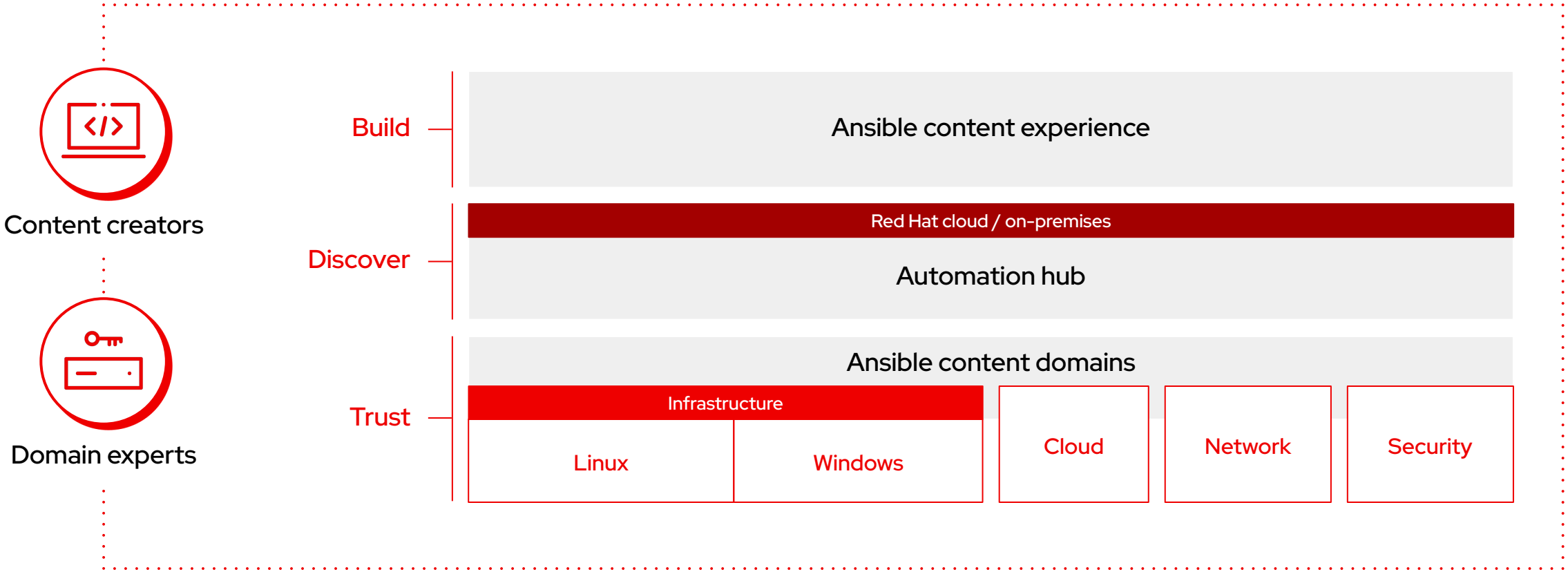
Section 1
The Ansible Basics

Red Hat
Ansible Automation Platform

Red Hat

Create

The automation lifecycle



Ansible playbooks

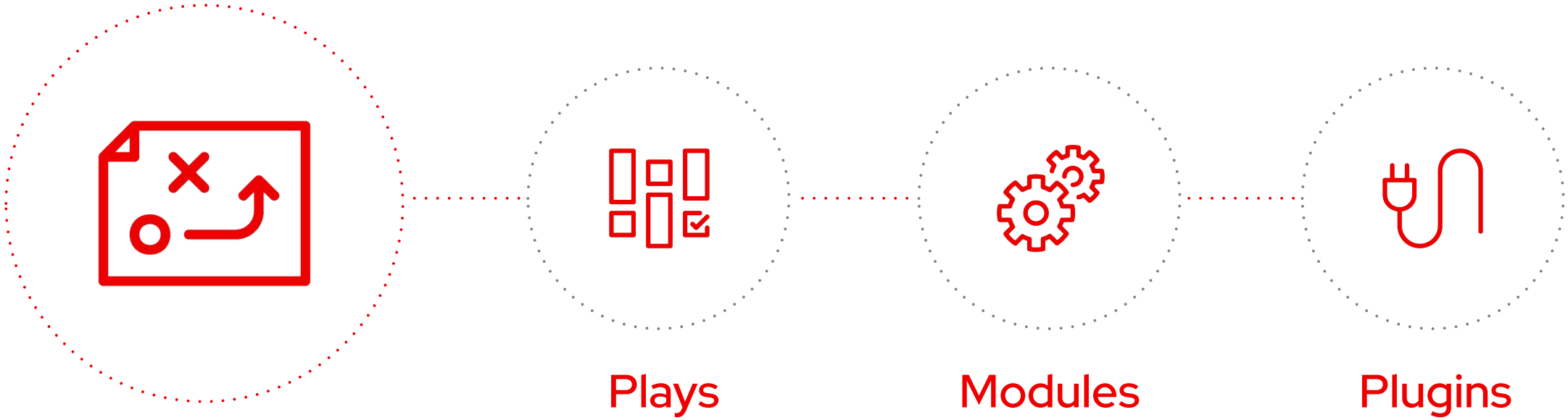
```yaml
---
- name: install and start apache
  hosts: web
  become: yes

  tasks:
    - name: httpd package is present
      yum:
        name: httpd
        state: latest

    - name: latest index.html file is present
      template:
        src: files/index.html
        dest: /var/www/html/

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

What makes up an Ansible playbook?



Ansible plays

What am I automating?

What are they?

Top level specification for a group of tasks. Will tell that play which hosts it will execute on and control behavior such as fact gathering or privilege level.

Building blocks for playbooks

Multiple plays can exist within an Ansible playbook that execute on different hosts.

```
---
- name: install and start apache
  hosts: web
  become: yes
```

Ansible modules

The “tools in the toolkit”

What are they?

Parametrized components with internal logic, representing a single step to be done.
The modules “do” things in Ansible.

Language

Usually Python, or Powershell for Windows setups. But can be of any language.

```
- name: latest index.html file ...
  template:
    src: files/index.html
    dest: /var/www/html/
```

Ansible plugins

The “extra bits”

What are they?

Plugins are pieces of code that augment Ansible’s core functionality. Ansible uses a plugin architecture to enable a rich, flexible, and expandable feature set.

Example become plugin:

```
---
- name: install and start apache
  hosts: web
  become: yes
```

Example filter plugins:

```
{{ some_variable | to_nice_json }}
{{ some_variable | to_nice_yaml }}
```

Ansible Inventory

The systems that a playbook runs against

What are they?

List of systems in your infrastructure that automation is executed against

```
[web]
webserver1.example.com
webserver2.example.com

[db]
dbserver1.example.com

[switches]
leaf01.internal.com
leaf02.internal.com
```

Ansible roles

Reusable automation actions

What are they?

Group your tasks and variables of your automation in a reusable structure. Write roles once, and share them with others who have similar challenges in front of them.

```
---
- name: install and start apache
  hosts: web
  roles:
    - common
    - webservers
```

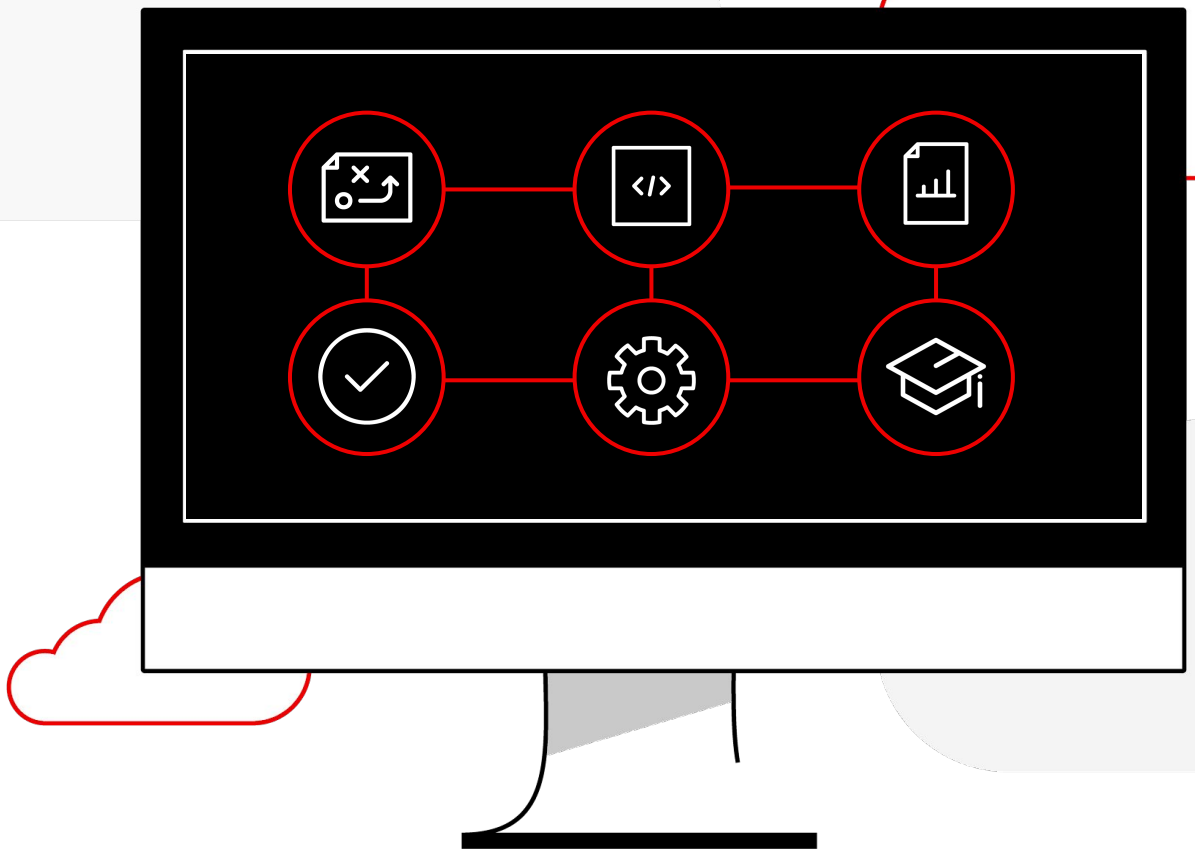
Collections

Simplified and consistent content delivery

What are they?

Collections are a data structure containing automation content:

- Modules
- Playbooks
- Roles
- Plugins
- Docs
- Tests



Red Hat

Collections

```
nginx_core
├── MANIFEST.json
├── playbooks
│   ├── deploy-nginx.yml
│   └── ...
├── plugins
├── README.md
└── roles
    ├── nginx
    │   ├── defaults
    │   ├── files
    │   │   └── ...
    │   ├── tasks
    │   └── templates
    │       └── ...
    ├── nginx_app_protect
    └── nginx_config
```

deploy-nginx.yml

```yaml
---
- name: Install NGINX Plus
  hosts: all
  tasks:
    - name: Install NGINX
      include_role:
        name: nginxinc.nginx
      vars:
        nginx_type: plus

    - name: Install NGINX App Protect
      include_role:
        name: nginxinc.nginx_app_protect
      vars:
        nginx_app_protect_setup_license: false
        nginx_app_protect_remove_license: false
        nginx_app_protect_install_signatures: false
```

90+

certified platforms

Infrastructure

Cloud

Network

Security

Red Hat

aws

ARISTA

Check Point
SOFTWARE TECHNOLOGIES LTD

NetApp™

Google

CISCO

CYBERARK

IBM

Microsoft

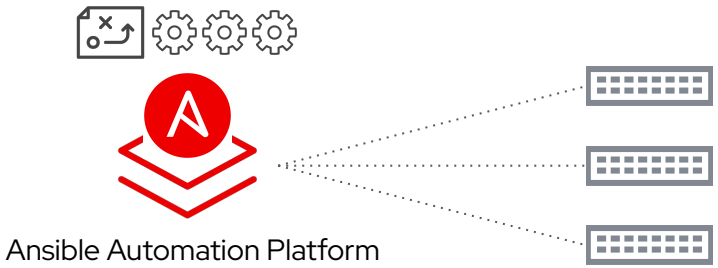
f5

FORTINET

Red Hat

How Ansible Automation Works

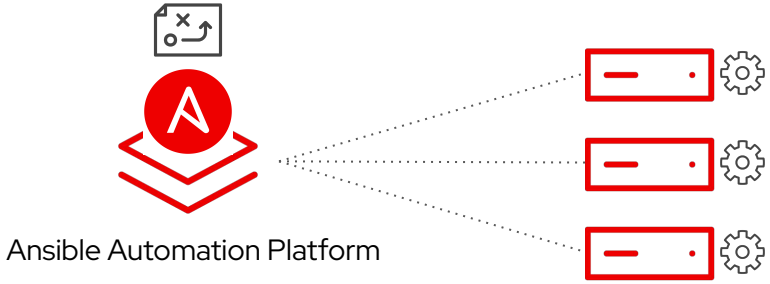
Module code is executed locally on the control node



Local Execution

Network Devices / API Endpoints

Module code is copied to the managed node, executed, then removed



Remote Execution

Linux / Windows Hosts

Red Hat

Inventory

- Ansible works against multiple systems in an inventory
- Inventory is usually file based
- Can have multiple groups
- Can have variables for each group or even host

Ansible Inventory

The Basics

An example of a static Ansible inventory including systems with IP addresses as well as fully qualified domain name (FQDN)

```
[myservers]
10.42.0.2
10.42.0.6
10.42.0.7
10.42.0.8
10.42.0.100
host.example.com
```

Ansible Inventory - The Basics

```
[app1srv]
appserver01 ansible_host=10.42.0.2
appserver02 ansible_host=10.42.0.3

[web]
node-[1:30]

[web:vars]
apache_listen_port=8080
apache_root_path=/var/www/mywebdocs/

[all:vars]
ansible_user=kev
ansible_ssh_private_key_file=/home/kev/.ssh/id_rsa
```

Ansible Inventory - Variables

```
[app1srv]
appserver01 ansible_host=10.42.0.2
appserver02 ansible_host=10.42.0.3

[web]
node-[1:30]

[web:vars]
apache_listen_port=8080
apache_root_path=/var/www/mywebdocs/

[all:vars]
ansible_user=ender
ansible_ssh_private_key_file=/home/ender/.ssh/id_rsa
```

Accessing the Ansible docs

With the use of the latest command utility ansible-navigator, one can trigger access to all the modules available to them as well as details on specific modules.

A formal introduction to ansible-navigator and how it can be used to run playbooks in the following exercise.

```
$ ansible-navigator doc -l -m stdout
add_host
amazon.aws.aws_az_facts
amazon.aws.aws_caller_facts
amazon.aws.aws_caller_info
.
.
.
.
.
```

Accessing the Ansible docs

Aside from listing a full list of all the modules, you can use ansible-navigator to provide details about a specific module.

In this example, we are getting information about the user module.

```
$ ansible-navigator doc user -m stdout

> ANSIBLE.BUILTIN.USER
(/usr/lib/python3.8/site-packages/ansible/m
odules/user.py)

Manage user accounts and user attributes.
For Windows targets, use the
[ansible.windows.win_user] module
        instead.
```

A play

```yaml
---
- name: install and start apache
  hosts: web
  become: yes

  tasks:
    - name: httpd package is present
      yum:
        name: httpd
        state: latest

    - name: latest index.html file is present
      template:
        src: files/index.html
        dest: /var/www/html/

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

Ansible playbooks

A task

```
---
- name: install and start apache
  hosts: web
  become: yes

  tasks:
    - name: httpd package is present
      yum:
        name: httpd
        state: latest

    - name: latest index.html file is present
      template:
        src: files/index.html
        dest: /var/www/html/

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

Ansible playbooks

A module

```yaml
---
- name: install and start apache
  hosts: web
  become: yes

  tasks:
    - name: httpd package is present
      yum:
        name: httpd
        state: latest

    - name: latest index.html file is present
      template:
        src: files/index.html
        dest: /var/www/html/

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

Running Playbooks

The most important colors of Ansible

A task executed as expected, no change was made.

A task executed as expected, making a change

A task failed to execute successfully

Running an Ansible Playbook

Using the latest ansible-navigator command

What is ansible-navigator?

ansible-navigator command line utility and text-based user interface (TUI) for running and developing Ansible automation content.

It replaces the previous command used to run playbooks “ansible-playbook”.

```
$ ansible-navigator run playbook.yml
```

ansible-navigator

Bye ansible-playbook, Hello ansible-navigator

How do I use ansible-navigator?

As previously mentioned, it replaces the ansible-playbook command.

As such it brings two methods of running playbooks:

- Direct command-line interface
- Text-based User Interface (TUI)

```
# Direct command-line interface method
$ ansible-navigator run playbook.yml -m stdout

# Text-based User Interface method
$ ansible-navigator run playbook.yml
```

ansible-navigator

Mapping to previous Ansible commands

ansible command	ansible-navigator command
`ansible-config`	`ansible-navigator config`
`ansible-doc`	`ansible-navigator doc`
`ansible-inventory`	`ansible-navigator inventory`
`ansible-playbook`	`ansible-navigator run`

ansible-navigator

Common subcommands

Name	Description	CLI Example	Colon command within TUI
collections	Explore available collections	ansible-navigator collections --help	:collections
config	Explore the current ansible configuration	ansible-navigator config --help	:config
doc	Review documentation for a module or plugin	ansible-navigator doc --help	:doc
images	Explore execution environment images	ansible-navigator images --help	:images
inventory	Explore and inventory	ansible-navigator inventory --help	:inventory
replay	Explore a previous run using a playbook artifact	ansible-navigator replay --help	:replay
run	Run a playbook	ansible-navigator run --help	:run
welcome	Start at the welcome page	ansible-navigator welcome --help	:welcome

Red Hat

Ansible playbooks

```yaml
---
- name: variable playbook test
  hosts: localhost

  vars:
    var_one: awesome
    var_two: ansible is
    var_three: "{{ var_two }} {{ var_one }}"

  tasks:
    - name: print out var_three
      debug:
        msg: "{{ var_three }}"
```

Ansible playbooks

```yaml
---
- name: variable playbook test
  hosts: localhost

  vars:
    var_one: awesome
    var_two: ansible is
    var_three: "{{ var_two }} {{ var_one }}"

  tasks:
    - name: print out var_three
      debug:
        msg: "{{ var_three }}"
```

ansible is awesome

Ansible Facts

- Just like variables, really...
- ... but: coming from the host itself!
- Check them out with the setup module

```yaml
tasks:
  - name: Collect all facts of host
    setup:
      gather_subset:
        - 'all'
```

Ansible playbooks

```yaml
---
- name: facts playbook
  hosts: localhost

  tasks:
    - name: Collect all facts of host
      setup:
        gather_subset:
          - ‘all’
```

```
$ ansible-navigator run playbook.yml
```

Ansible Navigator TUI

```
PLAY NAME            OK  CHANGED    UNREACHABLE     FAILED   SKIPPED    IGNORED   IN PROGRESS     TASK COUNT  PROGRESS
0|facts playbook     2       0              0          0         0          0             0              2   COMPLETE

RESULT      HOST             NUMBER     CHANGED      TASK                              TASK ACTION        DURATION
0|OK        localhost             0       False      Gathering Facts                   gather_facts            1s
1|OK        localhost             1       False      Collect all facts of host         setup                   1s

PLAY [facts playbook:1]
*******************************************************************************************************************
TASK [Collect all facts of host]
*******************************************************************************************************************
OK: [localhost]
.
.
12 |  ansible_facts:
 13|    ansible_all_ipv4_addresses:
 14|    - 10.0.2.100
 15|    ansible_all_ipv6_addresses:
 16|    - fe80::1caa:f0ff:fe15:23c4
```

Conditionals via VARS

Example of using a variable labeled *my_mood* and using it as a conditional on a particular task.

```yaml
vars:
   my_mood: happy

 tasks:
 - name: task, based on my_mood var
   debug:
     msg: "Yay! I am {{ my_mood }}!"
   when: my_mood == "happy"
```


Red Hat

Ansible Conditionals

```yaml
---
- name: variable playbook test
  hosts: localhost

  vars:
    my_mood: happy

  tasks:
    - name: task, based on my_mood var
      debug:
        msg: "Yay! I am {{ my_mood }}!"
      when: my_mood == "happy"
```

Alternatively

```yaml
    - name: task, based on my_mood var
      debug:
        msg: "Ask at your own risk. I'm {{ my_mood }}!"
      when: my_mood == "grumpy"
```

Ansible Conditionals w/ Facts

```yaml
---
- name: variable playbook test
  hosts: localhost

  tasks:
  - name: Install apache
    apt:
      name: apache2
      state: latest
    when: ansible_distribution == 'Debian' or
          ansible_distribution == 'Ubuntu'

  - name: Install httpd
    yum:
      name: httpd
      state: latest
    when: ansible_distribution == 'RedHat'
```

Using Previous Task State

```yaml
---
- name: variable playbook test
  hosts: localhost

  tasks:
  - name: Ensure httpd package is present
    yum:
      name: httpd
      state: latest
    register: http_results

  - name: Restart httpd
    service:
      name: httpd
      state: restart
    when: httpd_results.changed
```

Ansible Handler Tasks

```yaml
---
- name: variable playbook test
  hosts: localhost

  tasks:
  - name: Ensure httpd package is present
    yum:
      name: httpd
      state: latest
  notify: restart_httpd

  handlers:
  - name: restart_httpd
    service:
      name: httpd
      state: restart
```

Ansible Handler Tasks

```
tasks:
- name: Ensure httpd package is present
  yum:
    name: httpd
    state: latest
  notify: restart_httpd

- name: Standardized index.html file
  copy:
    content: "This is my index.html file for {{ ansible_host }}"
    dest: /var/www/html/index.html
  notify: restart_httpd
```

If **either** task notifies a **changed** result, the handler will be notified **ONCE**.

```
TASK [Ensure httpd package is present] ******************************************
ok: [web2]
ok: [web1]

TASK [Standardized index.html file] *********************************************
changed: [web2]
changed: [web1]

NOTIFIED: [restart_httpd] *******************************************************
changed: [web2]
changed: [web1]
```

unchanged

changed

handler runs once

Ansible Handler Tasks

```
tasks:
- name: Ensure httpd package is present
  yum:
    name: httpd
    state: latest
  notify: restart_httpd

- name: Standardized index.html file
  copy:
    content: "This is my index.html file for {{ ansible_host }}"
    dest: /var/www/html/index.html
  notify: restart_httpd
```

If **both** of these tasks notifies of a **changed** result, the handler will be notified **ONCE**.

```
TASK [Ensure httpd package is present] ******************************************
changed: [web2]
changed: [web1]

TASK [Standardized index.html file] *********************************************
changed: [web2]
changed: [web1]

NOTIFIED: [restart_httpd] *******************************************************
changed: [web2]
changed: [web1]
```

changed

changed

handler runs once

Ansible Handler Tasks

```yaml
tasks:
- name: Ensure httpd package is present
  yum:
    name: httpd
    state: latest
  notify: restart_httpd

- name: Standardized index.html file
  copy:
    content: "This is my index.html file for {{ ansible_host }}"
    dest: /var/www/html/index.html
  notify: restart_httpd
```

If **neither** task notifies a **changed** result, the handler ***does not run.***

```
TASK [Ensure httpd package is present] ***********************************************
ok: [web2]
ok: [web1]

TASK [Standardized index.html file] ***************************************************
ok: [web2]
ok: [web1]

PLAY RECAP ****************************************************************************
web2       : ok=2   changed=0   unreachable=0   failed=0   skipped=0   rescued=0 ignored=0
web1       : ok=2   changed=0   unreachable=0   failed=0   skipped=0   rescued=0 ignored=0
```

unchanged

unchanged

```yaml
---
- name: Ensure users
  hosts: node1
  become: yes

  tasks:
    - name: Ensure user is present
      user:
        name: dev_user
        state: present

    - name: Ensure user is present
      user:
        name: qa_user
        state: present

    - name: Ensure user is present
      user:
        name: prod_user
        state: present
```

```yaml
---
- name: Ensure users
  hosts: node1
  become: yes

  tasks:
    - name: Ensure user is present
      user:
        name: "{{item}}"
        state: present
      loop:
        - dev_user
        - qa_user
        - prod_user
```

```yaml
- name: Ensure apache is installed and started
  hosts: web
  become: yes
  vars:
    http_port: 80
    http_docroot: /var/www/mysite.com

  tasks:
    - name: Verify correct config file is present
      template:
        src: templates/httpd.conf.j2
        dest: /etc/httpd/conf/httpd.conf
```

```yaml
- name: Ensure apache is installed and started
  hosts: web
  become: yes
  vars:
    http_port: 80
    http_docroot: /var/www/mysite.com

  tasks:
    - name: Verify correct config file is present
      template:
        src: templates/httpd.conf.j2
        dest: /etc/httpd/conf/httpd.conf
```

```
## Excerpt from httpd.conf.j2

# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses.
#
# Listen 80   ## original line
Listen {{ http_port }}

# DocumentRoot: The directory out of which you will serve your
# documents.
# DocumentRoot "/var/www/html"
DocumentRoot {{ http_docroot }}
```

Role Structure

- Defaults: default variables with lowest precedence (e.g. port)
- Handlers: contains all handlers
- Meta: role metadata including dependencies to other roles
- Tasks: plays or tasks
  Tip: It's common to include tasks in main.yml with "when" (e.g. OS == xyz)
- Templates: templates to deploy
- Tests: place for playbook tests
- Vars: variables (e.g. override port)

```
user/
├── defaults
│   └── main.yml
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```


Red Hat

Thank you

linkedin.com/company/red-hat

youtube.com/AnsibleAutomation

facebook.com/ansibleautomation

twitter.com/ansible

github.com/ansible

Red Hat