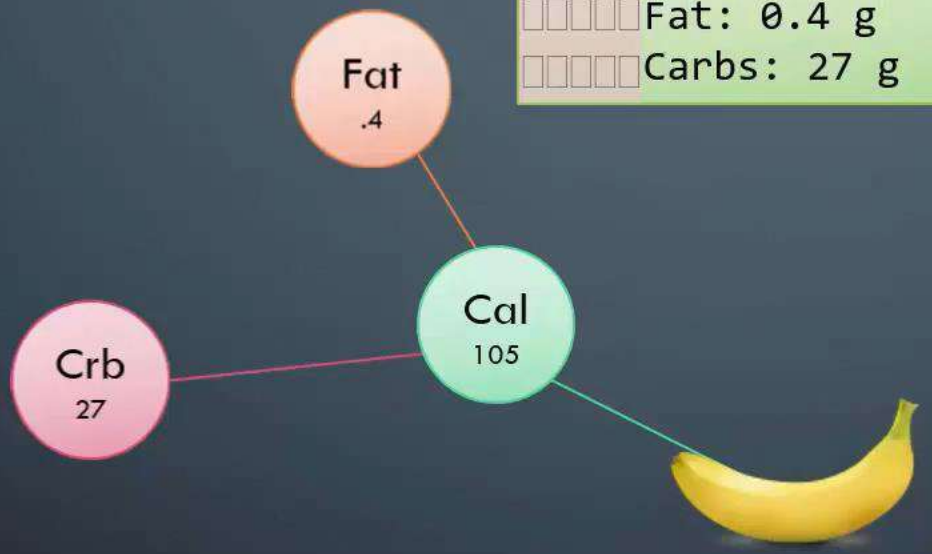
SPACES

Press Esc to exit full screen

Dictionary/Map

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

Equal number of spaces

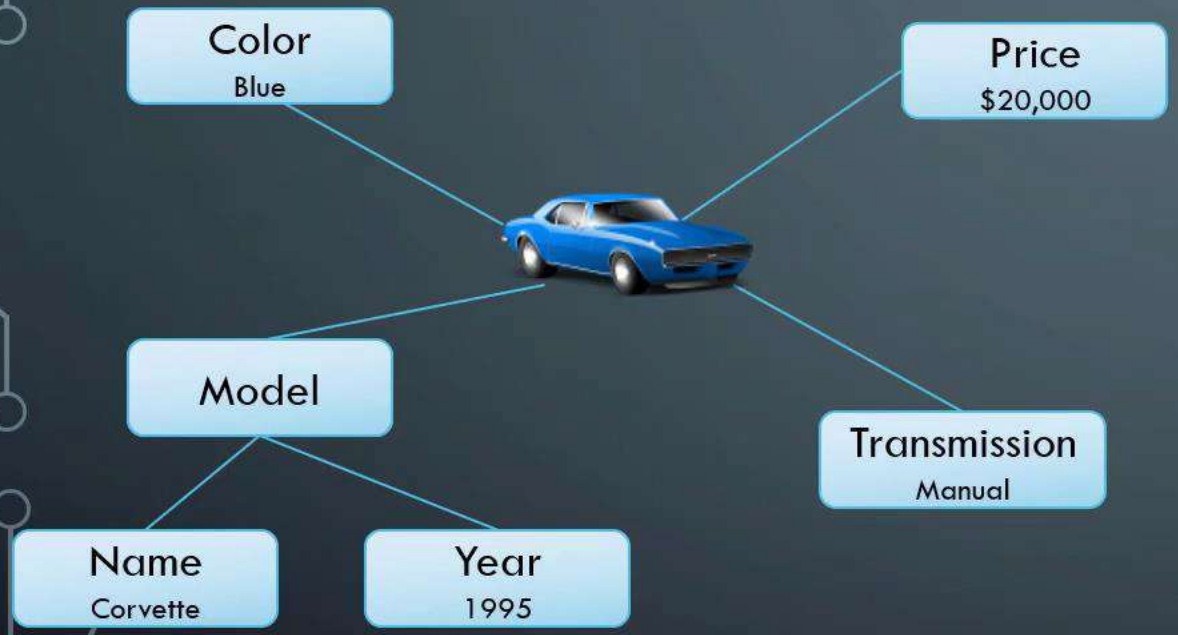


YAML - ADVANCED

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 In Dictionary

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

DICTIONARY vs LIST vs LIST OF DICTIONARIES

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

```
Color: Grey
Model:
        Name: Corvette
        Model: 1995
Transmission: Manual
Price: $22,000
```

```
Color: Red
Model:
        Name: Corvette
        Model: 1995
Transmission: Automatic
Price: $20,000
```

```
Color: Green
Model:
        Name: Corvette
        Model: 1995
Transmission: Manual
Price: $23,000
```

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

```
Color: Black
Model:
        Name: Corvette
        Model: 1995
Transmission: Automatic
Price: $25,000
```

List Of Dictionaries

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

Dictionary – Unordered
List – Ordered

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

Hash # – Comments

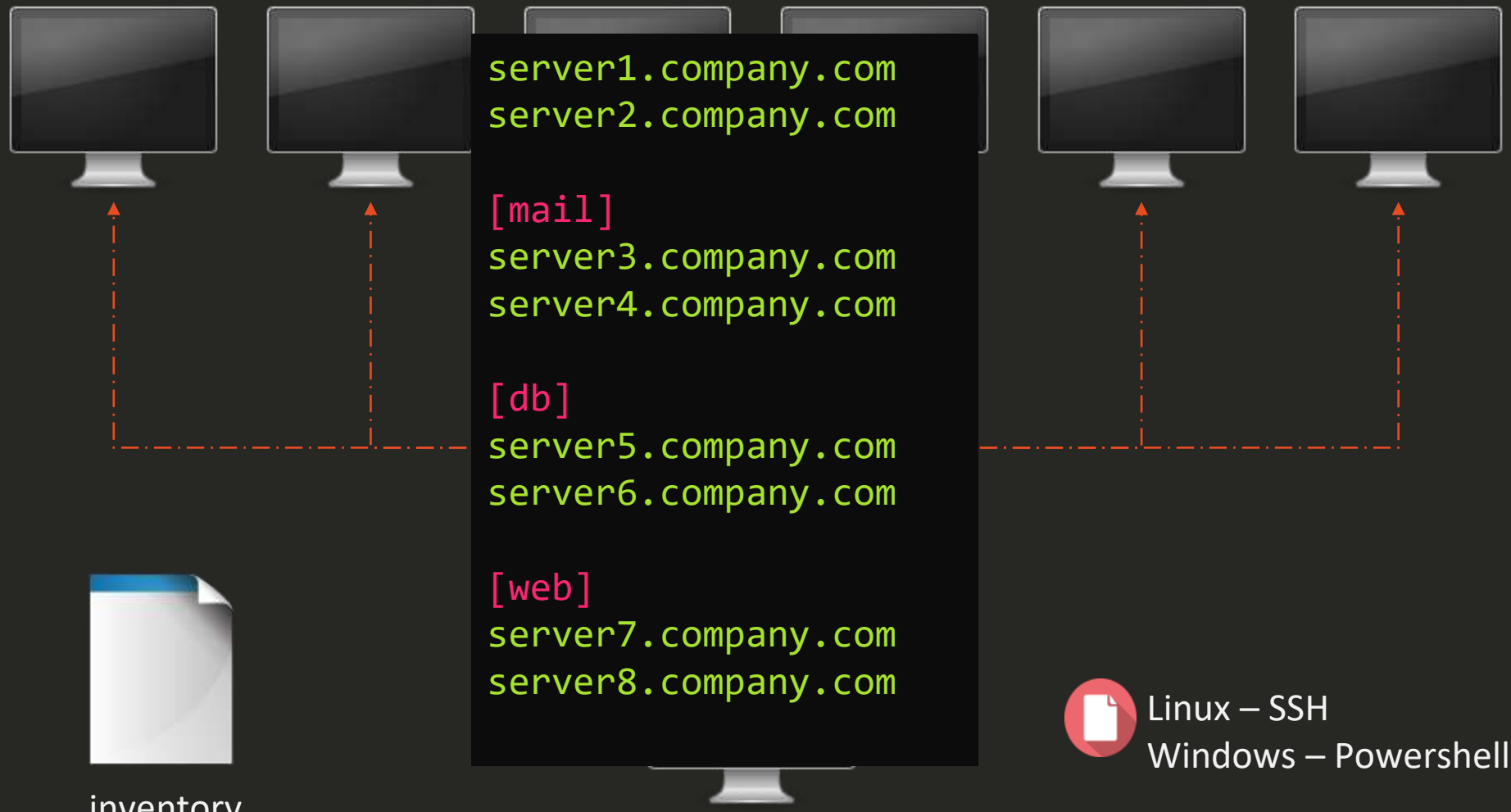
KodeKloud

Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Ansible

Inventory

inventory



inventory
/etc/ansible/hosts

Linux – SSH
Windows – Powershell Remoting

Agentless

More on inventory files

```
#Sample Inventory File

server1.company.com ansible_connection=ssh    ansible_user=root
server2.company.com ansible_connection=winrm  ansible_user=admin
server3.company.com ansible_connection=ssh    ansible_ssh_pass=P@#
server4.company.com ansible_connection=winrm

localhost ansible_connection=localhost
```

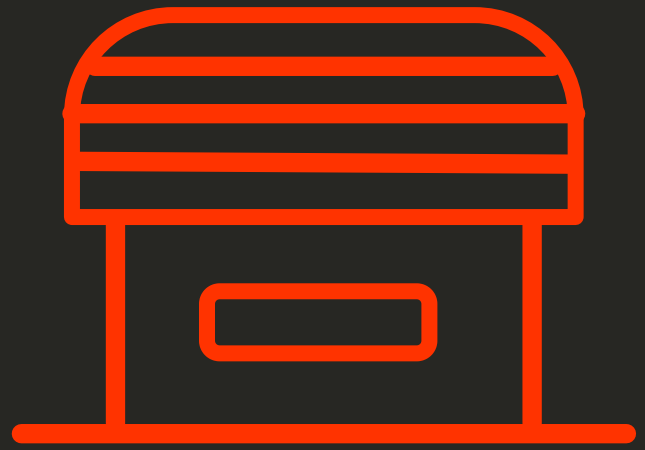
Inventory Parameters:

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

Security: Ansible Vault

Inventory Formats

Why Do We Need Different Inventory Formats?

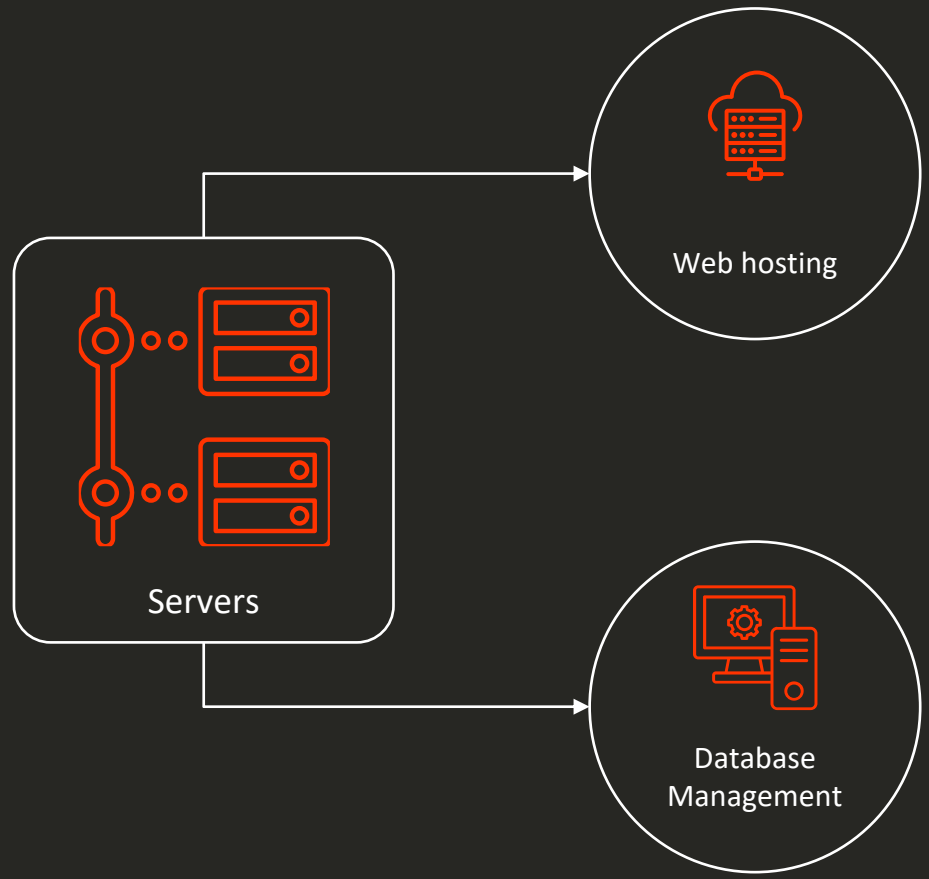


Start-up

Multinational Corporation

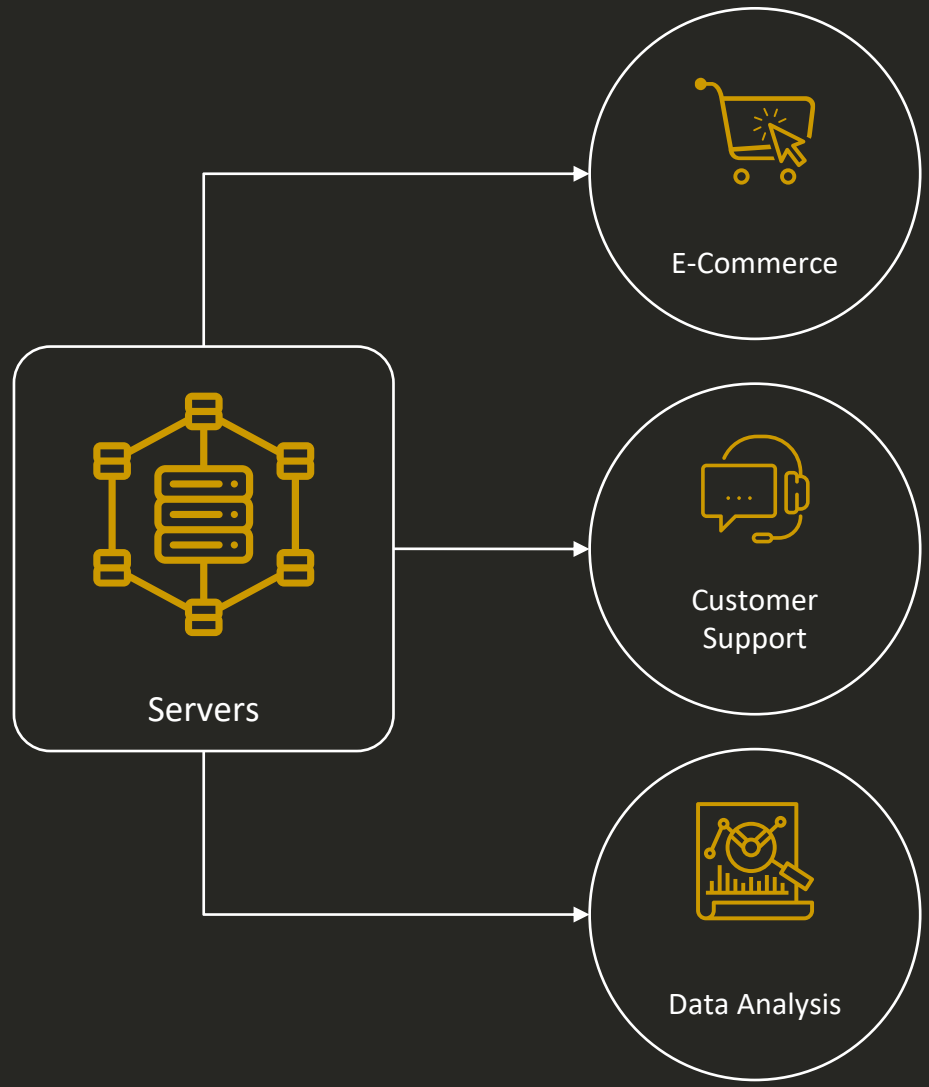
Why Do We Need Different Inventory Formats?

Start-up



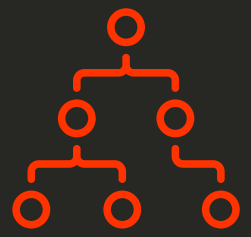
Why Do We Need Different Inventory Formats?

Multinational Corporation

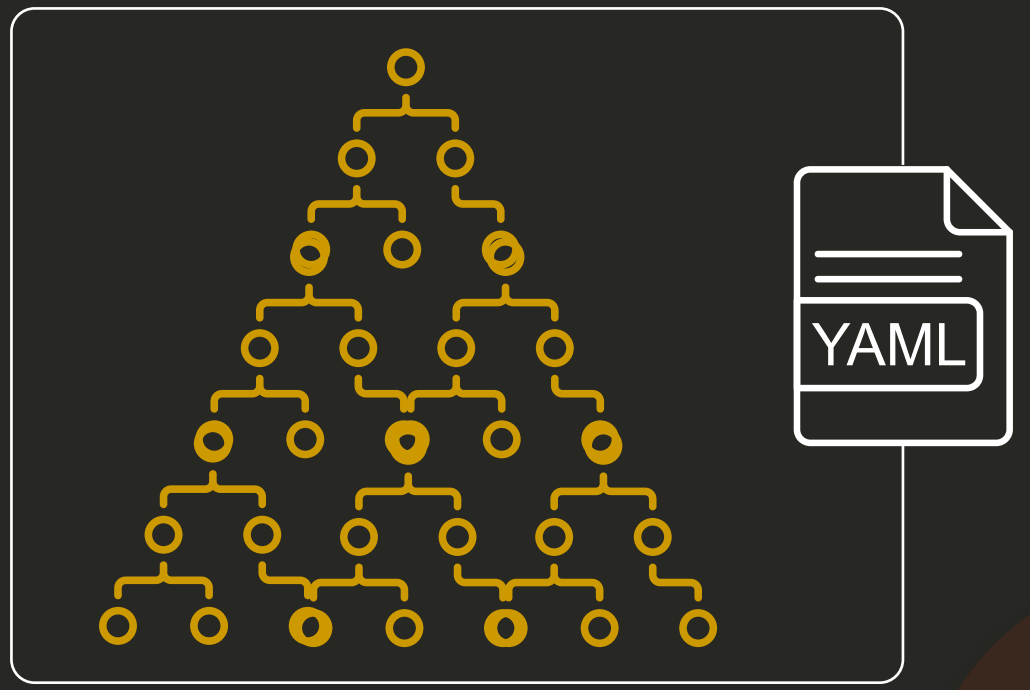


Why Do We Need Different Inventory Formats?

For Small Startup



For Multinational Corporation



Why Do We Need Different Inventory Formats?

Flexibility

Geographical Location

Different Inventory Format Types

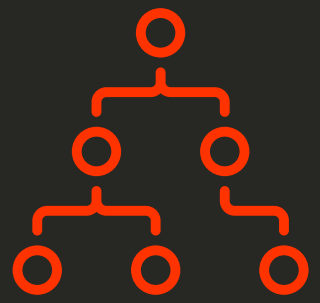
INI

YAML

Different Inventory Format Types

INI

YAML



The INI format is the simplest and most straightforward.

```
[webservers]
web1.example.com
Web2.example.com


[dbservers]
db1.example.com
db2.example.com
```

Different Inventory Format Types

INI

YAML

The YAML format is more structured and flexible than the INI format.

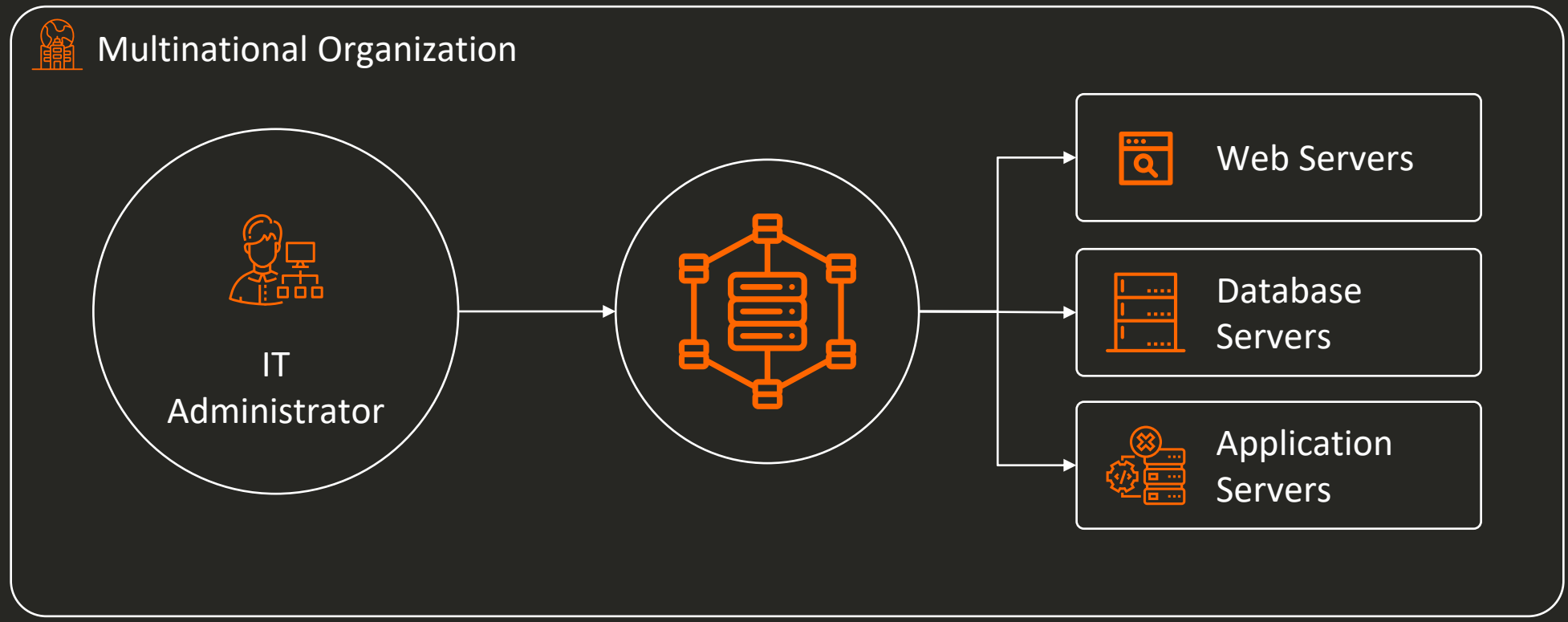
```yaml
all:
  children:
    webservers:
      hosts:
        web1.example.com:
        web2.example.com:
    dbservers:
      hosts:
        db1.example.com:
        db2.example.com:
```

KodeKloud

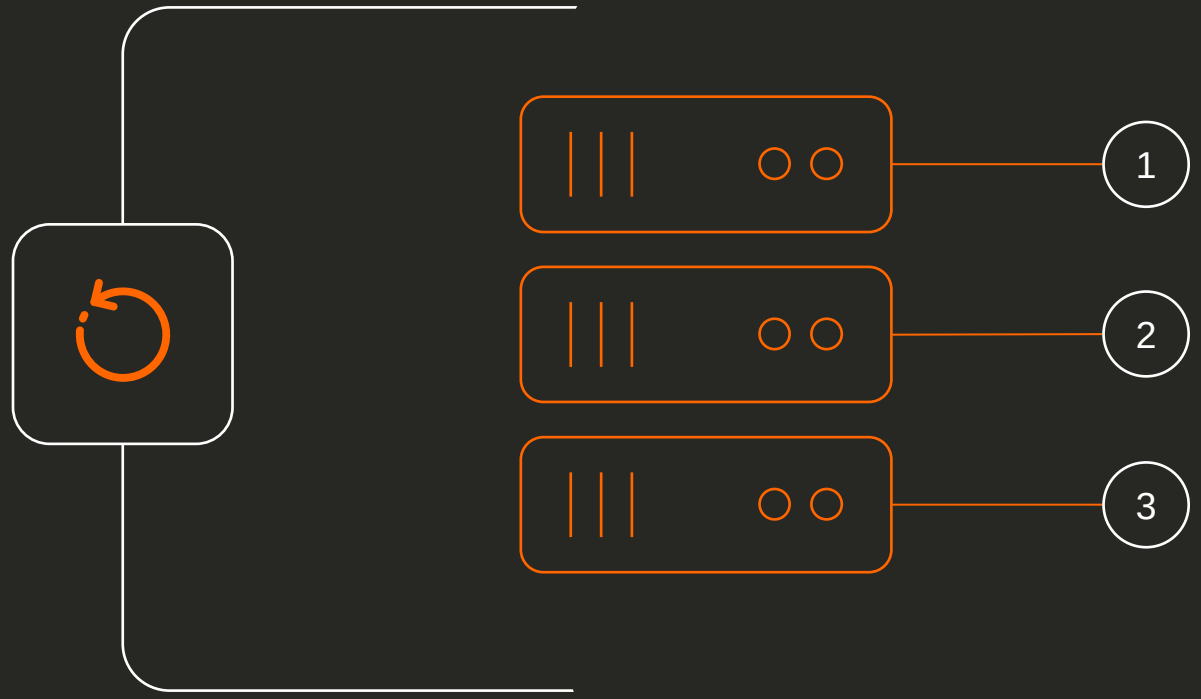
Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Grouping and Parent-Child Relationships

Introduction

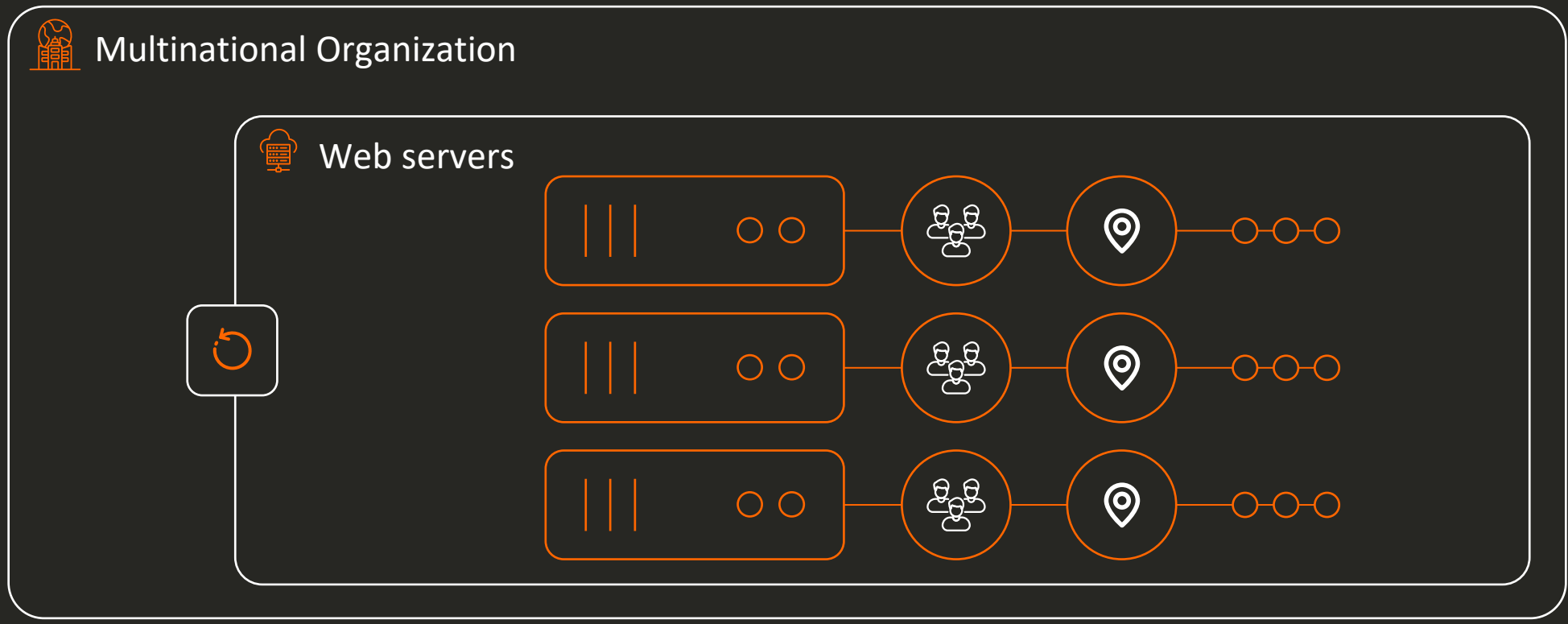


Why Do We Need Grouping?

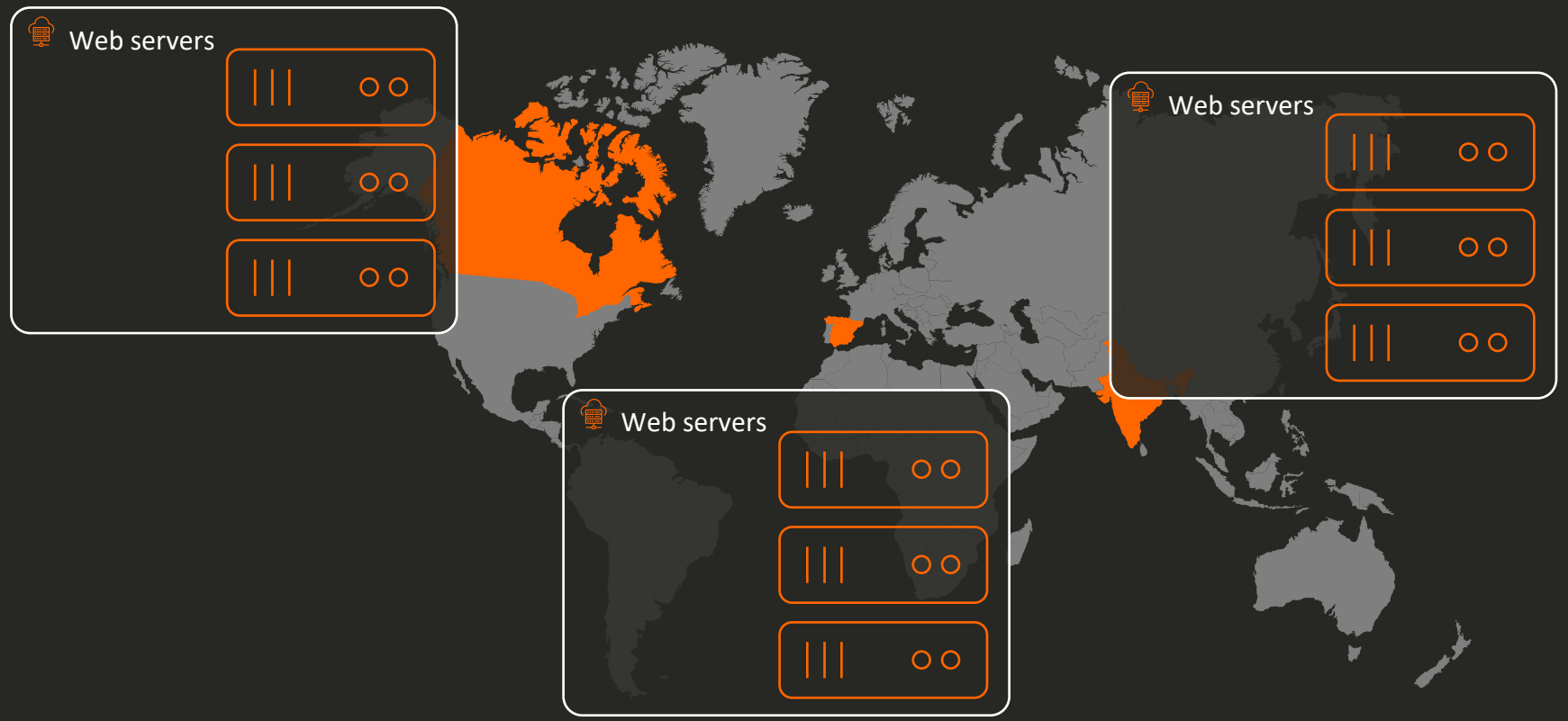


- Time Consuming
- Prone to error

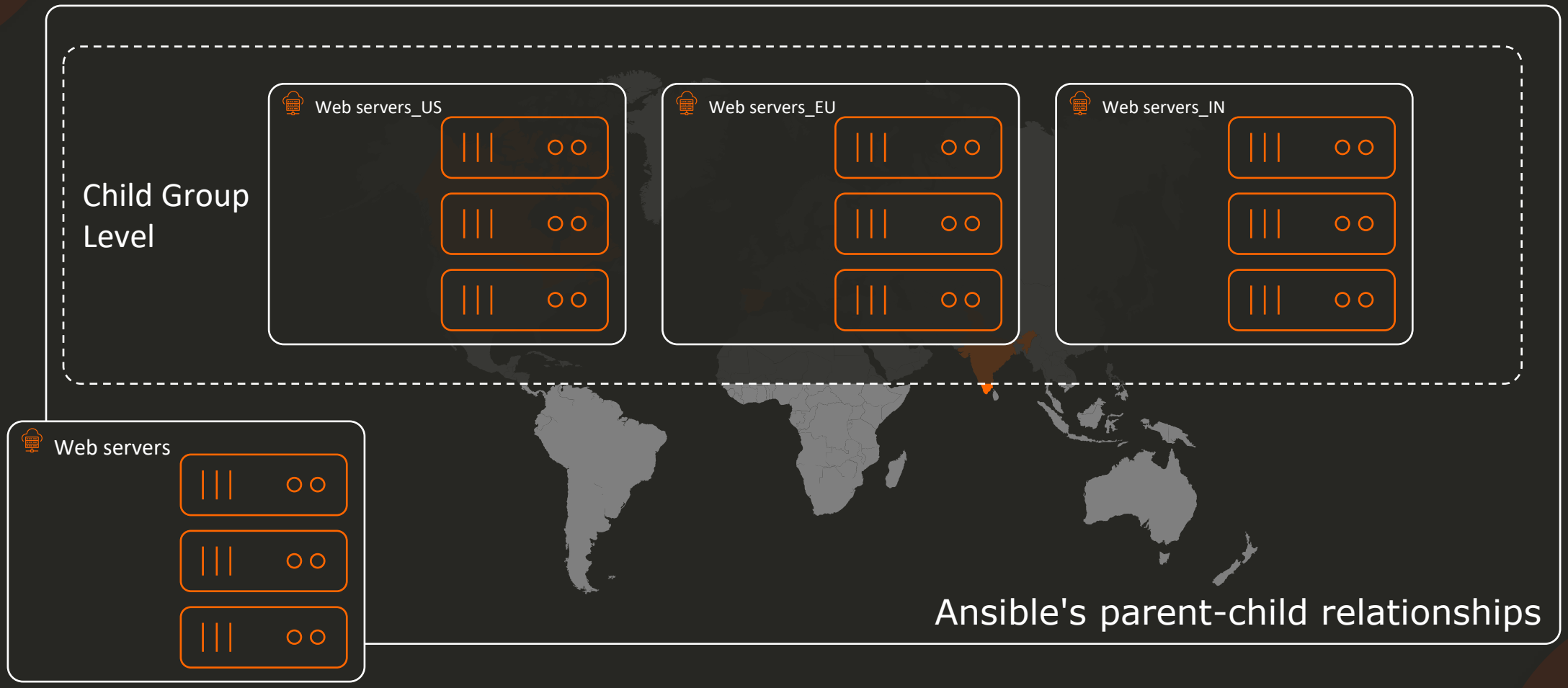
Why Do We Need Grouping?



Why Do We Need Parent-Child Relationships?



Why Do We Need Parent-Child Relationships?



Grouping and Parent-Child Relationships Examples

INI | YAML

```
[webservers:children]
webservers_us
Webservers_eu


[webservers_us]
server1_us.com ansible_host=192.168.8.101
server2_us.com ansible_host=192.168.8.102

[webservers_eu]
server1_eu.com ansible_host=10.12.0.101
server2_eu.com ansible_host=10.12.0.102
```

Grouping and Parent-Child Relationships Examples

INI | YAML

```yaml
all:
  children:
    webservers:
      children:
        webservers_us:
          hosts:
            server1_us.com:
                ansible_host: 192.168.8.101
            server2_us.com:
                ansible_host: 192.168.8.102
        webservers_eu:
          hosts:
            server1_eu.com:
                ansible_host: 10.12.0.101
            server2_eu.com:
                ansible_host: 10.12.0.102
```

KodeKloud

Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Ansible

Variables

Variable

- Stores information that varies with each host

inventory

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

Playbook.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'
```

variables

```
variable1: value1
variable2: value2
```

Using variables

Playbook.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 {{ dns_server }}’
```

```
-
  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=     snmp_port=        inter_ip_range=
```

```
#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`

Variable Types

String Variables

- String variables in Ansible are sequences of characters.
- They can be defined in a playbook, inventory, or passed as command line arguments.

```
username: "admin"
```

Number Variables

- Number variables in Ansible can hold integer or floating-point values.
- They can be used in mathematical operations.

```
max_connections: 100
```

Boolean Variables

- Boolean variables in Ansible can hold either true or false.
- They are often used in conditional statements.

```
debug_mode: true
```

Valid values	Description
True , 'true' , 't' , 'yes' , 'y' , 'on' , '1' , 1 , 1.0	Truthy values
False , 'false' , 'f' , 'no' , 'n' , 'off' , '0' , 0 , 0.0	Falsy values

List Variables

- List variables in Ansible can hold an ordered collection of values.
- The values can be of any type.

```
packages:
   - nginx
   - postgresql
   - git
```

String Values

List Variables

```yaml
- name: Install Packages Playbook
  hosts: your_target_hosts
  vars:
    packages:
      - nginx
      - postgresql
      - git

  tasks:
    - name: Display all packages
      debug:
        var: packages

    - name: Display the first package
      debug:
        msg: "The first package is {{ packages[0] }}"

    - name: Install packages using package manager (apt/yum)
      become: true  # To escalate privileges for package installation, if required
      # Replace with appropriate package management tasks based on the target system (apt/yum)
      # For this example, we'll just use the debug module to simulate the installation
      debug:
        msg: "Installing package {{ item }}"
      loop: "{{ packages }}"
```

Dictionary Variables

- Dictionary variables in Ansible can hold a collection of key-value pairs.
- The keys and values can be of any type.

```
user:
   name: "admin"
   password: "secret"
```

key-value pairs

Dictionary Variables

```yaml
- name: Access Dictionary Variable Playbook
  hosts: web_servers
  vars:
    user:
      name: "admin"
      password: "secret"

  tasks:
    - name: Display the entire user dictionary variable
      debug:
        var: user

    - name: Access specific values in the dictionary
      debug:
        msg: "Username:{{ user.name }}, Password:{{ user.password }}"
```

KodeKloud

Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Ansible Variable Precedence

Variable Precedence

/etc/ansible/hosts

```
web1 ansible_host=172.20.1.100
web2 ansible_host=172.20.1.101   dns_server=10.5.5.4
web3 ansible_host=172.20.1.102


[web_servers]
web1
Web2
web3

[web_servers:vars]
dns_server=10.5.5.3
```

Group Vars

Host Vars

Variable Precedence

```
---
- name: Configure DNS Server
  hosts: all

  vars:
    dns_server: 10.5.5.5
  tasks:
  - nsupdate:
      server: '{{ dns_server }}'
```

Group Vars

Host Vars

Play Vars

Variable Precedence

```
$ ansible-playbook playbook.yml --extra-vars dns_server=10.5.5.6
```

- Group Vars
- Host Vars
- Play Vars
- Extra Vars

Variable Precedence

- Role Defaults
- Group Vars
- Host Vars
- Host Facts
- Play Vars
- Role Vars
- Include Vars
- Set Facts
- Extra Vars

- role defaults [1]
- inventory file or script group vars [2]
- inventory group_vars/all [3]
- playbook group_vars/all [3]
- inventory group_vars/* [3]
- playbook group_vars/* [3]
- inventory file or script host vars [2]
- inventory host_vars/*
- playbook host_vars/*
- host facts / cached set_facts [4]
- inventory host_vars/* [3]
- playbook host_vars/* [3]
- host facts
- play vars
- play vars_prompt
- play vars_files
- role vars (defined in role/vars/main.yml)
- block vars (only for tasks in block)
- task vars (only for the task)
- include_vars
- set_facts / registered vars
- role (and include_role) params
- include params
- extra vars (always win precedence)

https://docs.ansible.com/ansible/2.5/user_guide/playbooks_variables.htm

KodeKloud

Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Ansible Register Variables

playbook

```
---
- name: Check /etc/hosts file
  hosts: all
  tasks:
  - shell: cat /etc/hosts


     register: result

  - debug:
      var:
```

```
PLAY [Check /etc/hosts file] ***************************************************

TASK [shell] *******************************************************************
changed: [web1]
changed: [web2]


PLAY RECAP *********************************************************************
web1            : ok=1   changed=1   unreachable=0   failed=0   skipped=0   rescued=0   ignored=0
web2            : ok=1   changed=1   unreachable=0   failed=0   skipped=0   rescued=0   ignored=0
```

Register Output

playbook

```
---
- name: Check /etc/hosts file
  hosts: all
  tasks:
  - shell: cat /etc/hosts


     register: result

  - debug:
      var:
               result
```

```
ok: [web2] => {
   "output": {
      "ansible_facts": {
         "discovered_interpreter_python": "/usr/bin/python"
      },
      "changed": true,
      "cmd": "cat /etc/hosts",
      "failed": false,
      "rc": 0,
      "start": "2019-09-12 05:25:34.158877",
      "end": "2019-09-12 05:25:34.161974",
      "delta": "0:00:00.003097",
      "stderr": "",
      "stderr_lines": [],
      "stdout": "127.0.0.1\tlocalhost\n::1\tlocalhost ip6-localhost ip6-loopback\
mcastprefix\nff02::1\tip6-allnodes\nff02::2\tip6-allrouters\n172.20.1.101\twe
      "stdout_lines": [
         "127.0.0.1\tlocalhost",
         "::1\tlocalhost ip6-localhost ip6-loopback",
         "fe00::0\tip6-localnet",
         "ff00::0\tip6-mcastprefix",
         "ff02::1\tip6-allnodes",
         "ff02::2\tip6-allrouters",
         "172.20.1.101\tweb2"
      ]
   }
}
```

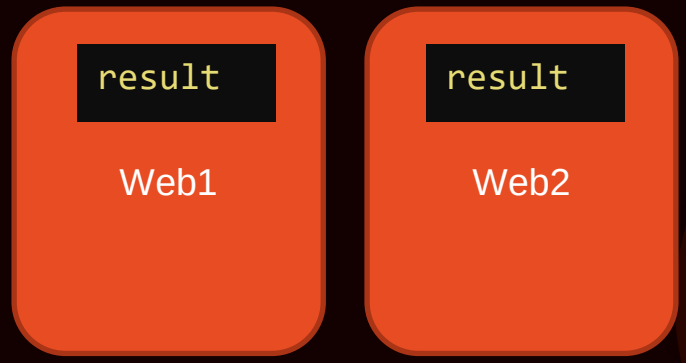
.rc

.stdout

Register Output Scope

playbook

```
---
- name: Check /etc/hosts file
  hosts: all
  tasks:
  - shell: cat /etc/hosts

     register: result

  - debug:
      var:
               result.rc


- name: Play2
  hosts: all
  tasks:
  - debug:
      var: result.rc
```



playbook

```
---
- name: Check /etc/hosts file
  hosts: all
  tasks:
  - shell: cat /etc/hosts
```

```
$ ansible-playbook -i inventory playbook.yml -v
```

```
PLAY [localhost] ***************************************************************

TASK [Gathering Facts] *********************************************************
ok: [localhost]

TASK [shell] *******************************************************************
changed: [localhost] => {"changed": true, "cmd": "cat /etc/hosts", "delta": "0:00:00.282432", "end": "2019-09-24 07:37:26.440478", "rc": 0,
"start": "2019-09-24 07:37:26.158046", "stderr": "", "stderr_lines": [], "stdout": "127.0.0.1\tlocalhost\n::1\tlocalhost ip6-localhost ip6-
loopback\nfe00::0\tip6-localnet\nff00::0\tip6-mcastprefix\nff02::1\tip6-allnodes\nff02::2\tip6-allrouters\n172.20.1.2\tf6d0e5fbbc0d",
"stdout_lines": ["127.0.0.1\tlocalhost", "::1\tlocalhost ip6-localhost ip6-loopback", "fe00::0\tip6-localnet", "ff00::0\tip6-mcastprefix",
"ff02::1\tip6-allnodes", "ff02::2\tip6-allrouters", "172.20.1.2\tf6d0e5fbbc0d"]}

PLAY RECAP *********************************************************************
localhost                  : ok=2    changed=1    unreachable=0    failed=0
```

KodeKloud

Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Ansible Variable Scopes

Variable Scopes

/etc/ansible/hosts

```
web1 ansible_host=172.20.1.100
web2 ansible_host=172.20.1.101 dns_server=10.5.5.4
web3 ansible_host=172.20.1.102
```

```
---
- name: Print dns server
  hosts: all
  tasks:
  - debug:
      msg: '{{ dns_server }}'
```

```
PLAY [Check /etc/hosts file]
**********************************************


TASK [debug] ************************************
ok: [web1] => {
   "dns_server": "VARIABLE IS NOT DEFINED!"
}
ok: [web2] => {
   "dns_server": "10.5.5.4"
}
ok: [web3] => {
   "dns_server": "VARIABLE IS NOT DEFINED!"
}
```

Variable Scopes - Host

Variable Scopes - Host

Variable Scopes - Play

```yaml
---
- name: Play1
  hosts: web1
  vars:
    ntp_server: 10.1.1.1
  tasks:
  - debug:
      var: ntp_server

- name: Play2
  hosts: web1
  tasks:
  - debug:
      var: ntp_server
```

```
PLAY [Play1] ***********************************************************


TASK [debug] ***********************************************************
ok: [web1] => {
   "ntp_server": "10.1.1.1"
}


PLAY [Play2] ***********************************************************

TASK [debug] ***********************************************************
ok: [web1] => {
   "ntp_server": "VARIABLE IS NOT DEFINED!"
}
```

Variable Scopes - Global

```
$ ansible-playbook playbook.yml --extra-vars “ntp_server=10.1.1.1”
```

```
---
- name: Play1
  hosts: web1
  vars:
    ntp_server: 10.1.1.1
  tasks:
  - debug:
      var: ntp_server

- name: Play2
  hosts: web1
  tasks:
  - debug:
      var: ntp_server
```

```
PLAY [Play1] ***********************************************************


TASK [debug] ***********************************************************
ok: [web1] => {
   "ntp_server": "10.1.1.1"
}


PLAY [Play2] ***********************************************************

TASK [debug] ***********************************************************
ok: [web1] => {
   "ntp_server": "10.1.1.1"
}
```

KodeKloud

Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Ansible Magic Variables

Variable Scopes

/etc/ansible/hosts

```
web1 ansible_host=172.20.1.100
web2 ansible_host=172.20.1.101  dns_server=10.5.5.4
web3 ansible_host=172.20.1.102
```

```
web1 ansible_host=172.20.1.100
web2 ansible_host=172.20.1.101  dns_server=10.5.5.4
web3 ansible_host=172.20.1.102
```

```
---
- name: Print dns server
  hosts: all
  tasks:
  - debug:

        msg: '{{ dns_server }}'
```

```
PLAY [Check /etc/hosts file]
**********************************************


TASK [debug] ************************************
ok: [web1] => {
   "dns_server": "VARIABLE IS NOT DEFINED!"
}
ok: [web2] => {
   "dns_server": "10.5.5.4"
}
ok: [web3] => {
   "dns_server": "VARIABLE IS NOT DEFINED!"
}
```

Variable Interpolation

```
inventory_hostname=web1
ansible_host=172.20.1.100
```

```
inventory_hostname=web2
ansible_host=172.20.1.101
dns_server=10.5.5.4
```

```
inventory_hostname=web3
ansible_host=172.20.1.102
```

Create Subprocess

```
web1 ansible_host=172.20.1.100
web2 ansible_host=172.20.1.101  dns_server=10.5.5.4
web3 ansible_host=172.20.1.102
```

```
---
- name: Print dns server
  hosts: all
  tasks:
  - debug:

        msg: '{{       hostvars['web2dns_server }}'
```

```
PLAY [Check /etc/hosts file]
**********************************************


TASK [debug] ************************************
ok: [web1] => {
   "dns_server": "10.5.5.4"
}
ok: [web2] => {
   "dns_server": "10.5.5.4"
}
ok: [web3] => {
   "dns_server": "10.5.5.4"
}
```

Variable Interpolation

```
inventory_hostname=web1
ansible_host=172.20.1.100
```

```
inventory_hostname=web2
ansible_host=172.20.1.101
dns_server=10.5.5.4
```

```
inventory_hostname=web3
ansible_host=172.20.1.102
```

Create Subprocess

```
---
- name: Print dns server
  hosts: all
  tasks:
  - debug:

        msg: '{{       hostvars['web2dns_server }}'
```

```
        msg: '{{ hostvars['web2'].ansible_host }}'
```

```
        msg: '{{ hostvars['web2'].ansible_facts.architecture }}'
```

```
        msg: '{{ hostvars['web2'].ansible_facts.devices }}'
```

```
        msg: '{{ hostvars['web2'].ansible_facts.mounts }}'
```

```
        msg: '{{ hostvars['web2'].ansible_facts.processor }}'
```

```
---
- name: Print dns server
  hosts: all
  tasks:
  - debug:

        msg: '{{       hostvars['web2dns_server }}'
```

```
msg: '{{ hostvars['web2'].ansible_host }}'
```

```
msg: '{{ hostvars['web2'].ansible_facts.architecture }}'
```

```
msg: '{{ hostvars['web2'].ansible_facts.devices }}'
```

```
msg: '{{ hostvars['web2'].ansible_facts.mounts }}'
```

```
msg: '{{ hostvars['web2'].ansible_facts.processor }}'
```

```
msg: '{{ hostvars['web2']['ansible_facts']['processor'] }}'
```

Magic Variable - hostvars

```
msg: ‘{{ hostvars['web2'].ansible_host }}’
```

```
msg: ‘{{ hostvars['web2'].ansible_facts.architecture }}’
```

```
msg: ‘{{ hostvars['web2'].ansible_facts.devices }}’
```

```
msg: ‘{{ hostvars['web2'].ansible_facts.mounts }}’
```

```
msg: ‘{{ hostvars['web2'].ansible_facts.processor }}’
```

```
msg: ‘{{ hostvars['web2']['ansible_facts']['processor'] }}’
```

Magic Variable - groups

/etc/ansible/hosts

```
web1 ansible_host=172.20.1.100
web2 ansible_host=172.20.1.101
web3 ansible_host=172.20.1.102


[web_servers]
web1
Web2
web3

[americas]
web1
web2

[asia]
web3
```

```
msg: '{{ groups['americas'] }}'

web1
web2
```

Magic Variable – group_names

/etc/ansible/hosts

```
web1 ansible_host=172.20.1.100
web2 ansible_host=172.20.1.101
web3 ansible_host=172.20.1.102


[web_servers]
web1
Web2
web3

[americas]
web1
web2

[asia]
web3
```

```
msg: '{{ group_names }}'  # web1
```

```
web_servers
americas
```

Magic Variable – inventory_hostname

/etc/ansible/hosts

```
web1 ansible_host=172.20.1.100
web2 ansible_host=172.20.1.101
web3 ansible_host=172.20.1.102


[web_servers]
web1
Web2
web3

[americas]
web1
web2

[asia]
web3
```

```
msg: '{{ inventory_hostname }}'  # web1
```

```
web1
```

Accessing information about other hosts with magic variables

Whether or not you define any variables, you can access information about your hosts with the Special Variables Ansible provides, including "magic

The most commonly used magic variables are `hostvars`, `groups`, `group_names`, and `inventory_hostname`.

`hostvars` lets you access variables for another host, including facts that have been gathered about that host. You can access host variables at any p
able to see the facts.

If your database server wants to use the value of a 'fact' from another node, or an inventory variable assigned to another node, it's easy to do so wit

```
{{ hostvars['test.example.com']['ansible_facts']['distribution'] }}
```

`groups` is a list of all the groups (and hosts) in the inventory. This can be used to enumerate all hosts within a group. For example:

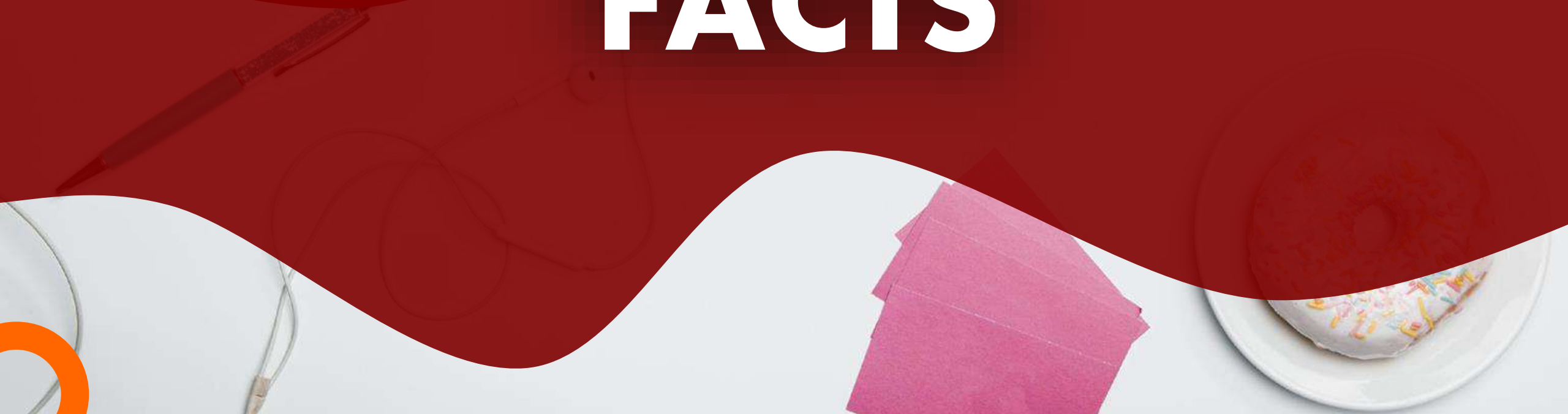
```
{% for host in groups['app_servers'] %}
   # something that applies to all app servers.
{% endfor %}
```

KodeKloud

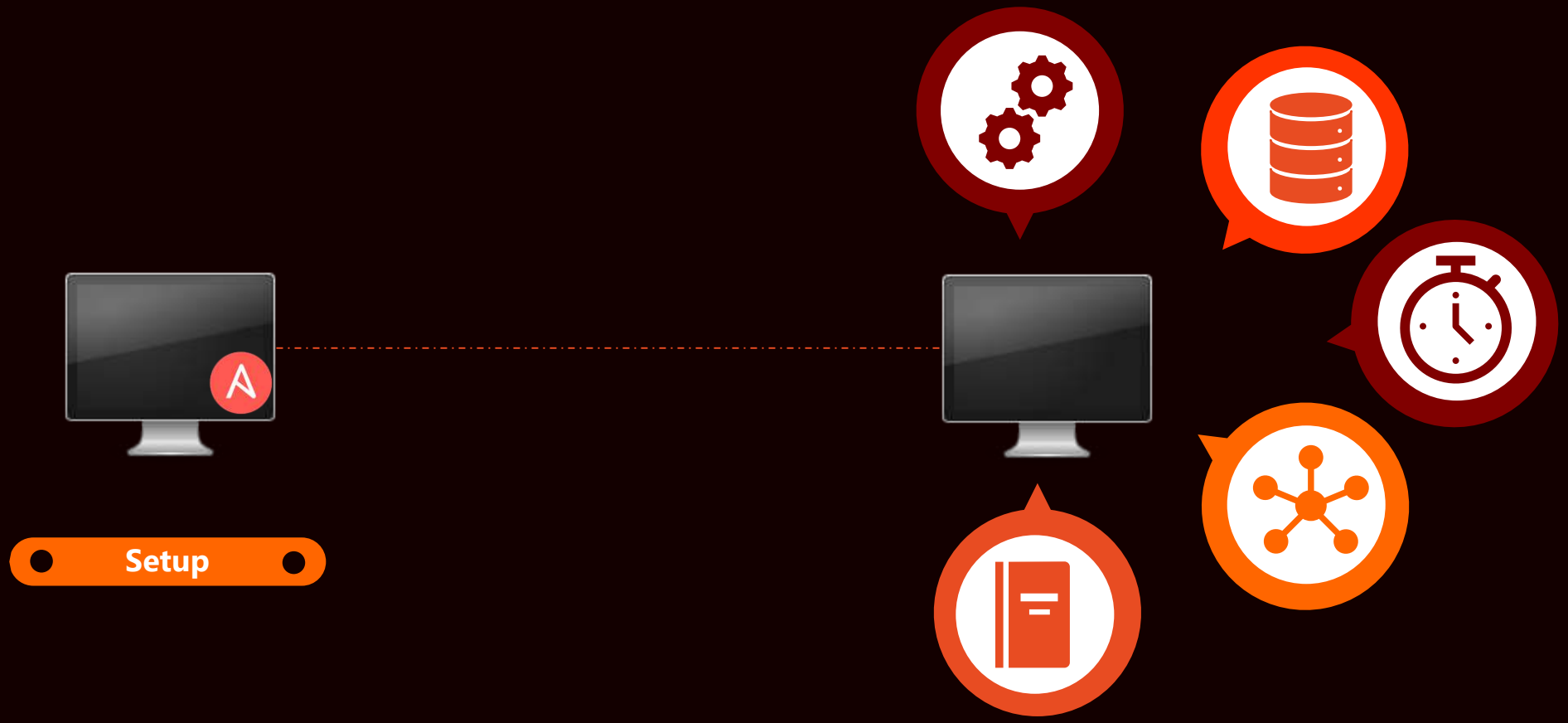
Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Ansible

FACTS



FACTS



```yaml
---
- name: Print hello message
  hosts: all
  tasks:
  - debug:

      msg: Hello from Ansible!
```

```
PLAY [Print hello message] ***************************************************

TASK [Gathering Facts] *******************************************************
ok: [web2]
ok: [web1]

TASK [debug] *****************************************************************
ok: [web1] => {
   "msg": "Hello from Ansible!"
}
ok: [web2] => {
   "msg": "Hello from Ansible!"
}
```

```yaml
---
- name: Print hello message
  hosts: all
  tasks:
  - debug:

      var: ansible_facts
```

```
PLAY [Reset nodes to previous state] ********************************************


TASK [Gathering Facts] **********************************************************
ok: [web2]
ok: [web1]


TASK [debug] ********************************************************
ok: [web1] => {
   "ansible_facts": {
      "all_ipv4_addresses": [
         "172.20.1.100"
      ],
      "architecture": "x86_64",
      "date_time": {
         "date": "2019-09-07",
      },
      "distribution": "Ubuntu",
      "distribution_file_variety": "Debian",
      "distribution_major_version": "16",
      "distribution_release": "xenial",
      "distribution_version": "16.04",
      "dns": {
         "nameservers": [
            "127.0.0.11"
         ],
      },
      "fqdn": "web1",
      "hostname": "web1",
      "interfaces": [
         "lo",
         "eth0"
      ],
      "machine": "x86_64",
      "memfree_mb": 72,
      "memory_mb": {
         "real": {
            "free": 72,
            "total": 985,
            "used": 913
         },
      },
```

```yaml
---
- name: Print hello message
  hosts: all
  tasks:
  - debug:

      var: ansible_facts
```


```
      "eth0"
    ],
    "machine": "x86_64",
    "memfree_mb": 72,
    "memory_mb": {
      "real": {
        "free": 72,
        "total": 985,
        "used": 913
      },
    },
    "memtotal_mb": 985,
    "module_setup": true,
    "mounts": [
      {
        "block_available": 45040,
        "block_size": 4096,
        "block_total": 2524608,
        "block_used": 2479568,
      },
    ],
    "nodename": "web1",
    "os_family": "Debian",
    "processor": [
      "0",
      "GenuineIntel",
      "Intel(R) Core(TM) i9-9980HK CPU @ 2.40GHz",
    ],
    "processor_cores": 2,
    "processor_count": 1,
    "processor_threads_per_core": 1,
    "processor_vcpus": 2,
    "product_name": "VirtualBox",
    "product_serial": "0",
    "product_uuid": "18A31B5D-FAC9-445F-9B6F-95B4B587F485",
    "product_version": "1.2",

  }
}
```

```yaml
---
- name: Print hello message
  hosts: all

  gather_facts: no
  tasks:
  - debug:

      var: ansible_facts
```

```
PLAY [Print hello message] ***************************************************


TASK [debug] *****************************************************************
ok: [web1] => {
    "ansible_facts": {}
}
ok: [web2] => {
    "ansible_facts": {}
}
```

```yaml
---
- name: Print hello message
  hosts: all

  gather_facts: no
  tasks:
  - debug:

      var: ansible_facts
```

```
PLAY [Print hello message] *****************************************************

TASK [debug] *******************************************************************
ok: [web1] => {
    "ansible_facts": {}
}
ok: [web2] => {
    "ansible_facts": {}
}
```

/etc/ansible/ansible.cfg

```
# plays will gather facts by default, which contain information about
# smart - gather by default, but don't regather if already gathered
# implicit - gather by default, turn off with gather_facts: False
# explicit - do not gather by default, must say gather_facts: True
gathering              = implicit
```

```
---
- name: Print hello message
  hosts: web1
  tasks:
  - debug: ansible_facts
```

/etc/ansible/hosts

```
web1
web2
```

KodeKloud

Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Ansible

Playbooks

Ansible playbooks

```
# 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

- 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

playbook.yml

```yaml
-
  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
```

Playbook format

playbook.yml

```yaml
-
  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
```

Hosts

playbook.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
```

inventory

```
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
```

module

playbook.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-doc -l

Run

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

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

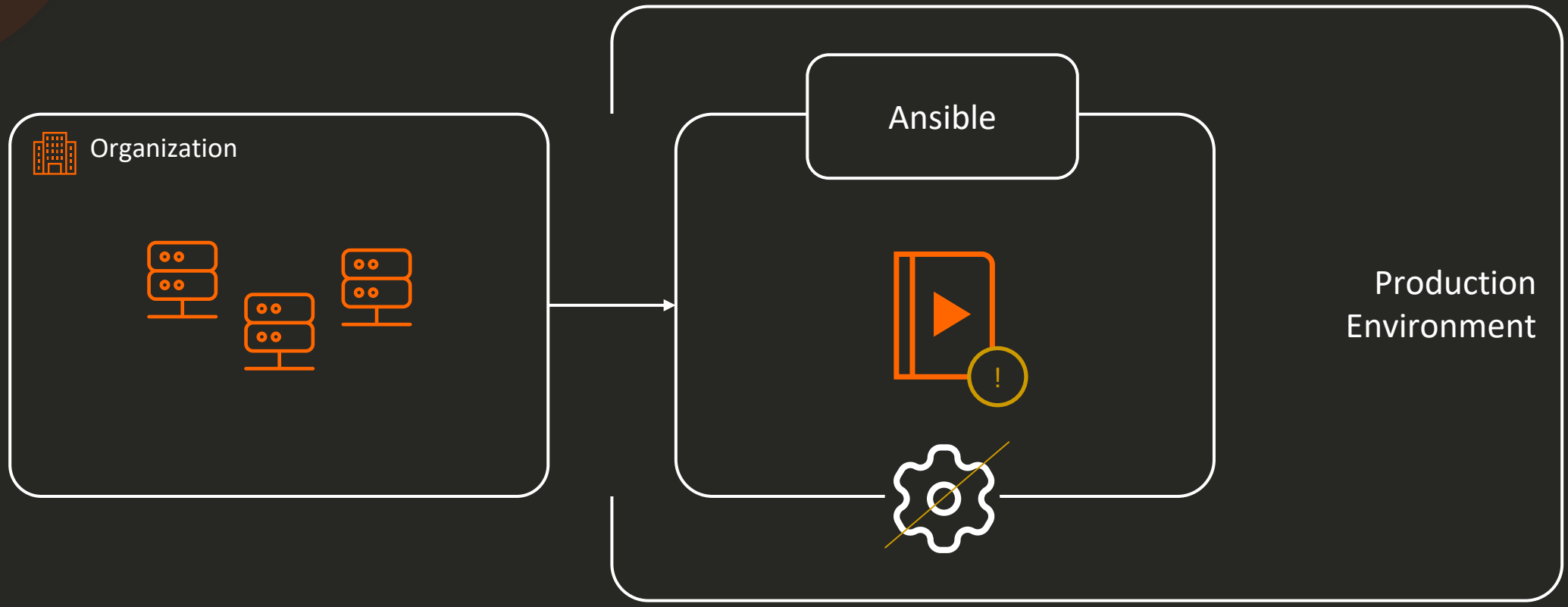
```
ansible-playbook --help
```

KodeKloud

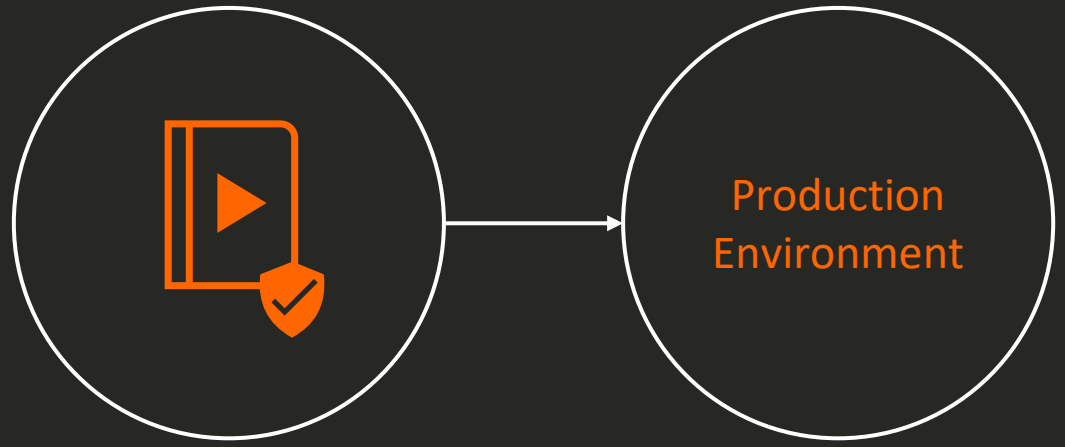
Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Verifying Playbooks

Introduction



Why Do We Need to Verify Playbooks?



Challenging and time-consuming to resolve

How to Verify Playbooks in Ansible?

Check Mode

Diff Mode

How to Verify Playbooks in Ansible?

Check Mode

Diff Mode

Ansible's "dry run" where no actual changes are made on hosts

Allows preview of playbook changes without applying them

Use the --check option to run a playbook in check mode

How to Verify Playbooks in Ansible?

Check Mode

Diff Mode

install_nginx.yml

```
---

- hosts: webservers
  tasks:
    - name: Ensure nginx is installed
      apt:
        name: nginx
        state: present
      become: yes
```

How to Verify Playbooks in Ansible?

Check Mode

Diff Mode

```
$ ansible-playbook install_nginx.yml --check
```

```
- PLAY [webservers] ************************************************************

  TASK [Gathering Facts] ******************************************************
  ok: [webserver1]

  TASK [Ensure nginx is installed] ********************************************
  changed: [webserver1]

  PLAY RECAP
- ********************************************************************
  webserver1: ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

How to Verify Playbooks in Ansible?

Check Mode

Diff Mode

Provides a before-and-after comparison of playbook changes

Understand and verify the impact of playbook changes before applying them

Utilize the --diff option to run a playbook in diff mode

How to Verify Playbooks in Ansible?

Check Mode

Diff Mode

configure_nginx.yml

```
---

- hosts: webservers
  tasks:
    - name: Ensure the configuration line is present
      lineinfile:
        path: /etc/nginx/nginx.conf
        line: 'client_max_body_size 100M;'
      become: yes
```

How to Verify Playbooks in Ansible?

Check Mode

Diff Mode

```
$ ansible-playbook configure_nginx.yml --check --diff
```

```
- PLAY [webservers] ************************************************************
  TASK [Gathering Facts] ********************************************************
  ok: [webserver1]
  TASK [Ensure the configuration line is present] *******************************
  --- before: /etc/nginx/nginx.conf (content)
  +++ after: /etc/nginx/nginx.conf (content)
  @@ -20,3 +20,4 @@
  # some existing configuration lines
  # more existing configuration lines
  #
  +client_max_body_size 100M;
  changed: [webserver1]
  PLAY RECAP
- ********************************************************************
  webserver1: ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Syntax Check

Ensures playbook syntax is error-free

Use the --syntax-check option

Syntax Check

configure_nginx.yml

```
---

- hosts: webservers
  tasks:
    - name: Ensure the configuration line is present
      lineinfile:
        path: /etc/nginx/nginx.conf
        line: 'client_max_body_size 100M;'
      become: yes
```

```
$ ansible-playbook configure_nginx.yml --syntax-check
```

```
playbook: configure_nginx.yml
```

Syntax Check

configure_nginx.yml

```
---
- hosts: webservers
  tasks:
    - name: Ensure the configuration line is present
      lineinfile
        path: /etc/nginx/nginx.conf
        line: 'client_max_body_size 100M;'
      become: yes
```

“:” is missing

Syntax Check

```
$ ansible-playbook configure_nginx.yml --syntax-check
```

```
ERROR! Syntax Error while loading YAML.
  did not find expected key

The error appears to be in '/path/to/configure_nginx.yml': line 5, column 9, but may
be elsewhere in the file depending on the exact syntax problem.

The offending line appears to be:

      lineinfile
        path: /etc/nginx/nginx.conf
        ^ here
```

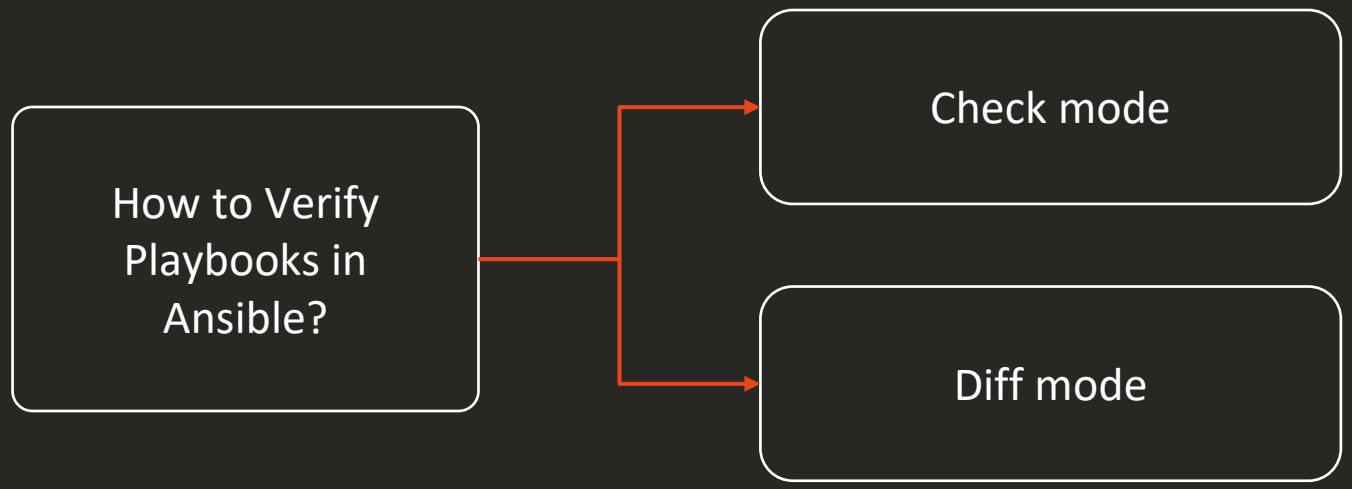
KodeKloud

Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

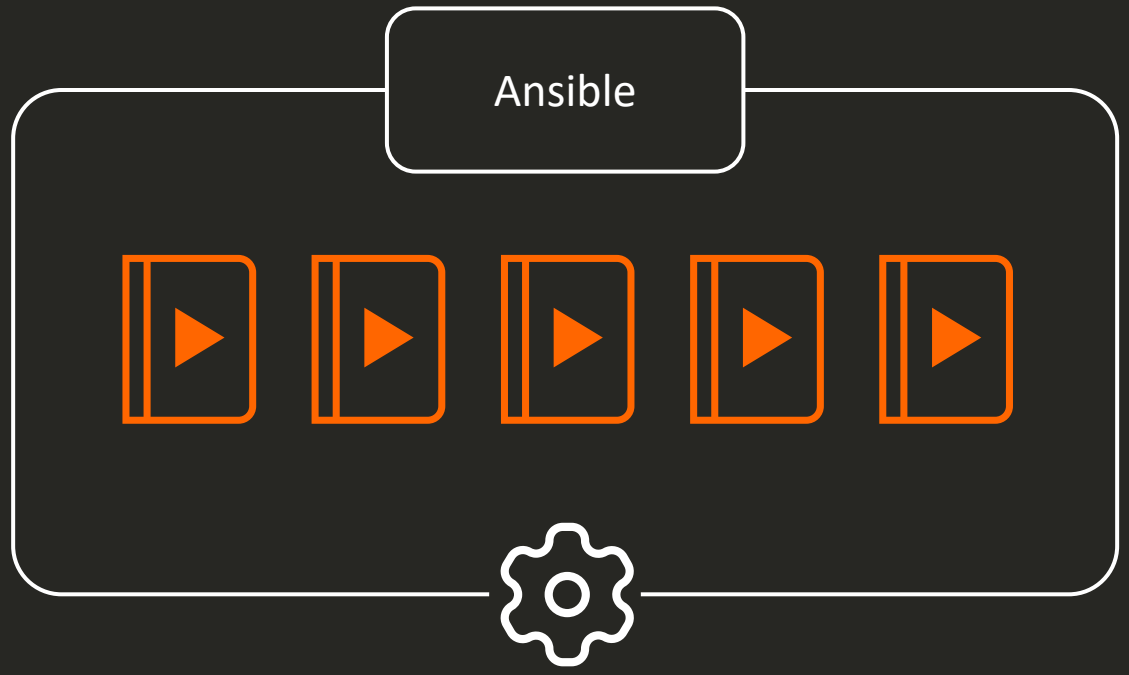
Ansible-lint

Introduction

The importance of verifying playbooks in Ansible



Introduction



Why Do We Need ansible-lint?

Ansible Lint is a command-line tool that performs linting on Ansible playbooks, roles, and collections.

It checks your code for potential errors, bugs, stylistic errors, and suspicious constructs.

It's akin to having a seasoned Ansible mentor guiding you, providing valuable insights, and catching issues that might have slipped past your notice.

How to Use ansible-lint?

style_example.yml

```
- name: Style Example Playbook
  hosts: localhost
  tasks:
    - name: Ensure nginx is installed and started
      apt:
        name: nginx
        state: latest
        update_cache: yes

    - name: Enable nginx service at boot

service:
        name: nginx
        enabled: yes
        state: started
- name: Copy nginx configuration file
      copy:
        src: /path/to/nginx.conf
        dest: /etc/nginx/nginx.conf
      notify:
        - Restart nginx service

  handlers:
    - name: Restart nginx service
      service:
        name: nginx
        state: restarted
```

How to Use ansible-lint?

```
$ ansible-lint style_example.yml
```

```
[WARNING]: incorrect indentation: expected 2 but found 4 (syntax/indentation)
style_example.yml:6

[WARNING]: command should not contain whitespace (blacklisted: ['apt']) (commands)
style_example.yml:6

[WARNING]: Use shell only when shell functionality is required (deprecated in favor of 'cmd') (commands)
style_example.yml:6

[WARNING]: command should not contain whitespace (blacklisted: ['service']) (commands)
style_example.yml:12

[WARNING]: 'name' should be present for all tasks (task-name-missing) (tasks)
style_example.yml:14
```

KodeKloud

Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Ansible

Conditionals

```yaml
---
- name: Install NGINX
  hosts: debian_hosts
  tasks:
  - name: Install NGINX on Debian
    apt:
      name: nginx
      state: present
```

```yaml
---
- name: Install NGINX
  hosts: redhat_hosts
  tasks:
  - name: Install NGINX on Redhat
    yum:
      name: nginx
      state: present
```

NOTE: PARTS OF CODE HIDDEN FOR BREVITY

Conditional - when

```
---
- name: Install NGINX
  hosts: all
  tasks:
  - name: Install NGINX on Debian
    apt:
      name: nginx
      state: present
    when:  ansible_os_family == “Debian”

  - name: Install NGINX on Redhat
    yum:
      name: nginx
      state: present
    when:  ansible_os_family == “RedHat”
```

Operator - or

```
---
- name: Install NGINX
  hosts: all
  tasks:
  - name: Install NGINX on Debian
    apt:
      name: nginx
      state: present
    when:  ansible_os_family == “Debian”

  - name: Install NGINX on Redhat
    yum:
      name: nginx
      state: present
    when:  ansible_os_family == “RedHat”  or
           ansible_os_family == “SUSE”
```

Operator - and

```
---
- name: Install NGINX
  hosts: all
  tasks:
  - name: Install NGINX on Debian
    apt:
      name: nginx
      state: present
    when:  ansible_os_family == “Debian”    and
           ansible_distribution_version == “16.04”

  - name: Install NGINX on Redhat
    yum:
      name: nginx
      state: present
    when:  ansible_os_family == “RedHat”  or
           ansible_os_family == “SUSE”
```

Conditionals in Loops

```
---
- name: Install NGINX
  hosts: all
  tasks:
  - name: Install NGINX on Debian
    apt:
      name: nginx
      state: present
```

Conditionals in Loops

```
---
- name: Install Softwares
  hosts: all
  vars:
     packages:
        - name: nginx
          required: True
        - name: mysql
          required : True
        - name: apache
          required : False
  tasks:
  - name: Install “{{ item.name }}” on Debian
    apt:
      name: “{{ item.name }}”
      state: present


    loop: "{{ packages }}"
```

Conditionals in Loops

```yaml
---
- name: Install Softwares
  hosts: all
  vars:
     packages:
        - name: nginx
          required: True
        - name: mysql
          required : True
        - name: apache
          required : False
  tasks:
  - name: Install  “{{ item.name }}” on Debian
    apt:
      name: “{{ item.name }}”
      state: present


    loop: "{{ packages }}"
```

```yaml
- name: Install “{{ item.name }}” on Debian
  vars:
   item:
    name: nginx
    required: True
  apt:
    name: “{{ item.name }}”
    state: present
  when:   item.required == True
```

```yaml
- name: Install “{{ item.name }}” on Debian
  vars:
   item:
    name: mysql
    required: True
  apt:
    name: “{{ item.name }}”
    state: present
  when:   item.required == True
```

```yaml
-  name: Install “{{ item.name }}” on Debian
   vars:
    item:
     name: apache
     required: False
   apt:
    name: “{{ item.name }}”
    state: present
  when:   item.required == True
```

Conditionals in Loops

```
---
- name: Install Softwares
  hosts: all
  vars:
     packages:
        - name: nginx
          required: True
        - name: mysql
          required : True
        - name: apache
          required : False
  tasks:
  - name: Install “{{ item.name }}” on Debian
    apt:
      name: “{{ item.name }}”
      state: present

    when:   item.required == True

    loop: "{{ packages }}"
```

Conditionals & Register

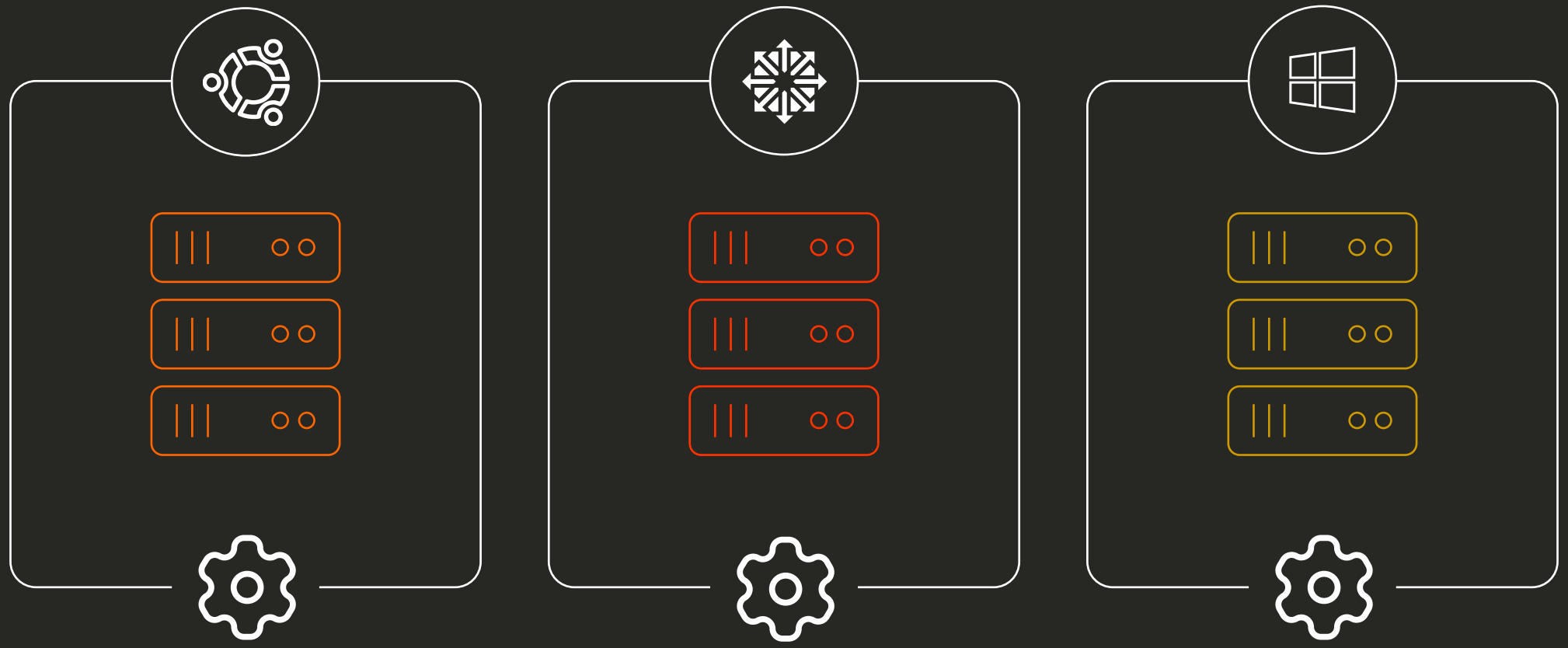
```
- name: Check status of a service and email if its down
  hosts: localhost
  tasks:
    - command: service httpd status
      register: result

    -  mail:
       to: admin@company.com
       subject: Service Alert
       body: Httpd Service is down
       when: result.stdout.find('down') != -1
```

KodeKloud

Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Ansible Conditionals based on facts, variables, re-use

Introduction



Scenario 1

```
- name: Install Nginx on Ubuntu 18.04
-  apt:
   - name: nginx=1.18.0
   - state: present
- when: ansible_facts['os_family'] == 'Debian' and ansible_facts['distribution_major_version'] == '18'
```

Scenario 2

```
- name: Deploy configuration files
-  template:
   - src: “{{ app_env }}_config.j2”
   - dest: “/etc/myapp/config.conf”
   vars:
     app_env: production
```

Scenario 3

```
- name: Install required packages
   apt:
   name:
   - - package1
   - - package2
   state: present

- name: Create necessary directories and set permissions

…

- name: Start web application service
   service:
     name: myapp
     state: started
   when: environment == 'production'
```

KodeKloud

Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Ansible

Loops

LOOPS

```
-
  name: Create users
  hosts: localhost
  tasks:
   - user: name={{ item }}   state=present
     loop:
       - joe
       - george
       - ravi
       - mani
       - kiran
       - jazlan
       - emaan
       - mazin
       - izaan
       - mike
       - menaal
       - shoeb
       - rani
```

LOOPS - Visualize

```
-
  name: Create users
  hosts: localhost
  tasks:
   - user: name='{{ item }}'   state=present
     loop:
       - joe
       - george
       - ravi
       - mani
       - kiran
       - jazlan
       - emaan
       - mazin
       - izaan
       - mike
       - menaal
       - shoeb
       - rani
```

```
-
  name: Create users
  hosts: localhost
  tasks:

  -  var:  item=
     user: name= "{{ item }}"     state=present
  -  var:  item=
     user: name= "{{ item }}"     state=present
  -  var:  item=
     user: name= "{{ item }}"     state=present
  -  var:  item=
     user: name= "{{ item }}"     state=present
  -  var:  item=
     user: name= "{{ item }}"     state=present
  -  var:  item=
     user: name= "{{ item }}"     state=present
  -  var:  item=
     user: name= "{{ item }}"     state=present
  -  var:  item=
     user: name= "{{ item }}"     state=present
  -  var:  item=
     user: name= "{{ item }}"     state=present
```

LOOPS - Visualize

```
-
  name: Create users
  hosts: localhost
  tasks:
   - user: name='{{ item }}'   state=present
     loop:
       - joe
       - george
       - ravi
       - mani
       - kiran
       - jazlan
       - emaan
       - mazin
       - izaan
       - mike
       - menaal
       - shoeb
       - rani
```

```
-
  name: Create users
  hosts: localhost
  tasks:

  - var:  item=joe
    user: name= "{{ item }}"     state=present
  - var:  item=george
    user: name= "{{ item }}"     state=present
  - var:  item=ravi
    user: name= "{{ item }}"     state=present
  - var:  item=mani
    user: name= "{{ item }}"     state=present
  - var:  item=kiran
    user: name= "{{ item }}"     state=present
  - var:  item=jazlan
    user: name= "{{ item }}"     state=present
  - var:  item=emaan
    user: name= "{{ item }}"     state=present
  - var:  item=mazin
    user: name= "{{ item }}"     state=present
  - var:  item=izaan
    user: name= "{{ item }}"     state=present
```

LOOPS - Visualize

```
-
  name: Create users
  hosts: localhost
  tasks:
   - user: name ‘{{ ???? }}’   state=present  uid= ‘{{ ? }}’
     loop:
       - name: joe
         uid: 1010
       - name: george
         uid: 1011
       - name: ravi
         uid: 1012
       - name: mani
         uid: 1013
       - name: kiran
         uid: 1014
       - name: jazlan
         uid: 1015
       - name: emaan
         uid: 1016
       - name: mazin
         uid: 1017
       - name: izaan
         uid: 1018
       - name: mike
```

```
-
  name: Create users
  hosts: localhost
  tasks:

   -  var:  item=joe
      user: name= “{{ item }}”     state=present

   -  var:  item=george
      user: name= “{{ item }}”     state=present

   -  var:  item=ravi
      user: name= “{{ item }}”     state=present

   -  var:  item=mani
      user: name= “{{ item }}”     state=present

   -  var:  item=kiran
      user: name= “{{ item }}”     state=present

   -  var:  item=jazlan
      user: name= “{{ item }}”     state=present

   -  var:  item=emaan
      user: name= “{{ item }}”     state=present

   -  var:  item=mazin
      user: name= “{{ item }}”     state=present

   -  var:  item=izaan
      user: name= “{{ item }}”     state=present
```

LOOPS - Visualize

```
-
  name: Create users
  hosts: localhost
  tasks:
   - user: name ‘{{ ???? }}’   state=present  uid= ‘{{ ? }}’
     loop:
       - name: joe
         uid: 1010
       - name: george
         uid: 1011
       - name: ravi
         uid: 1012
       - name: mani
         uid: 1013
       - name: kiran
         uid: 1014
       - name: jazlan
         uid: 1015
       - name: emaan
         uid: 1016
       - name: mazin
         uid: 1017
       - name: izaan
         uid: 1018
       - name: mike
```

```
-
  name: Create users
  hosts: localhost
  tasks:
   -  var:
       item:

      user: name= “{{ ???? }}” state=present uid=“{?}”
   -  var:
       item:

      user: name= “{{ ???? }}” state=present uid=“{?}”
   -  var:
       item:

      user: name= “{{ ???? }}” state=present uid=“{?}”
   -  var:
       item:

      user: name= “{{ ???? }}” state=present uid=“{?}”
```

LOOPS - Visualize

```
-
  name: Create users
  hosts: localhost
  tasks:
   - user: name ‘{{ ???? }}’   state=present  uid= ‘{{ ? }}’
     loop:
       - name: joe
         uid: 1010
       - name: george
         uid: 1011
       - name: ravi
         uid: 1012
       - name: mani
         uid: 1013
       - name: kiran
         uid: 1014
       - name: jazlan
         uid: 1015
       - name: emaan
         uid: 1016
       - name: mazin
         uid: 1017
       - name: izaan
         uid: 1018
       - name: mike
```

```
-
  name: Create users
  hosts: localhost
  tasks:
   -  var:
       item:
         name: joe
         uid: 1010
      user: name=‘{{ item.name }}’ state=present uid=‘{{ item.uid

   -  var:
       item:
         name: george
         uid: 1011
      user: name=‘{{ item.name }}’ state=present uid=‘{{ item.uid

   -  var:
       item:
         name: ravi
         uid: 1012
      user: name=‘{{ item.name }}’ state=present uid=‘{{ item.uid
   -  var:
       item:
         name: mani
         uid: 1013
      user: name=‘{{ item.name }}’ state=present uid=‘{{ item.uid
```

LOOPS - Visualize

```yaml
-
  name: Create users
  hosts: localhost
  tasks:
   - user: name= '{{ item.name }}' state=present  uid='{{ item.uid }}'
     loop:
       - name: joe
         uid: 1010
       - name: george
         uid: 1011
       - name: ravi
         uid: 1012
       - name: mani
         uid: 1013
       - name: kiran
         uid: 1014
       - name: jazlan
         uid: 1015
       - name: emaan
         uid: 1016
       - name: mazin
         uid: 1017
       - name: izaan
         uid: 1018
       - name: mike
```

-

```yaml
-
  name: Create users
  hosts: localhost
  tasks:
   -  var:
       item:
         name: joe
         uid: 1010
      user: name='{{ item.name }}' state=present uid='{{ item.uid
   -  var:
       item:
         name: george
         uid: 1011
      user: name='{{ item.name }}' state=present uid='{{ item.uid
   -  var:
       item:
         name: ravi
         uid: 1012
      user: name='{{ item.name }}' state=present uid='{{ item.uid
   -  var:
       item:
         name: mani
         uid: 1013
      user: name='{{ item.name }}' state=present uid='{{ item.uid
```

With_*

```
-
  name: Create users
  hosts: localhost
  tasks:
   - user: name='{{ item }}'   state=present
     loop:
       - joe
       - george
       - ravi
       - mani
```

```
-
  name: Create users
  hosts: localhost
  tasks:
   - user: name='{{ item }}'   state=present
     with_items:
       - joe
       - george
       - ravi
       - mani
```

With_*

```
-
  name: Create users
  hosts: localhost
  tasks:
   - user: name='{{ item }}'   state=present
     with_items:
       - joe
       - george
       - ravi
       - mani
```

```
-
  name: View Config Files
  hosts: localhost
  tasks:
   - debug: var=item
     with_file:
       - "/etc/hosts"
       - "/etc/resolv.conf"
       - "/etc/ntp.conf"
```

```
-
  name: Get from multiple URLs
  hosts: localhost
  tasks:
   - debug: var=item
     with_url:
       - "https://site1.com/get-servers"
       - "https://site2.com/get-servers"
       - "https://site3.com/get-servers"
```

```
-
  name: Check multiple mongodbs
  hosts: localhost
  tasks:
   - debug: msg="DB={{ item.database }} PID={{ item.pid}}"
     with_mongodb:
       - database: dev
         connection_string: "mongodb://dev.mongo/"
       - database: prod
         connection_string: "mongodb://prod.mongo/"
```

With_*

with_items
with_file
with_url
with_mongodb

with_dict
with_etcd
with_env
with_filetree
With_ini
With_inventory_hostnames
With_k8s
With_manifold
With_nested
With_nios
With_openshift
With_password
With_pipe
With_rabbitmq

With_redis
With_sequence
With_skydive
With_subelements
With_template
With_together
With_varnames

KodeKloud

Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Ansible

Modules

modules

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

command

Executes a command on a remote node

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.

playbook.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
```

```
   - name: Copy file from source to destination
     copy: src=/source_file dest=/destination
```

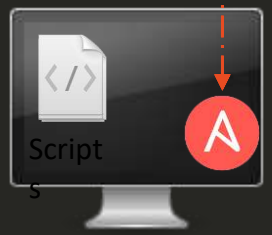
script

- Runs a local script on a remote node after transferring it

playbook.yml

```
- name: Play 1
  hosts: localhost
  tasks:
   - name: Run a script on remote server
     script: /some/local/script.sh -arg1 -arg2
```

1. Copy script to remote systems
2. Execute script on remote systems



Service

- Manage Services – Start, Stop, Restart

playbook.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

   - name: Start the nginx service
     service:
       name: nginx
       state: started
```

playbook.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.

lineinfile

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

/etc/resolv.conf

```
nameserver 10.1.250.1
nameserver 10.1.250.2
```

playbook.yml

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

/etc/resolv.conf

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

```
nameserver 10.1.250.10
```

script.sh

```
#Sample script

echo “nameserver 10.1.250.10” >> /etc/resolv.conf
```

/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
```

KodeKloud

Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Introduction to Ansible Plugins

Introduction

Challenges

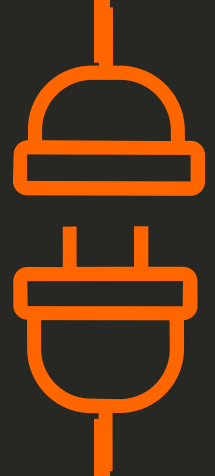
Inventory solution for real-time data

Require the ability to provision cloud resources with custom configurations

Dynamically configure

Solution

Ansible Plugins



What is a Plugin?

A piece of code that modifies the functionality of Ansible

Enhance various aspects of Ansible

Flexible and Powerful way to customize

Inventory

Modules

Callbacks

Inventory Plugin

Module Plugin

Action Plugin

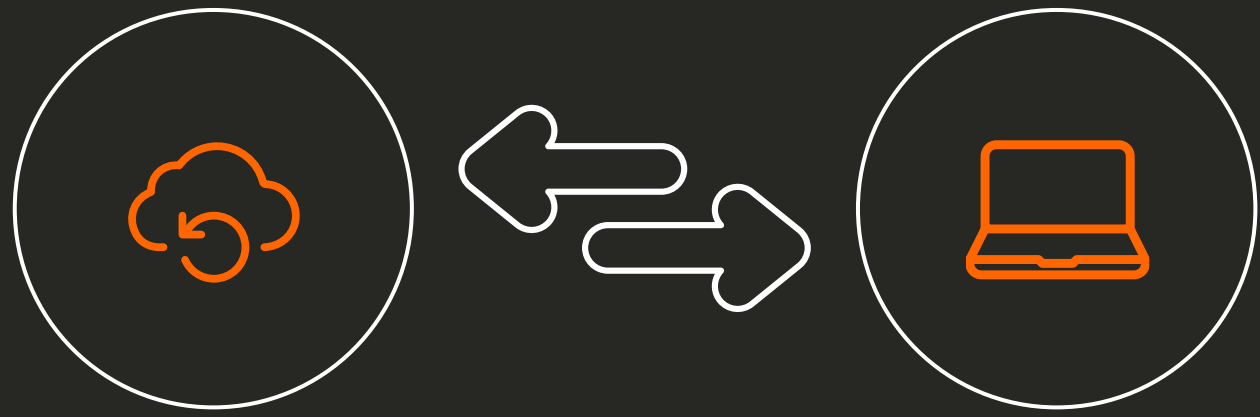
Callback Plugin

How Ansible Plugins Overcome the mentioned Challenges?

Dynamic Inventory Plugin

Module Plugin

Action Plugin

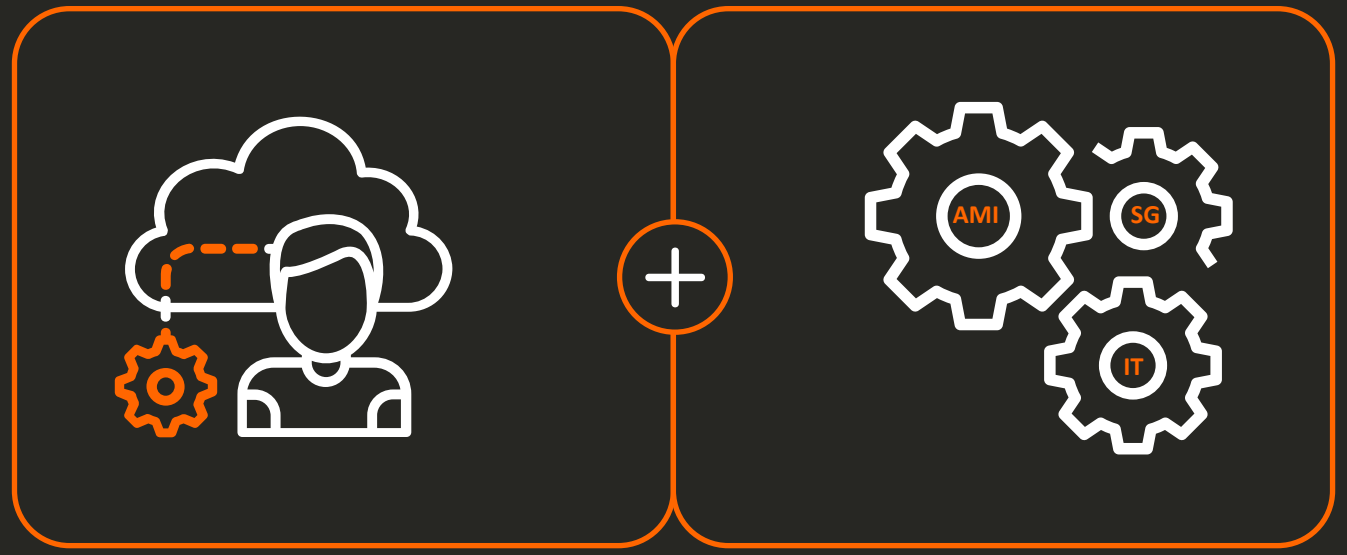


How Ansible Plugins Overcome the mentioned Challenges?

Dynamic Inventory Plugin

Module Plugin

Action Plugin

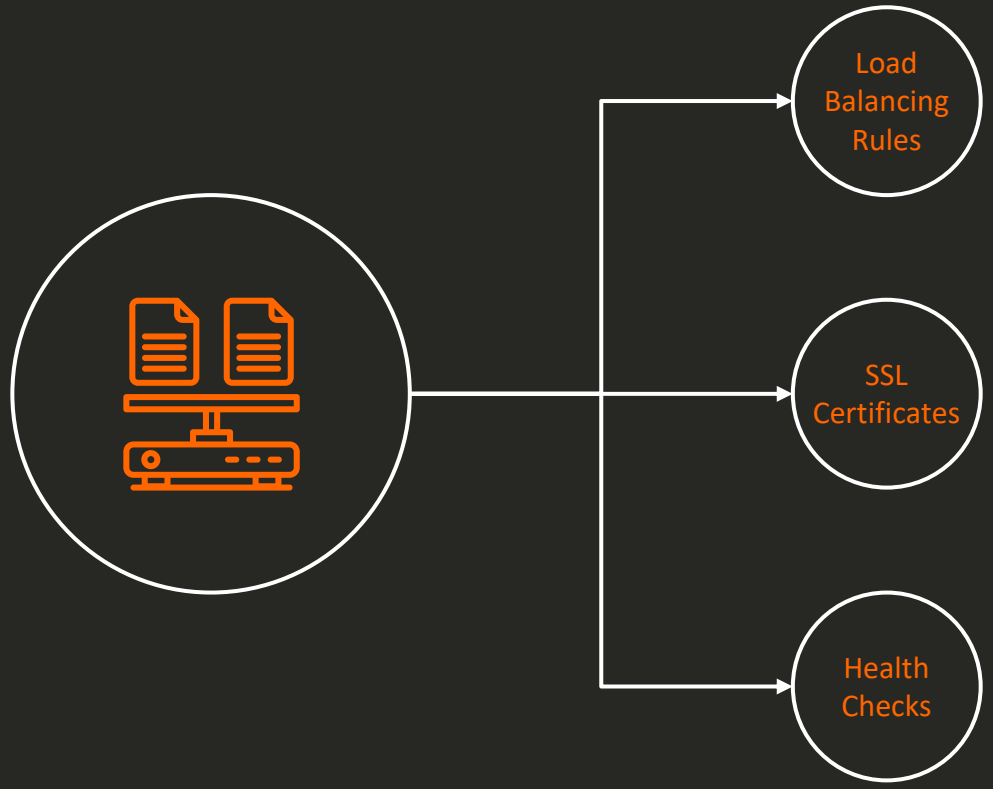


How Ansible Plugins Overcome the mentioned Challenges?

Dynamic Inventory Plugin

Module Plugin

Action Plugin



Other Plugins

- Lookup Plugins

- Filter Plugins

- Connection Plugins

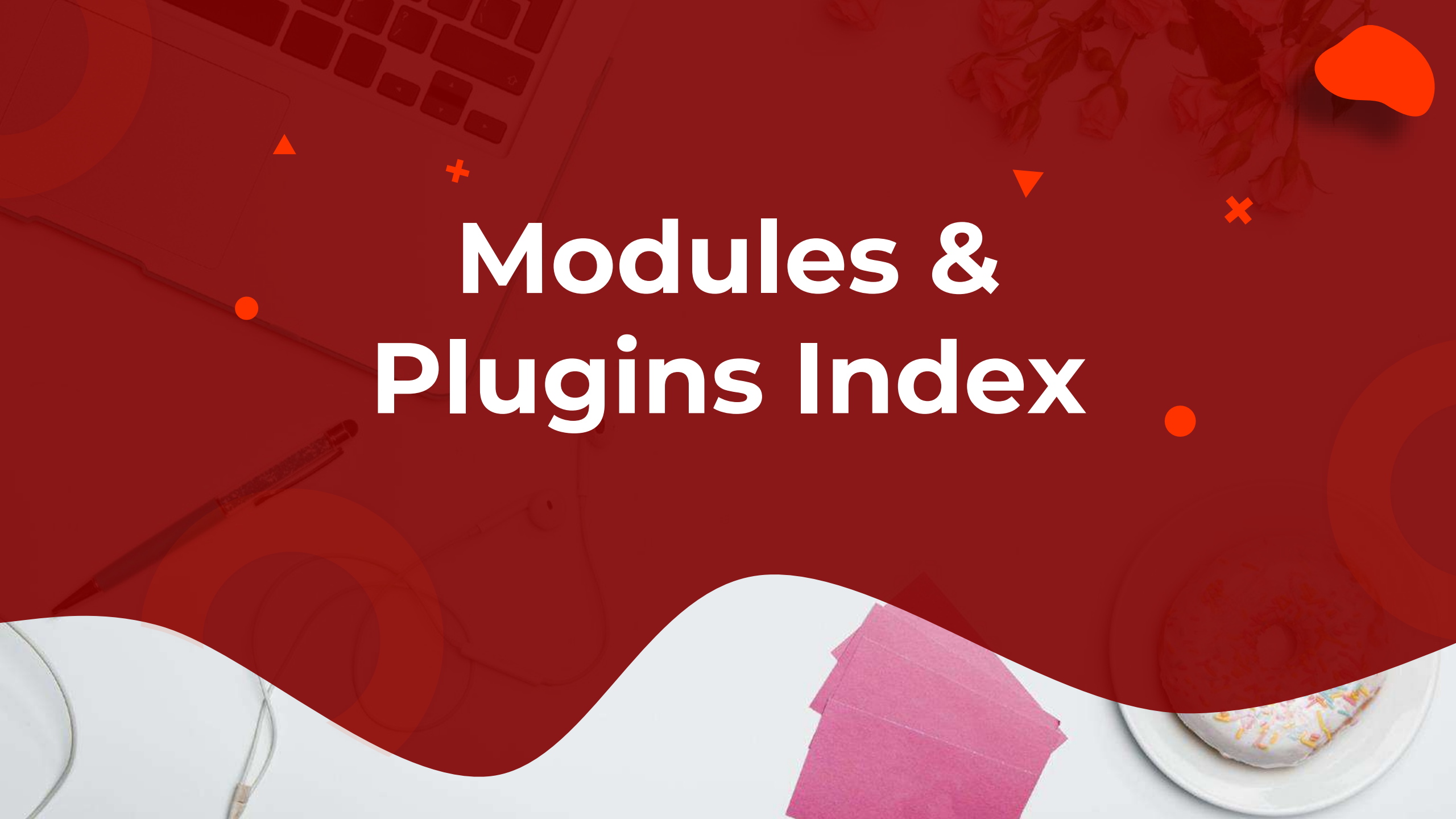
- Inventory Plugins

- Callback Plugins

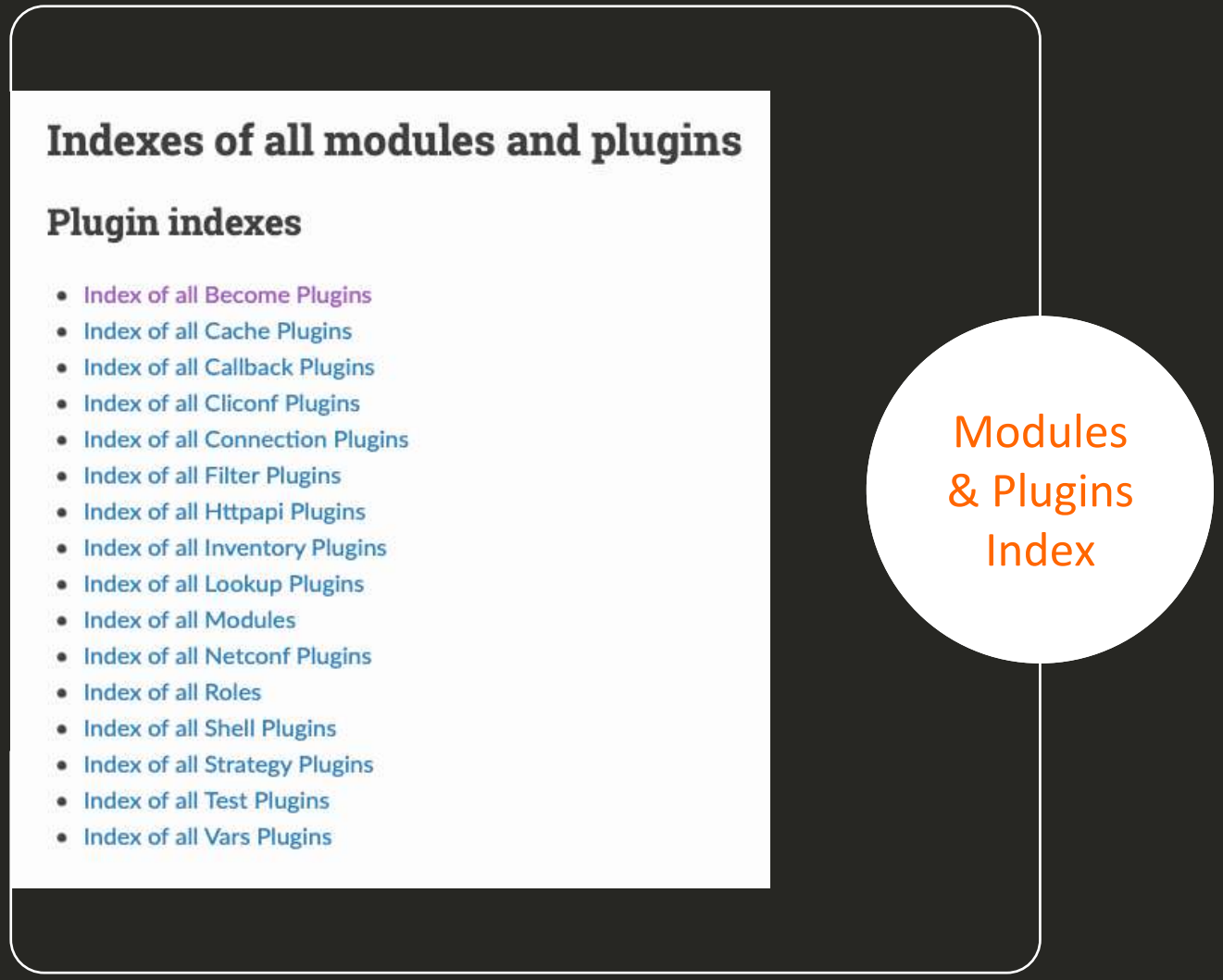
KodeKloud

Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Modules & Plugins Index



Introduction



Modules & Plugins Index

cisco.ios

Modules & Plugins Index

Comprehensive resource offering detailed information on Ansible's available modules and plugins

Navigate Ansible's extensive modules and plugins, helping you find the right tool for optimal automation

Without an index, tracking and understanding the numerous Ansible modules and plugins would be difficult

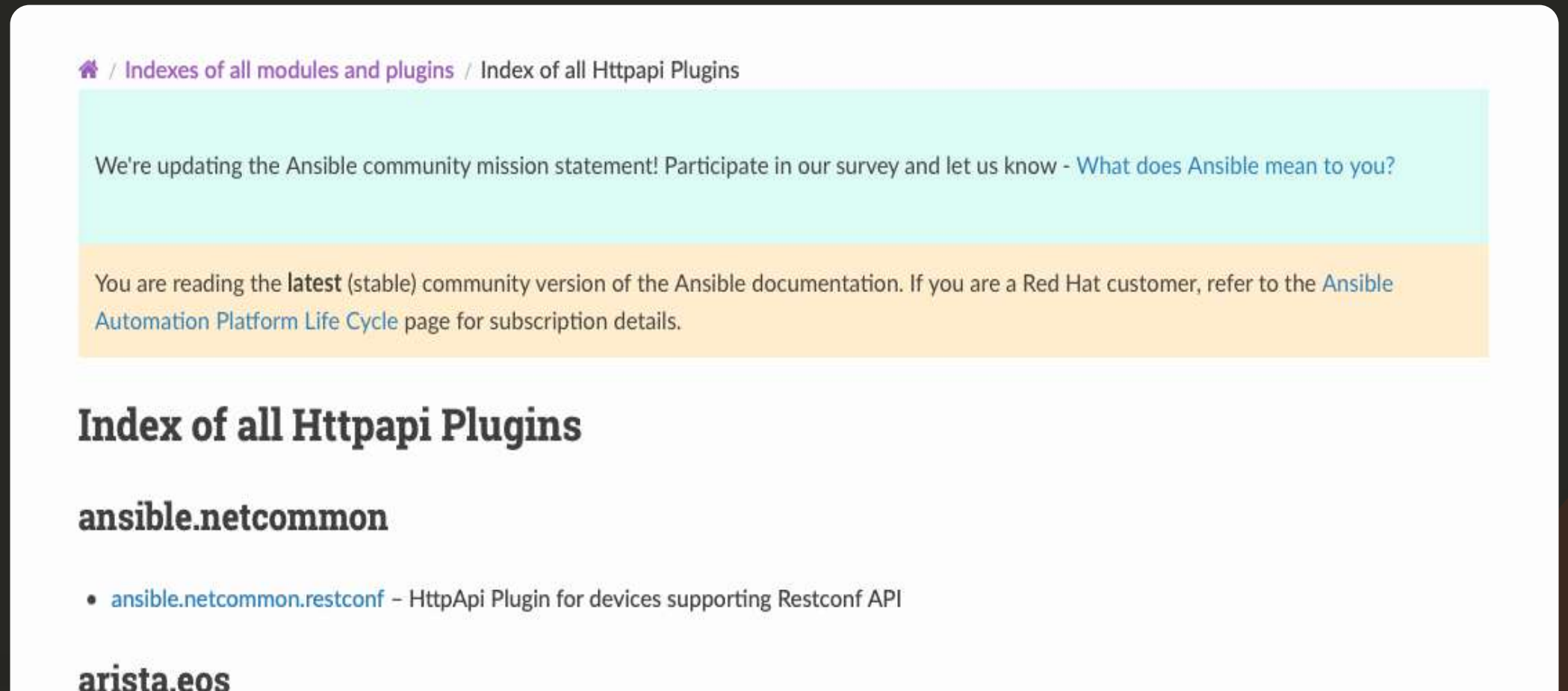
Key Features

Search & Filtering

Detailed Documentation

Version Compatibility

Community Contributions



Key Features

Search & Filtering

Detailed Documentation

Version Compatibility

Community Contributions

Index of all Modules

amazon.aws

- amazon.aws.autoscaling_group – Create or delete AWS AutoScaling Groups (ASGs)
- amazon.aws.autoscaling_group_info – Gather information about EC2 Auto Scaling Groups (ASGs) in AWS
- amazon.aws.aws_az_info – Gather information about availability zones in AWS
- amazon.aws.aws_caller_info – Get information about the user and account being used to make AWS calls
- amazon.aws.backup_plan – Manage AWS Backup Plans
- amazon.aws.backup_plan_info – Describe AWS Backup Plans
- amazon.aws.backup_restore_job_info – List information about backup restore jobs
- amazon.aws.backup_selection – Create, delete and modify AWS Backup selection
- amazon.aws.backup_selection_info – Describe AWS Backup Selections
- amazon.aws.backup_tag – Manage tags on backup plan, backup vault, recovery point
- amazon.aws.backup_tag_info – List tags on AWS Backup resources
- amazon.aws.backup_vault – Manage AWS Backup Vaults
- amazon.aws.backup_vault_info – Describe AWS Backup Vaults

Key Features

Search & Filtering | **Detailed Documentation** | Version Compatibility | Community Contributions

amazon.aws.aws_ec2 inventory – EC2 inventory source

Note

This inventory plugin is part of the amazon.aws collection (version 6.2.0).

You might already have this collection installed if you are using the `ansible` package. It is not included in `ansible-core`. To check whether it is installed, run `ansible-galaxy collection list`.

To install it, use: `ansible-galaxy collection install amazon.aws`. You need further requirements to be able to use this inventory plugin, see Requirements for details.

To use it in a playbook, specify: `amazon.aws.aws_ec2`.

- Synopsis
- Requirements
- Parameters
- Notes
- Examples

Key Features

Search & Filtering | Detailed Documentation | **Version Compatibility** | Community Contributions

Requirements

The below requirements are needed on the local controller node that executes this inventory.

- python >= 3.6
- boto3 >= 1.22.0
- botocore >= 1.25.0

Key Features

Search & Filtering | Detailed Documentation | Version Compatibility | **Community Contributions**

Authors

- Sloane Hertel (@s-hertel)

Hint

Configuration entries for each entry type have a low to high priority order. For example, a variable that is lower in the list will override a variable that is higher up.

Collection links

Issue Tracker | Repository (Sources) | Communication

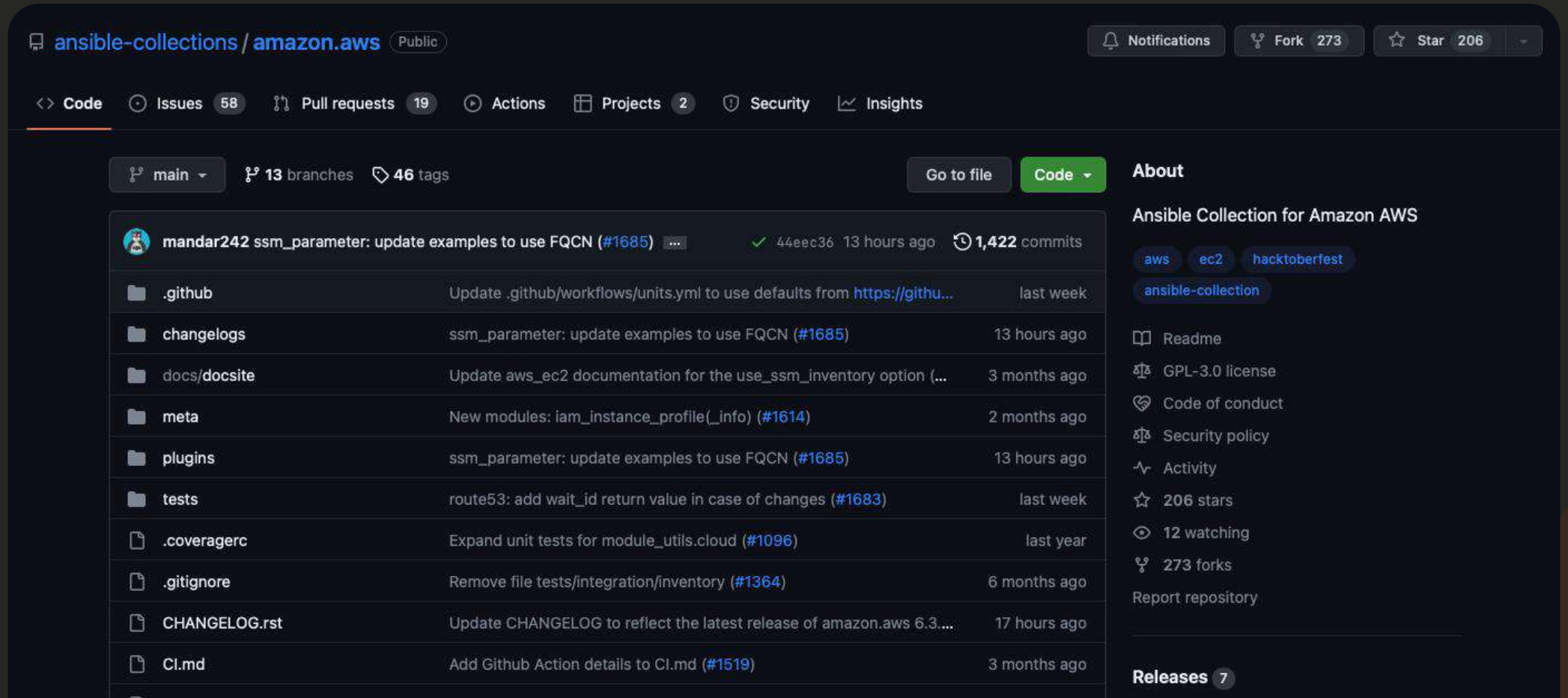
Key Features

Search & Filtering

Detailed Documentation

Version Compatibility

Community Contributions

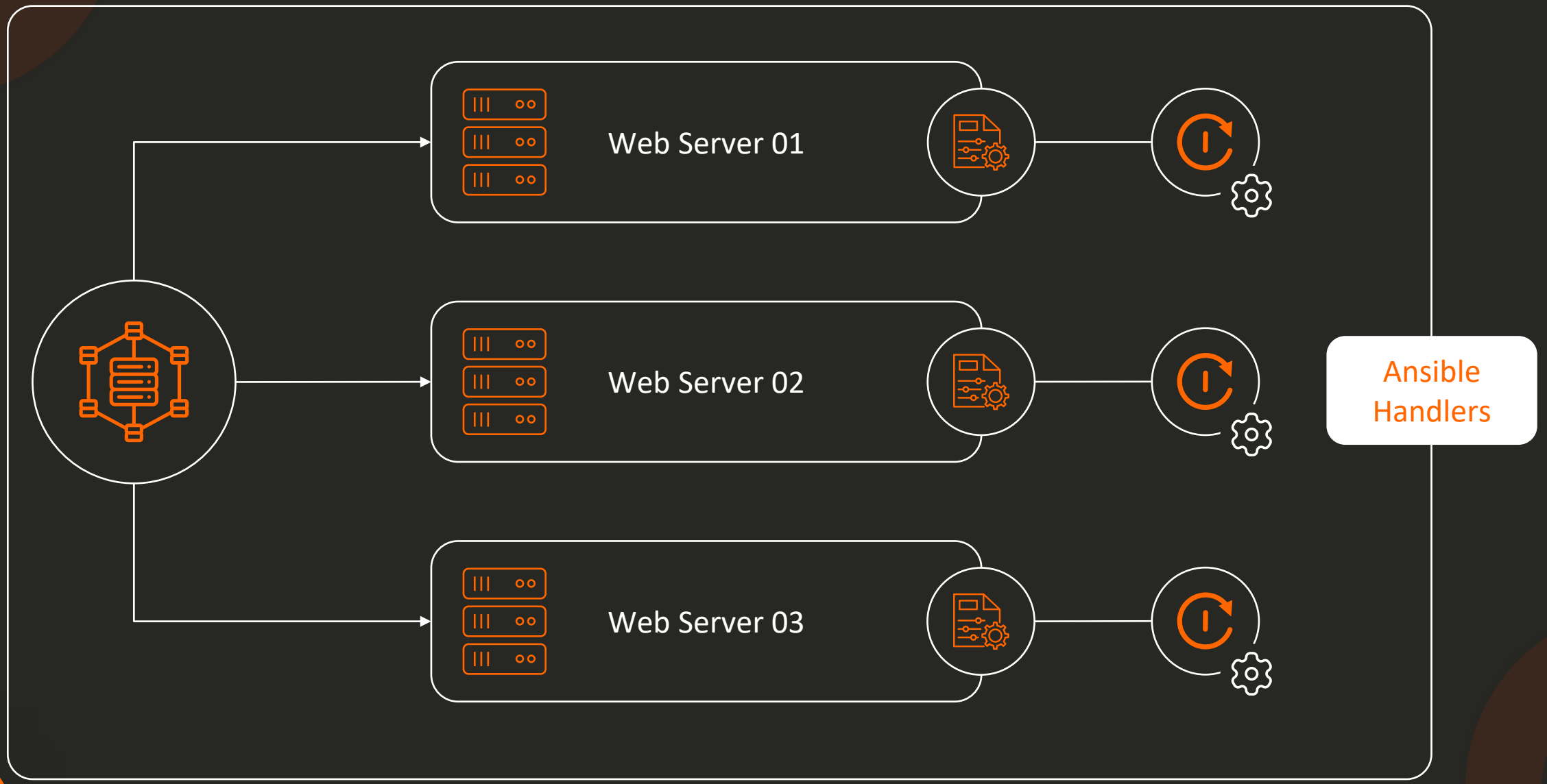


KodeKloud

Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Introduction to Handlers

Introduction



Ansible Handlers

Tasks triggered by events/notifications.

Defined in playbook, executed when notified by a task

Manage actions based on system state/configuration changes

Ansible Handlers

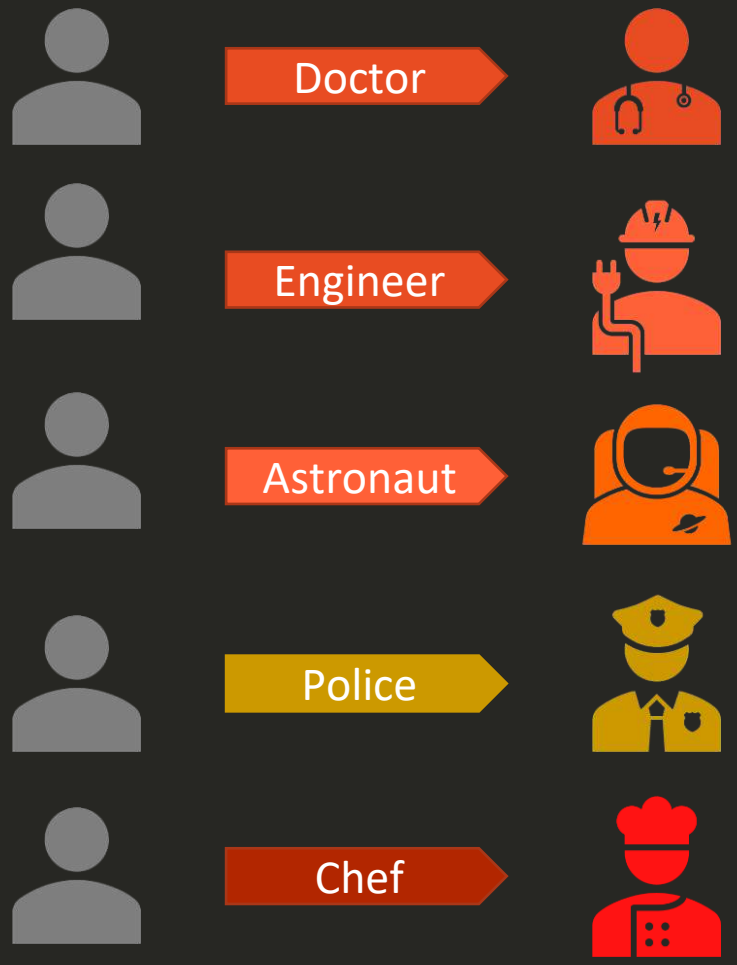
```
- name: Deploy Application
  hosts: application_servers
  tasks:
    - name: Copy Application Code
      copy:
        src: app_code/
        dest: /opt/application/
      notify: Restart Application Service

  handlers:
    - name: Restart Application Service
      service:
        name: application_service
        state: restarted
```

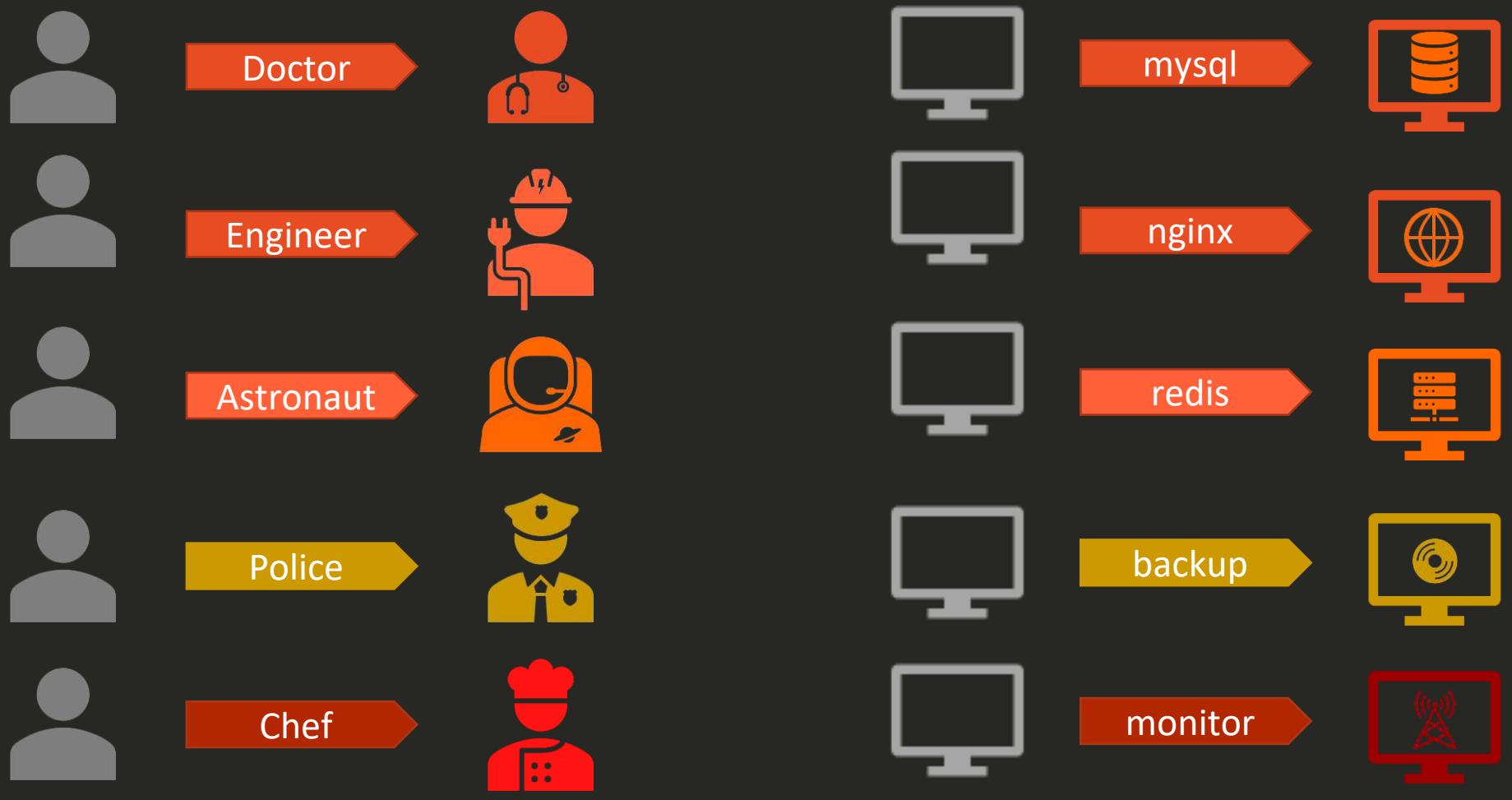
KodeKloud

Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Ansible

Roles


Doctor
Engineer
Astronaut
Police
Chef


Doctor
Engineer
Astronaut
Police
Chef
mysql
nginx
redis
backup
monitor

Doctor

- Go to medical school
- Earn medical degree
- Complete Residency Program
- Obtain License

Engineer

- Go to engineering school
- Earn bachelor's degree
- Gain field experience
- Gain postgraduate degree

mysql

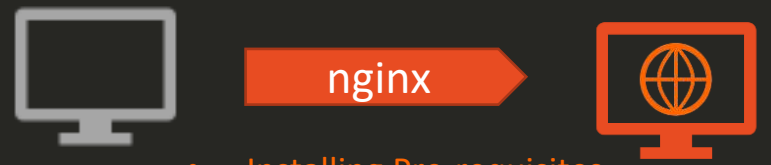
- Installing Pre-requisites
- Installing mysql packages
- Configuring mysql service
- Configuring database and users

nginx

- Installing Pre-requisites
- Installing nginx packages
- Configuring nginx service
- Configuring custom web pages

```
- name: Install and Configure MySQL
  hosts: db-server
  tasks:
     - name: Install Pre-Requisites
       yum: name=pre-req-packages state=present

     - name: Install MySQL Packages
       yum: name=mysql state=present

     - name: Start MySQL Service
       service: name=mysql state=started

     - name: Configure Database
       mysql_db: name=db1 state=present
```

- Installing Pre-requisites
- Installing mysql packages
- Configuring mysql service
- Configuring database and users



- Installing Pre-requisites
- Installing nginx packages
- Configuring nginx service
- Configuring custom web pages

Re-Use

mysql

- Installing Pre-requisites
- Installing mysql packages
- Configuring mysql service
- Configuring database and users

```
- name: Install and Configure MySQL
  hosts: db-server1……db-server100
  roles:
      - mysql
```

MySQL-Role

```
tasks:
    - name: Install Pre-Requisites
      yum: name=pre-req-packages state=present

    - name: Install MySQL Packages
      yum: name=mysql state=present

    - name: Start MySQL Service
      service: name=mysql state=started

    - name: Configure Database
      mysql_db: name=db1 state=present
```

Organize

Re-Use

mysql

- Installing Pre-requisites
- Installing mysql packages
- Configuring mysql service
- Configuring database and users

MySQL-Role

tasks

```
tasks:
   - name: Install Pre-Requisites
     yum: name=pre-req-packages state=present

   - name: Install MySQL Packages
     yum: name=mysql state=present

   - name: Start MySQL Service
     service: name=mysql state=started

   - name: Configure Database
     mysql_db: name=db1 state=present
```

vars

```
mysql_packages:
  - mysql
  - mysql-server
db_config:
   db_name: db1
```

handlers

defaults

```
mysql_user_name: root
mysql_user_password: root
```

templates

GALAXY

Login

Home

Search

Community

sensu

Deploy a full Sensu monitoring stack; including redis, RabbitMQ & the Uchiwa dashboard

sensu

alerting amqp cloud dashboard metrics monitoring rabbitmq redis sensu stack system web

build failing

5 / 5 Score 592903 Downloads

Last Imported: 13 hours ago

deploy

Ansible role to deploy scripting applications like PHP, Python, Ruby, etc. in a Capistrano style

ansistrano

cloud deploy deployment web

build passing

2.6 / 5 Score 146130 Downloads

Last Imported: 12 days ago

newrelic-infra

Role to install New Relic Infrastructure agent

newrelic

cloud ec2 monitoring system

build passing

4.7 / 5 Score 117312 Downloads

Last Imported: 12 days ago

aws-inspector

AWS Inspector installation for Linux.

geerlingguy

aws awsagent cloud inspector security system

build passing

5 / 5 Score 116745 Downloads

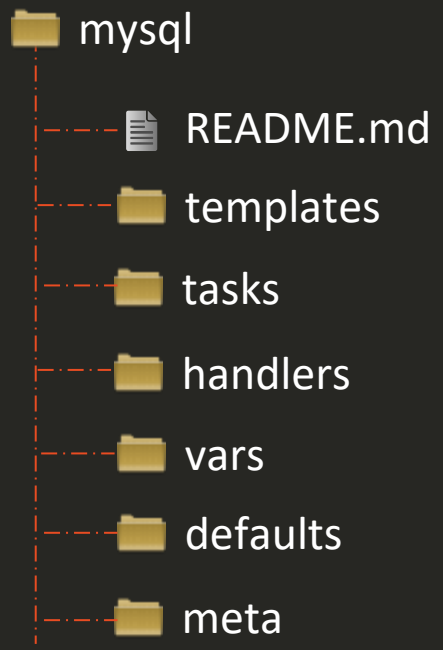
Last Imported: 11 hours ago

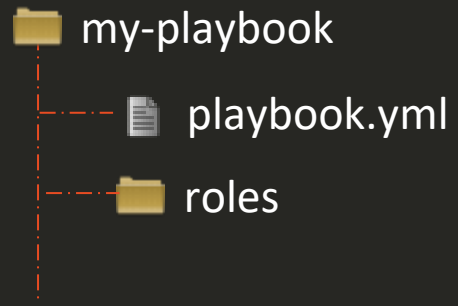
Organize

Re-Use

Share

```
$ ansible-galaxy init mysql
```





playbook.yml

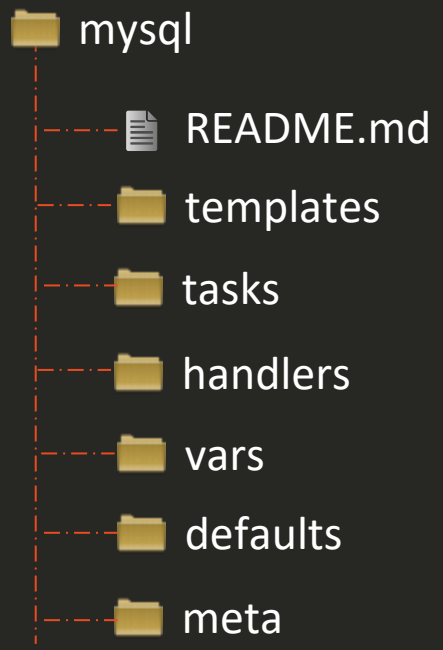
```yaml
- name: Install and Configure MySQL
  hosts: db-server
  roles:
      - mysql
```

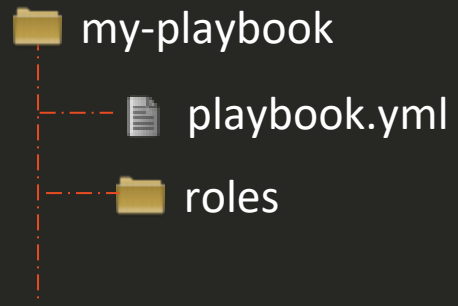
Organize

Re-Use

Share

```
$ ansible-galaxy init mysql
```





playbook.yml

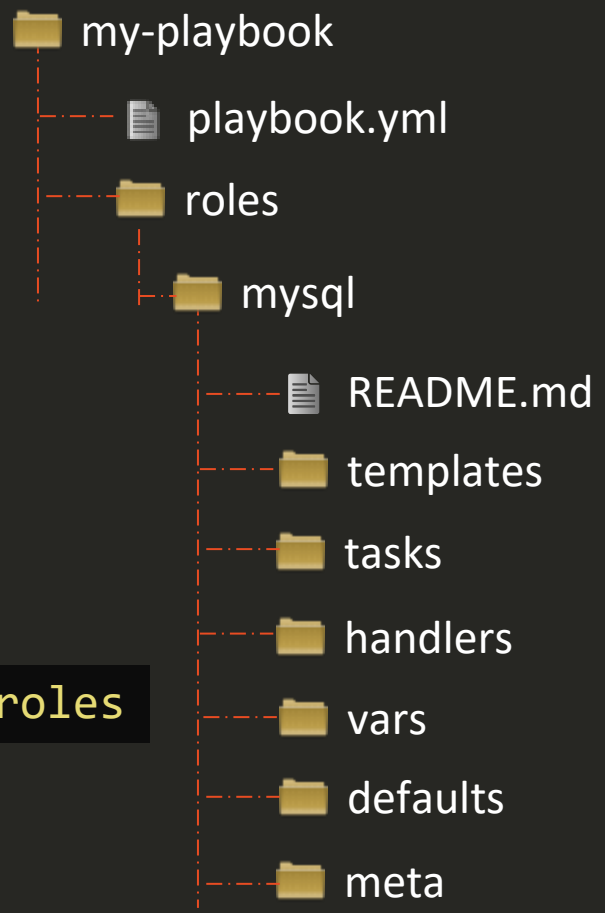
```yaml
- name: Install and Configure MySQL
  hosts: db-server
  roles:
      - mysql
```

Organize

Re-Use

Share

```
$ ansible-galaxy init mysql
```



/etc/ansible/ansible.cfg

```
roles_path    = /etc/ansible/roles
```

playbook.yml

```yaml
- name: Install and Configure MySQL
  hosts: db-server
  roles:
      - mysql
```

Organize

Re-Use

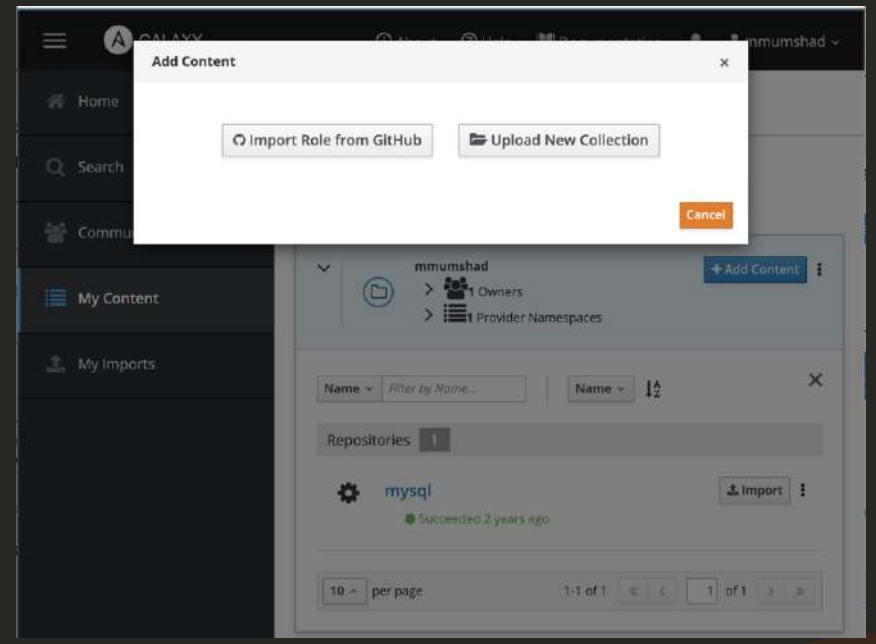
Share

```
$ ansible-galaxy init mysql
```

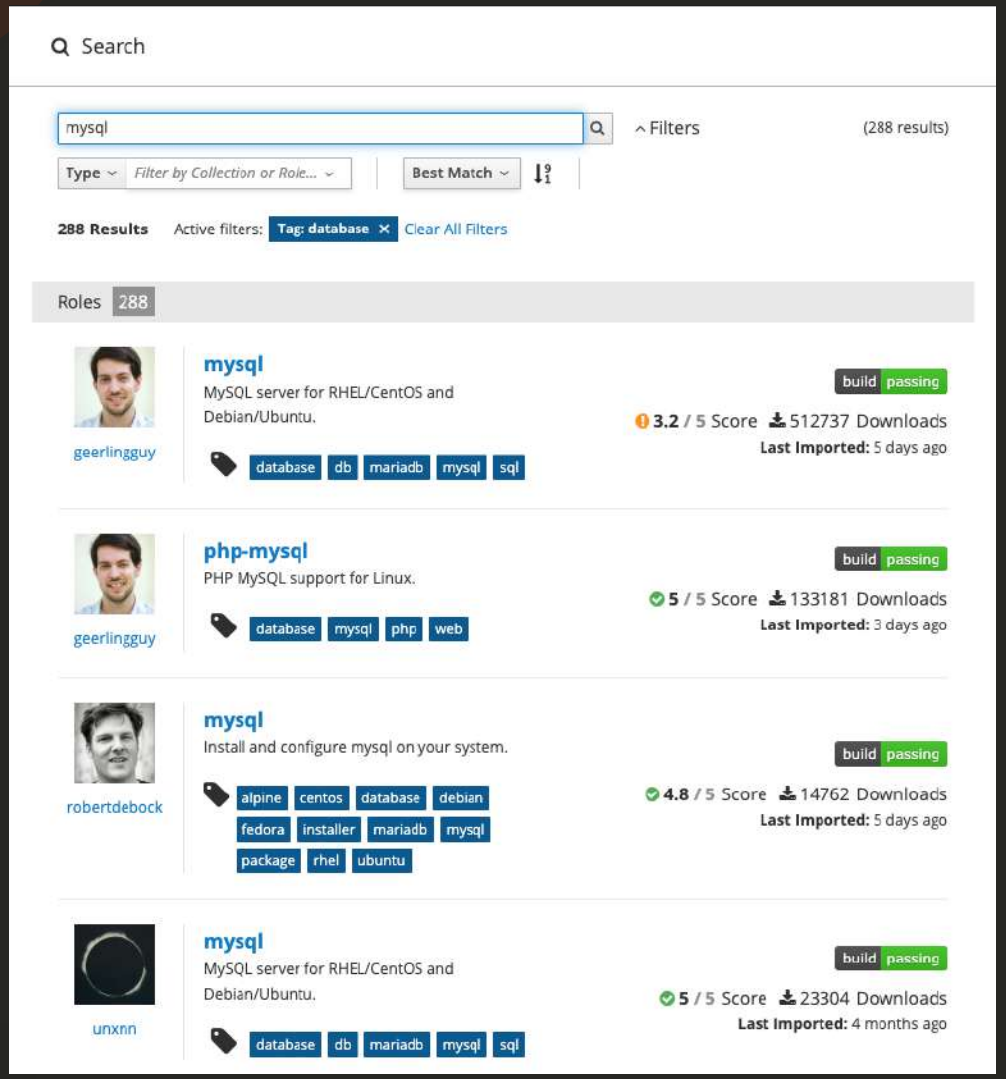
- my-playbook
  - playbook.yml
  - roles
    - mysql
      - README.md
      - templates
      - tasks
      - handlers
      - vars
      - defaults
      - meta

playbook.yml

```yaml
- name: Install and Configure MySQL
  hosts: db-server
  roles:
    - mysql
```



Find Roles




```
$ ansible-galaxy search mysql
```

Found 1126 roles matching your search. Showing first 1000.

Name	Description
0utsider.ansible_zabbix_agent	Installing and maintaining zabbix-agent for
1mr.unattended	install and configure unattended upgrade
1nfinitum.mysql	Simply installs MySQL 5.7 on Xenial.
4linuxdevops.mysql-server	Instalacao e Configuracao do servidor MySQL
5KYDEV0P5.skydevops-mysql	Install and configure MySQL Database
AAbouZaid.yourls	Manage Yourls, a URL shortener web app.
AAROC.AAROC_fg-db	your description
aaronpederson.ansible-autodeploy	Simple deployment tool with hooks
abednarik.mysqld-exporter	Install and configure mysqld_exporter
abelboldu.openstack-glance	
abelboldu.openstack-keystone	
abelboldu.openstack-neutron-controller	OpenStack Neutron controller node
abelboldu.openstack-nova-controller	OpenStack Nova controller node
achaussier.mysql-backup	configure mysql-backup with xtrabackup and
achaussier.mysql-server	Install mysql-server package
achilleskal.ansible_mysql8	your description
adarnimrod.mysql	Provision a MySQL server

Use Role

```
$ ansible-galaxy install geerlingguy.mysql
```

```
- downloading role 'mysql', owned by geerlingguy
- downloading role from https://github.com/geerlingguy/ansible-role-mysql/archive/2.9.5.tar.gz
- extracting geerlingguy.mysql to /etc/ansible/roles/geerlingguy.mysql
- geerlingguy.mysql (2.9.5) was installed successfully
```

playbook.yml

```yaml
-
  name: Install and Configure MySQL
  hosts: db-server
  roles:
    - geerlingguy.mysql
```

```yaml
-
  name: Install and Configure MySQL
  hosts: db-server
  roles:
    - role: geerlingguy.mysql
      become: yes
      vars:
        mysql_user_name: db-user
```

Use Role

Playbook-all-in-one.yml

```yaml
-
  name: Install and Configure MySQL
  hosts: db-and-webserver
  roles:
    - geerlingguy.mysql
    - nginx
```

Playbook-distributed.yml

```yaml
-
  name: Install and Configure MySQL
  hosts: db-server
  roles:
    - geerlingguy.mysql

-
  name: Install and Configure Web Server
  hosts: web-server
  roles:
    - nginx
```

List Roles

```
$ ansible-galaxy list
```

```
- geerlingguy.mysql
- kodekloud1.mysql
```

```
$ ansible-config dump | grep ROLE
```

```
EFAULT_PRIVATE_ROLE_VARS(default) = False
DEFAULT_ROLES_PATH(default) = [u'/root/.ansible/roles', u'/usr/share/ansible/roles', u'/etc/ansible/roles']
GALAXY_ROLE_SKELETON(default) = None
GALAXY_ROLE_SKELETON_IGNORE(default) = ['^.git$', '^.*/.git_keep$']
```

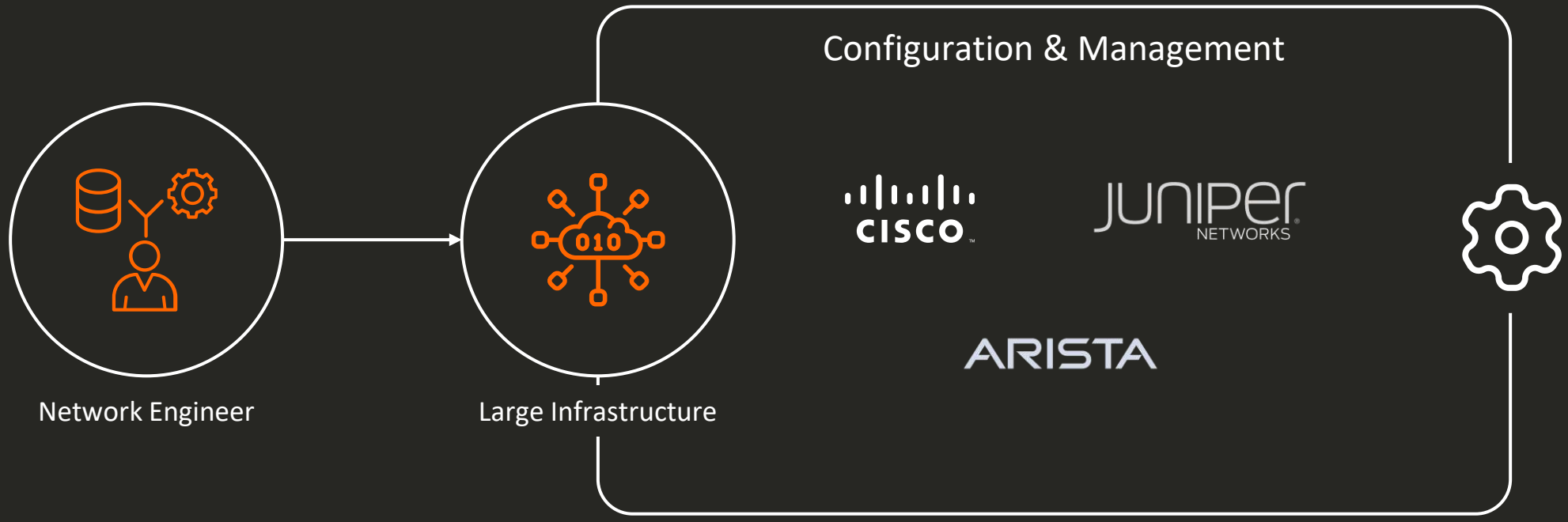
```
$ ansible-galaxy install geerlingguy.mysql -p ./roles
```

KodeKloud

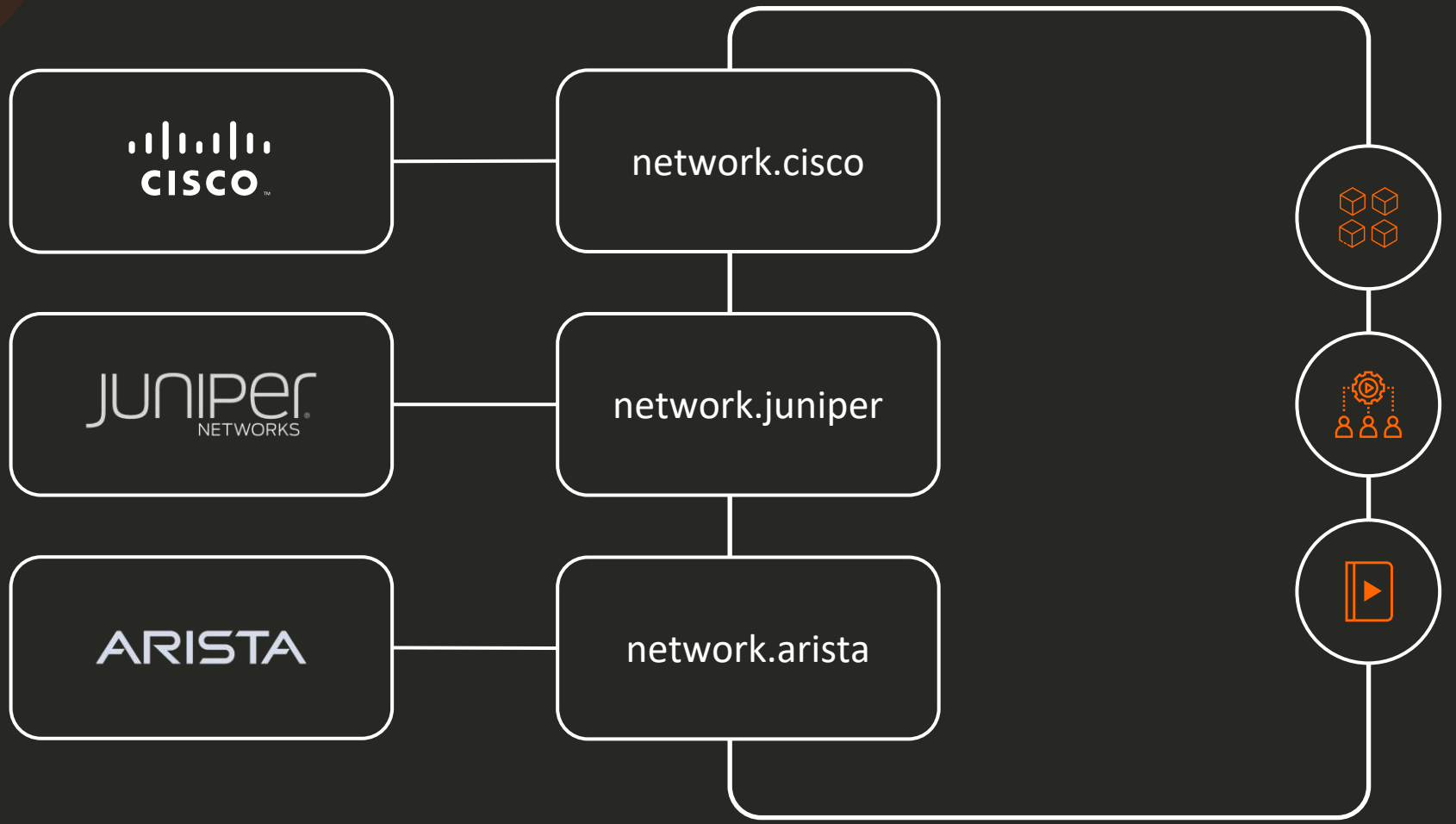
Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Ansible Collections

Introduction



Introduction



Example

```
$ ansible-galaxy collection install network.cisco
```

What Are Ansible Collections?

Package and distribute modules, roles, plugins, etc..

Self-contained

Community and Vendor-created

Benefits

Expanded Functionality

Modularity and Reusability

Simplified Distribution and Management

```
$ ansible-galaxy collection install amazon.aws
```

```
---
- hosts: localhost
  collections:
    - amazon.aws

  tasks:
    - name: Create an S3 bucket
      aws_s3_bucket:
        name: my-bucket
        region: us-west-1
```

Benefits

Expanded Functionality

Modularity and Reusability

Simplified Distribution and Management

```
my_collection/
├── docs/
├── galaxy.yml
├── plugins/
│   └── modules/
│       └── my_custom_module.py
├── README.md
└── roles/
    └── my_custom_role/
        └── tasks/
            └── main.yml
```

Benefits

Expanded Functionality

Modularity and Reusability

Simplified Distribution and Management

```
---
- hosts: localhost
  collections:
    - my_namespace.my_collection

  roles:
    - my_custom_role

  tasks:
    - name: Use custom module
      my_custom_module:
        param: value
```

Benefits

Expanded Functionality

Modularity and Reusability

Simplified Distribution and Management

requirements.yml

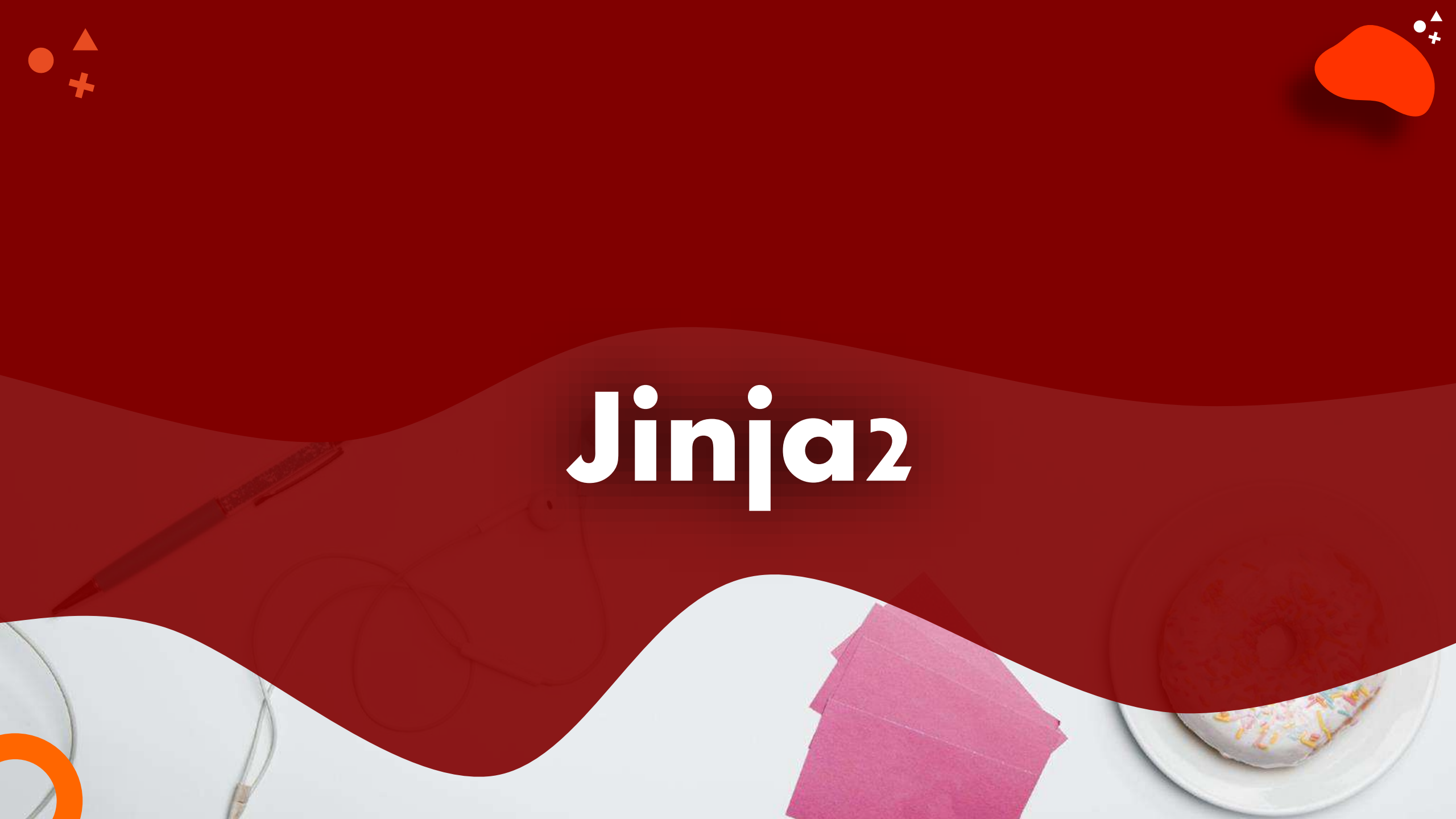
```
---
collections:
  - name: amazon.aws
    version: "1.5.0"
  - name: community.mysql
    src: https://github.com/ansible-collections/community.mysql
    version: "1.2.1"
```

```
$ ansible-galaxy collection install -r requirements.yml
```

KodeKloud

Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Jinja2



Templating?

Template

```
Hi    ,


I am glad to invite you along with your family
members –                , to attend the party
arranged by us on the completion of 10 successful
years of our company. We would be happy to mark your
presence along with family at the party and would
love to celebrate the success together.


Sincerely,

Andrews,
CEO
```

Variables

Sam

Mary and Adam

Anil

Achu and George

Michelle

Sarah

Shabab

Aliah and Medina

Templating Engine

Hi Sam,

I am glad to invite you along with your family members – Mary and Adam, to attend the party arranged by us on the completion of 10 successful years of our company. We would be happy to mark your presence along with family at the party and would love to celebrate the success together.

Sincerely,

Andrews,
CEO

Hi Anil ,

I am glad to invite you along with your family members – Achu and George, to attend the party arranged by us on the completion of 10 successful years of our company. We would be happy to mark your presence along with family at the party and would love to celebrate the success together.

Sincerely,

Andrews,
CEO

Hi **Michelle** ,

I am glad to invite you along with your family members – Sarah, to attend the party arranged by us on the completion of 10 successful years of our company. We would be happy to mark your presence along with family at the party and would love to celebrate the success together.

Hi **Shabab** ,

I am glad to invite you along with your family members – **Aliah and Medina**, to attend the party arranged by us on the completion of 10 successful years of our company. We would be happy to mark your presence along with family at the party and would love to celebrate the success together.

HTML

Template

```
<!DOCTYPE html>
<html>
  <head>
    <title>{{ title }}</title>
  </head>
  <body>
     {{ msg }}
  </body>
</html>
```

Variables

```
title: Our Site

msg: Welcome!
```

Outcome

```
<!DOCTYPE html>
<html>
  <head>
    <title>Our Site</title>
  </head>
  <body>
     Welcome!
  </body>
</html>
```

ANSIBLE

Template

```
- hosts: web1
  tasks:
    -
      file:
        path: {{ file }}
        state: touch
```

Variables

```
file: /tmp/1.txt
```

Outcome

```
- hosts: web1
  tasks:
    -
      file:
        path: /tmp/1.txt
        state: touch
```

Template

```
[mysqld]
innodb-buffer-pool-size={{ pool_size }}
datadir={{ datadir }}
user={{ mysql_user }}
symbolic-links={{ link_id }}
port={{ mysql_port }}
```

Variables

```
pool_size: 5242880
datadir: /var/lib/mysql
mysql_user: mysql
link_id: 0
mysql_port: 3306
```

Outcome

```
[mysqld]
innodb-buffer-pool-size=5242880
datadir=/var/lib/mysql
user=mysql
symbolic-links=0
port=3306
```

jinja2

Project Links

Donate to Pallets
Jinja Website
PyPI releases
Source Code
Issue Tracker

Quick search

Go

Jinja

Jinja is a modern and designer-friendly templating language for Python, modelled after Django's templates. It is fast, widely used and secure with the optional sandboxed template execution environment:

```
<title>{% block title %}{% endblock %}</title>
<ul>
{% for user in users %}
  <li><a href="{{ user.url }}">{{ user.username }}</a></li>
{% endfor %}
</ul>
```

Features:

- sandboxed execution
- powerful automatic HTML escaping system for XSS prevention
- template inheritance
- compiles down to the optimal python code just in time
- optional ahead-of-time template compilation
- easy to debug. Line numbers of exceptions directly point to the correct line in the template.
- configurable syntax

Contents:

String manipulation

- 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

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
{{  [“The”, “name”, “is”, “Bond”] | join(“ “) }}  => The name is Bond
```

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

Loops

```
{% for number in [0,1,2,3,4] %}
{{ number }}
{% endfor %}
```

```
0
1
2
3
4
```

Conditions

```
{% for number in [0,1,2,3,4] %}

{% if number == 2 %}
    {{ number }}
{% endif %}

{% endfor %}
```

```
2
```

Ansible

Jinja2 in Ansible

Ansible Filters

abs()	float()	lower()	round()	tojson()
attr()	forceescape()	map()	safe()	trim()
batch()	format()	max()	select()	truncate()
capitalize()	groupby()	min()	selectattr()	unique()
center()	indent()	pprint()	slice()	upper()
default()	int()	random()	sort()	urlencode()
dictsort()	join()	reject()	string()	urlize()
escape()	last()	rejectattr()	striptags()	wordcount()
filesizeformat()	length()	replace()	sum()	wordwrap()
first()	list()	reverse()	title()	xmlattr()

b64decode()	basename()	combine()
b64encode()	dirname()	extract()
to_uuid()	expanduser()	flatten()
to_json()	expandvars()	dict2items()
to_nice_json()	realpath()	items2dict()
from_json()	relpath()	subelements()
to_yaml()	splitext()	random_mac()
to_nice_yaml()	win_basename()	rejectattr()
from_yaml()	win_dirnameh()	comment()
from_yaml_all()	win_splitdrive()	mandatory()

Filters - 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”
```

Jinja2 in Playbooks

/etc/ansible/hosts

```
web1 ansible_host=172.20.1.100 dns_server=10.5.5.4
web2 ansible_host=172.20.1.101 dns_server=10.5.5.4
web3 ansible_host=172.20.1.102 dns_server=10.5.5.4
```

```
---
- name: Update dns server
  hosts: all
  tasks:
  - nsupdate:
      server: '{{ dns_server }}'
```

```
---
- name: Update dns server
  hosts: all
  tasks:
  - nsupdate:
      server: 10.5.5.4
```

KodeKloud

Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Ansible

Templates

Templates

/etc/ansible/hosts

```
[web_servers]
web1 ansible_host=172.20.1.100
web2 ansible_host=172.20.1.101
web3 ansible_host=172.20.1.102
```

playbook.yml

```
-
  hosts: web_servers
  tasks:
    - name: Copy index.html to remote servers
      copy:
        src: index.html
        dest: /var/www/nginx-default/index.html
```

index.html

```
<!DOCTYPE html>
<html>
<body>

This is a Web Server

</body>
</html>
```

Templates

/etc/ansible/hosts

```
[web_servers]
web1 ansible_host=1
web2 ansible_host=1
```

This is a Web Server

This is web1 server

This is web2 server

This is web3 server

```
src: index.
dest: /var/
```

Templates

/etc/ansible/hosts

```
[web_servers]
web1 ansible_host=172.20.1.100
web2 ansible_host=172.20.1.101
web3 ansible_host=172.20.1.102
```

playbook.yml

```
-
  hosts: web_servers
  tasks:
    - name: Copy index.html to remote servers
      copy:
        src: index.html
        dest: /var/www/nginx-default/index.html
```

index.html

```
<!DOCTYPE html>
<html>
<body>

This is a Web Server

</body>
</ht
```

web1

index.html

```
<!DOCTYPE html>
<html>
<body>

This is web1 Server

</body>
</html>
```

web2

index.html

```
<!DOCTYPE html>
<html>
<body>

This is web2 Server

</body>
</html>
```

web3

index.html

```
<!DOCTYPE html>
<html>
<body>

This is web3 Server

</body>
</html>
```

Templates

/etc/ansible/hosts

```
[web_servers]
web1 ansible_host=172.20.1.100
web2 ansible_host=172.20.1.101
web3 ansible_host=172.20.1.102
```

playbook.yml

```
-
  hosts: web_servers
  tasks:
    - name: Copy index.html to remote servers
      copy:
        src: index.html
        dest: /var/www/nginx-default/index.html
```

index.html

```
<!DOCTYPE html>
<html>
<body>

This is a Web Server

</body>
</html>
```

web3

index.html

```
<!DOCTYPE html>
<html>
<body>

This is {{ name }} Server

</body>
</html>
```

Templates

/etc/ansible/hosts

```
[web_servers]
web1 ansible_host=172.20.1.100
web2 ansible_host=172.20.1.101
web3 ansible_host=172.20.1.102
```

playbook.yml

```
-
  hosts: web_servers
  tasks:
    - name: Copy index.html to remote servers
      template:
        src: index.html.j2
        dest: /var/www/nginx-default/index.html
```

index.html.j2

```
<!DOCTYPE html>
<html>
<body>

This is {{ inventory_hostname }} Server

</body>
</html>
```

```
[web_servers]
web1 ansible_host=172.20.1.100
web2 ansible_host=172.20.1.101
web3 ansible_host=172.20.1.102
```

Variable Interpolation

Gather Facts

Execute Playbook

Create file from Template

Copy to target host

Create Subprocess

```
inventory_hostname=web1
ansible_host=172.20.1.100

ansible_facts=<Host Facts>
```

playbook.yml

```
-
  hosts: web_servers
  tasks:
    - name: Copy index.html to remote servers
      copy:
        src: index.html
        dest: /var/www/nginx-default/index.html
```

```
inventory_hostname=web2
ansible_host=172.20.1.101

ansible_facts=<Host Facts>
```

playbook.yml

```
-
  hosts: web_servers
  tasks:
    - name: Copy index.html to remote servers
      copy:
        src: index.html
        dest: /var/www/nginx-default/index.html
```

```
inventory_hostname=web3
ansible_host=172.20.1.102

ansible_facts=<Host Facts>
```

playbook.yml

```
-
  hosts: web_servers
  tasks:
    - name: Copy index.html to remote servers
      copy:
        src: index.html
        dest: /var/www/nginx-default/index.html
```

web1

index.html

```
<!DOCTYPE html>
<html>
<body>

This is web1 Server

</body>
</html>
```

web2

index.html

```
<!DOCTYPE html>
<html>
<body>

This is web2 Server

</body>
</html>
```

web3

index.html

```
<!DOCTYPE html>
<html>
<body>

This is web3 Server

</body>
</html>
```

Template Examples

nginx.conf.j2

```
server {
    location / {
        fastcgi_pass  {{host}}:{{port}};
        fastcgi_param QUERY_STRING    $query_string;
    }

    location ~ \  gif|jpg|png $ {
        root {{ image_path }};
    }
}
```

nginx.conf

```
server {
    location / {
        fastcgi_pass  localhost:9000
        fastcgi_param QUERY_STRING    $query_string;
    }

    location ~ \  gif|jpg|png $ {
        root /data/images;
    }
}
```

Template Examples

redis.conf.j2

```
bind {{ ip_address }}

protected-mode yes

port {{ redis_port | default('6379') }}

tcp-backlog 511

# Unix socket.
timeout 0

# TCP keepalive.
tcp-keepalive {{tcp_keepalive | default('300') }}

daemonize no

supervised no
```

redis.conf

```
bind 192.168.1.100

protected-mode yes

port 6379

tcp-backlog 511

# Unix socket.
timeout 0

# TCP keepalive.
tcp-keepalive 300

daemonize no

supervised no
```

Template Examples

/etc/resolv.conf.j2

```
{% for name_server in name_servers %}
nameserver    name_server
{% endfor %}
```

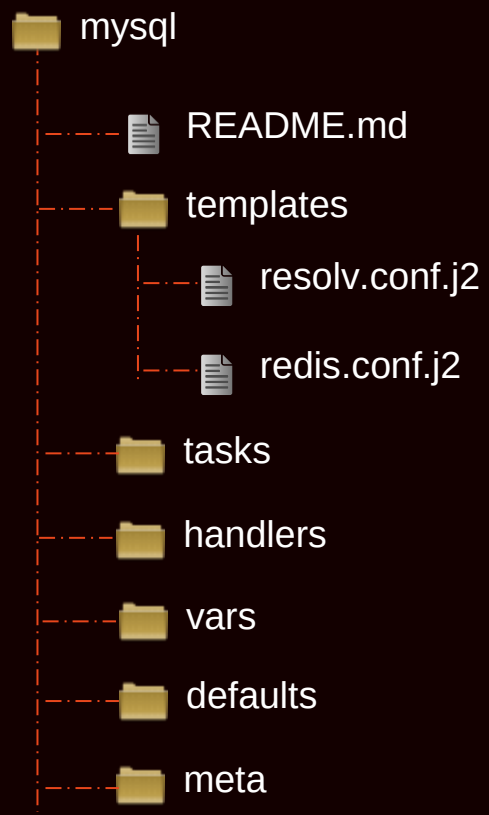
/etc/resolv.conf

```
nameserver 10.1.1.2
nameserver 10.1.1.3
nameserver 8.8.8.8
```

variable

```
name_servers:
  - 10.1.1.2
  - 10.1.1.3
  - 8.8.8.8
```

Templates in Roles



KodeKloud

Check out our full course on Ansible for the Absolute Beginners here: https://kode.wiki/3sufvat

Ansible

Install

Control Node

Redhat or CentOS –
```
$ sudo yum install ansible
```

Fedora –
```
$ sudo dnf install ansible
```

Ubuntu –
```
$ sudo apt-get install ansible
```

PIP –
```
$ sudo pip install ansible
```

Ansible Control Machine

- Playbooks
- Inventory
- Modules

Control Machine - Linux Only

Additional Options:

- Install from source on GIT
- Build RPM yourself

https://docs.ansible.com/ansible/latest/installation_guide/

Install Control Node on Redhat or CentOS

Redhat or CentOS – `$ sudo yum install ansible`

Install via PIP

Install pip if not present

```
$ sudo yum install epel-release
```

```
$ sudo yum install python-pip
```

Install Ansible using pip

```
$ sudo pip install ansible
```

Upgrade Ansible using pip

```
$ sudo pip install --upgrade ansible
```

Install Specific Version of Ansible using pip

```
$ sudo pip install ansible==2.4
```