Ansible for Beginners

DI (FH) René Koch
Freelancer
The Checkmk Conference #10, 13.06.2024

About me

- René Koch
- Self employeed consultant for:
  - Red Hat Ansible (Automation Platform)
  - Red Hat Enterprise Linux
  - Red Hat Satellite
  - Red Hat Identity Management (IPA)
  - (previously) Icinga 2

About me

- René Koch
  - rkoch@rk-it.at
  - +43 660 / 464 0 464
  - https://at.linkedin.com/in/ren%C3%A9-koch-86516972

Please introduce yourself

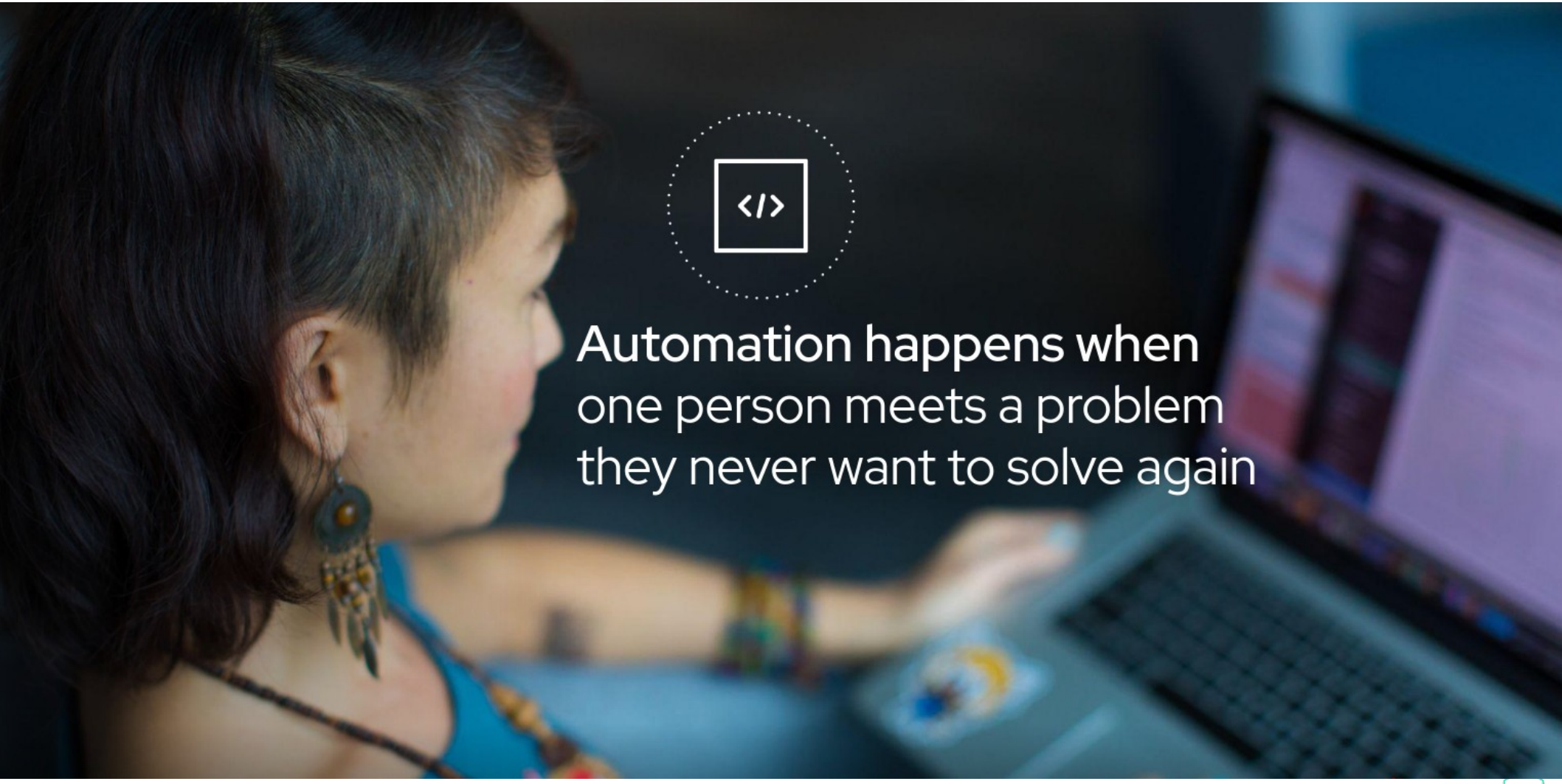
Time table

- 08:30 – 10:00: Workshop
- 10:00 – 10:15: Break
- 10:15 – 12:00: Workshop
- 12:00 – 13:15: Lunch break
- 13:15 – 15:00: Workshop
- 15:00 – 15:15: Break
- 15:15 – 16:45: Workshop

Table of content

What is Ansible?

What is Ansible?



Source: https://github.com/ansible/workshops/blob/devel/decks/ansible_rhel.pdf

What is Ansible?

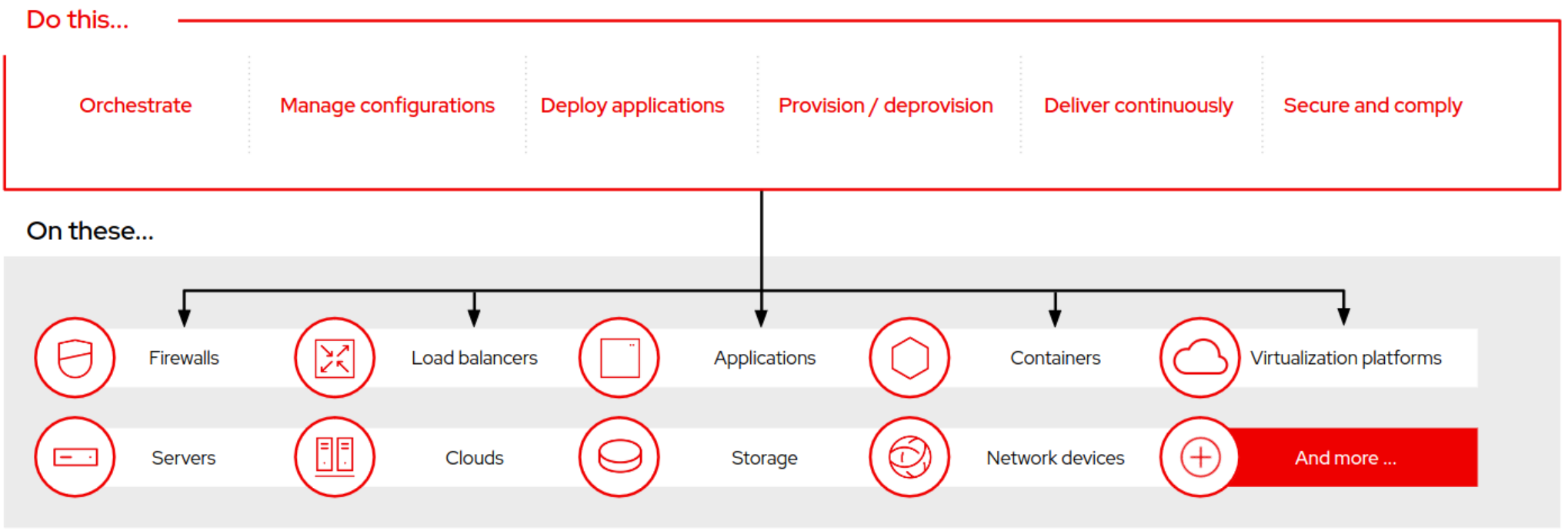
- Automation of provisioning, application deployment and Configuration management
- No agent is required on the target machine
- Use of SSH, WinRM and APIs
- Parallel execution of tasks on multiple machines
- Easy (to read) automation language (YAML)

Source: https://www.openssh.com/

What is Ansible?

Automate the deployment and management of automation

Your entire IT footprint



Source: https://github.com/ansible/workshops/blob/devel/decks/ansible_rhel.pdf

What is Ansible?

- Competitors:

What is Ansible?

Defining community (or free) Ansible, AWX, and Red Hat Ansible Automation Platform

Community Ansible

Free, unsupported open source command line tool for automation.

AWX

Free, unsupported open source software. A GUI and API tool for wrapping around community Ansible.

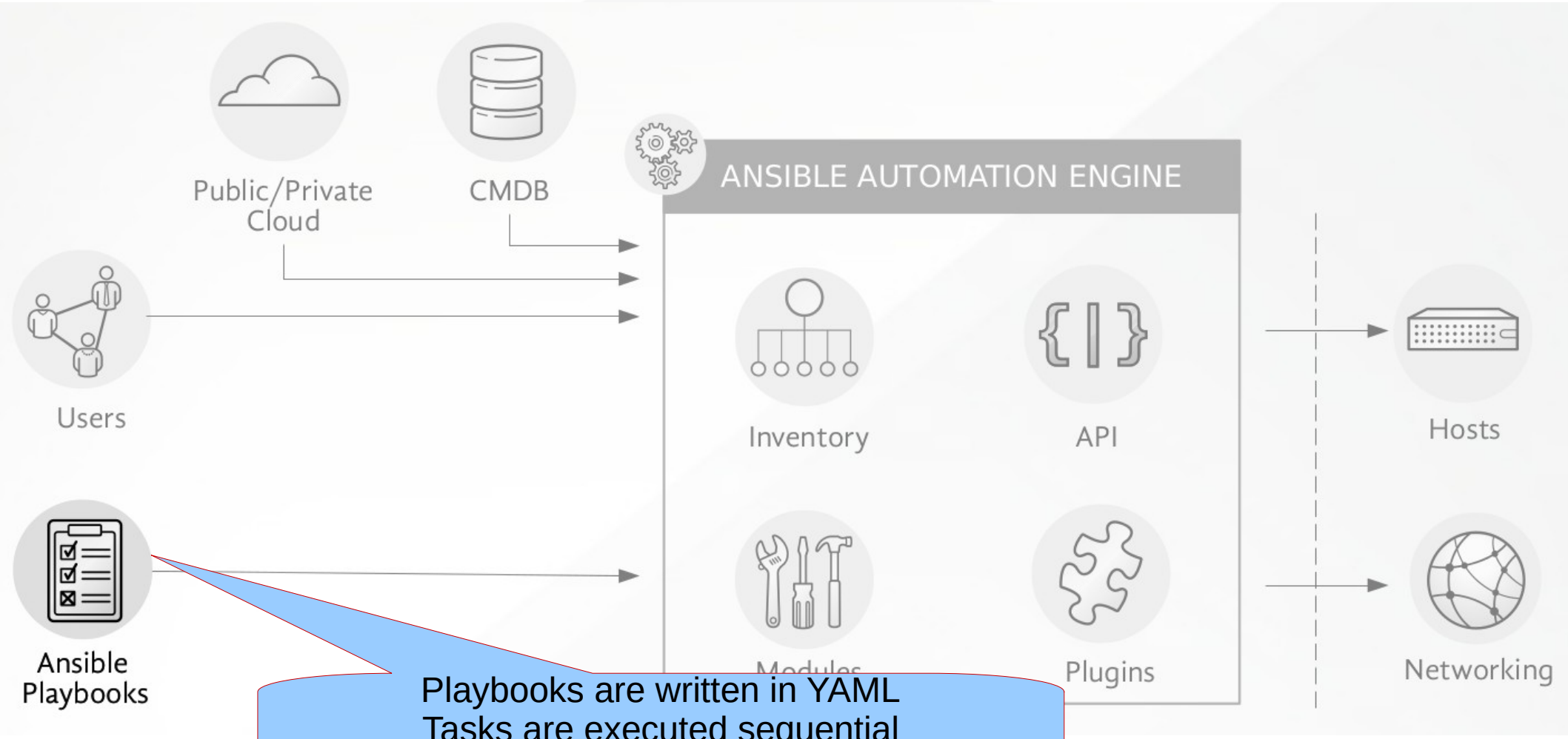
Red Hat Ansible Automation Platform

Subscription enterprise product. Combines 20+ community projects into a fully supported automation platform for your enterprise.

Source: https://www.redhat.com/en/technologies/management/ansible/compare-awx-vs-ansible-automation-platform?hsLang=en-us

Ansible Core

The Ansible basics



Playbooks are written in YAML
Tasks are executed sequential
Tasks execute Ansible modules

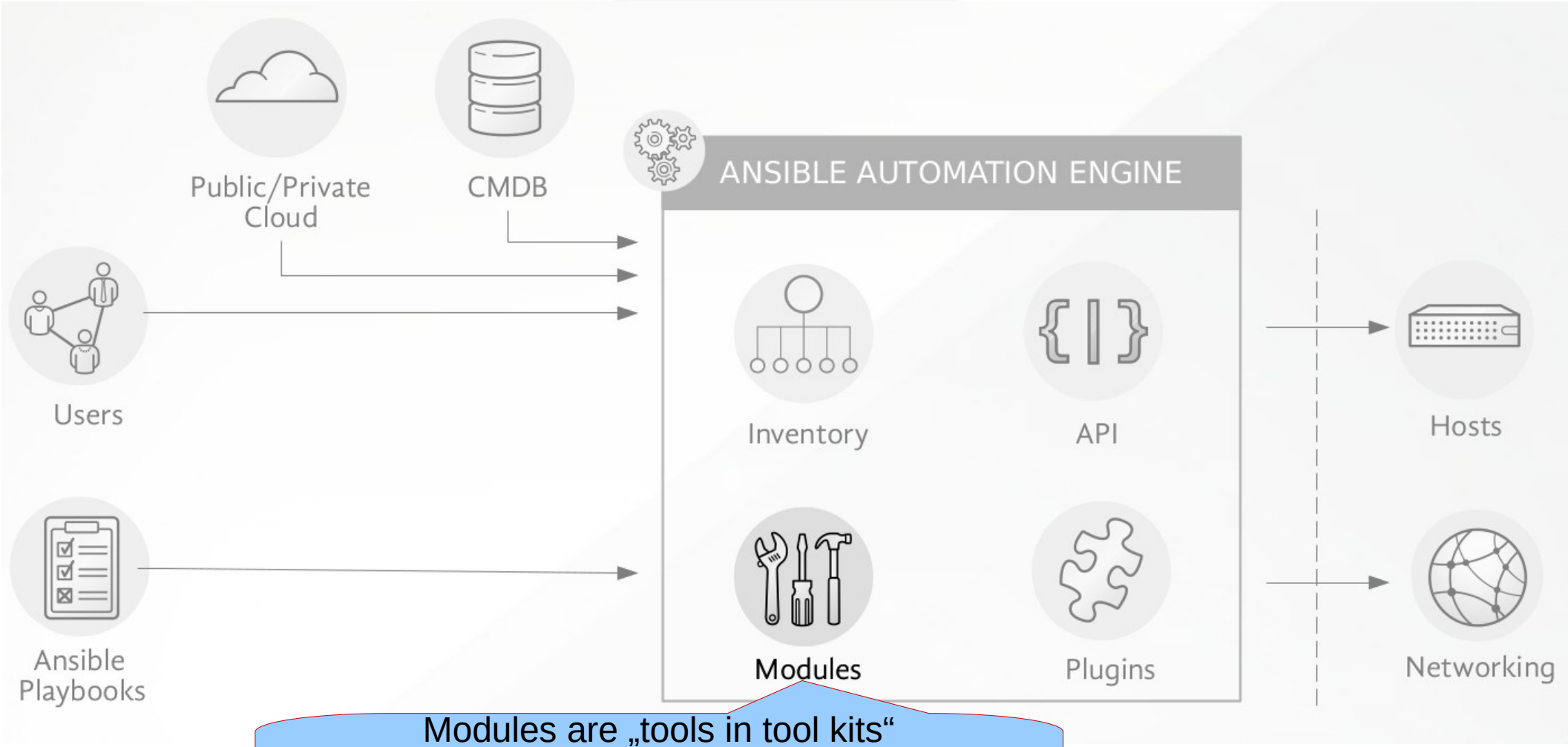
Playbooks

```
---

- name: Configure nginx webserver
  hosts: webservers
  become: true
  tasks:
    - name: Enable epel repository
      ansible.builtin.yum:
        name: epel-release
        state: present

    - name: Install nginx
      ansible.builtin.yum:
        name: nginx
        state: present

    - name: Copy static.html
      ansible.builtin.copy:
        src: files/static.html
        dest: /usr/share/nginx/html/static.html
```

Playbooks

```
$ ansible-playbook web-notls.yml

PLAY [Configure nginx webserver] ***********************************************

TASK [setup] *******************************************************************
ok: [instance-1]

TASK [Enable epel repository] **************************************************
ok: [instance-1]

TASK [Install nginx] ***********************************************************
changed: [instance-1]

TASK [Copy index.html] *********************************************************
changed: [instance-1]

TASK [Start and enable nginx] **************************************************
changed: [instance-1]


PLAY RECAP *********************************************************************
instance-1                 : ok=9    changed=5    unreachable=0    failed=0
```

The Ansible basics

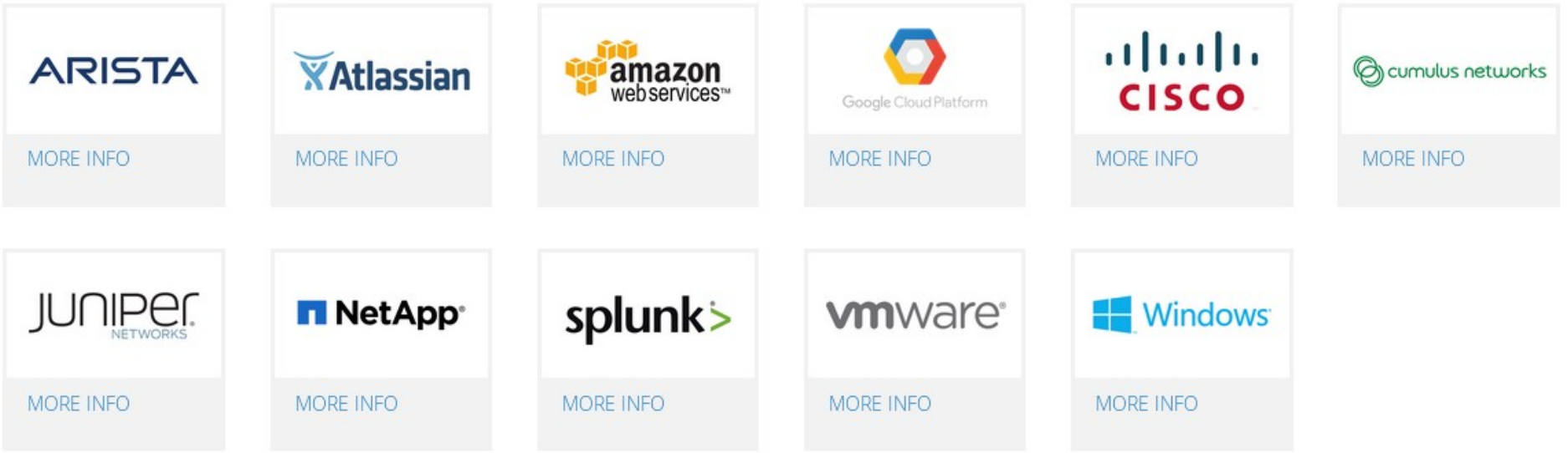


Modules are „tools in tool kits“
Python, Powershell orr any other language
Increase easy deployment of Ansible code

Modules

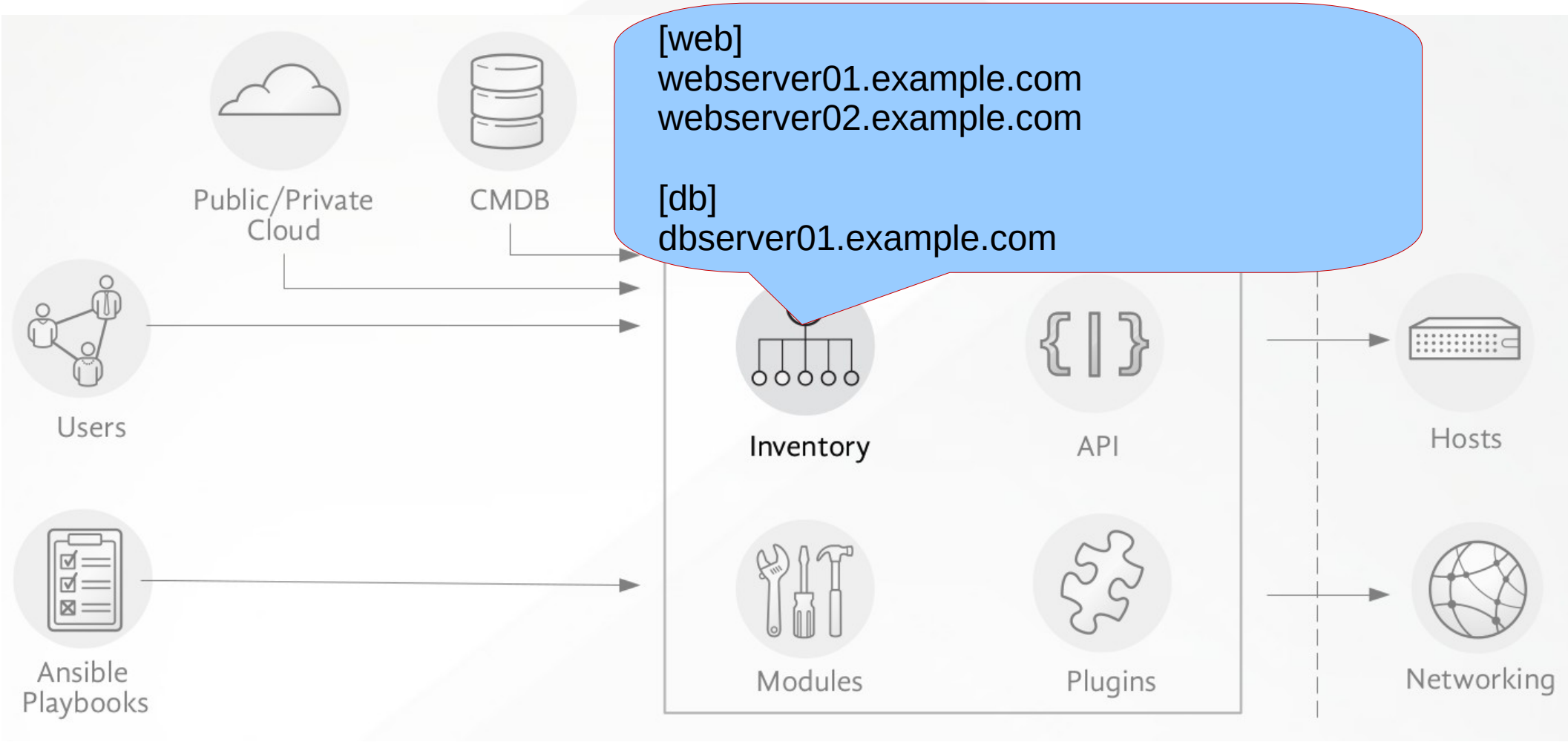
https://docs.ansible.com/ansible/latest/collections/index_module.html

- yum/apt/zypper/...
- service
- template
- user
- file
- copy

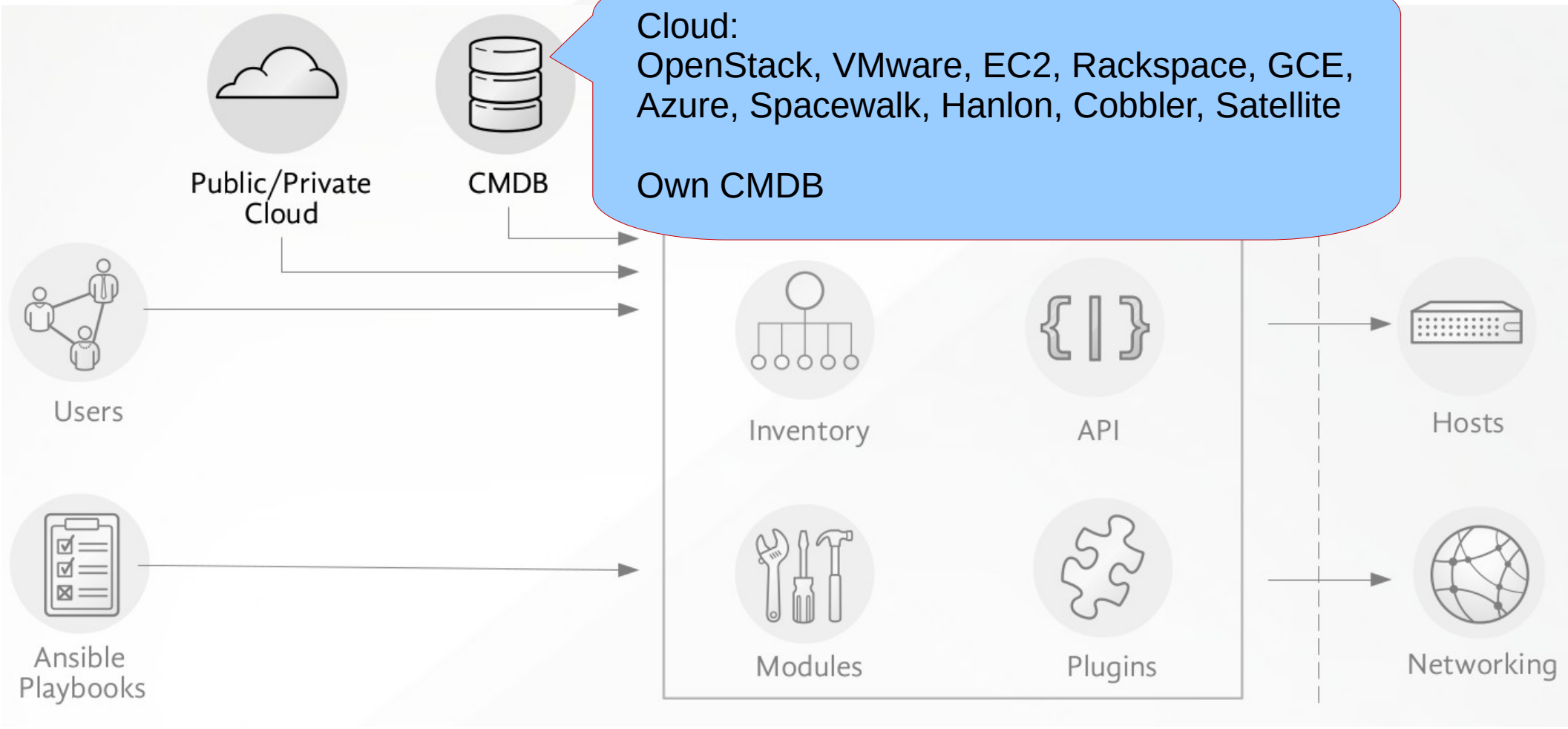


Quelle: https://www.ansible.com/how-ansible-works

The Ansible basics

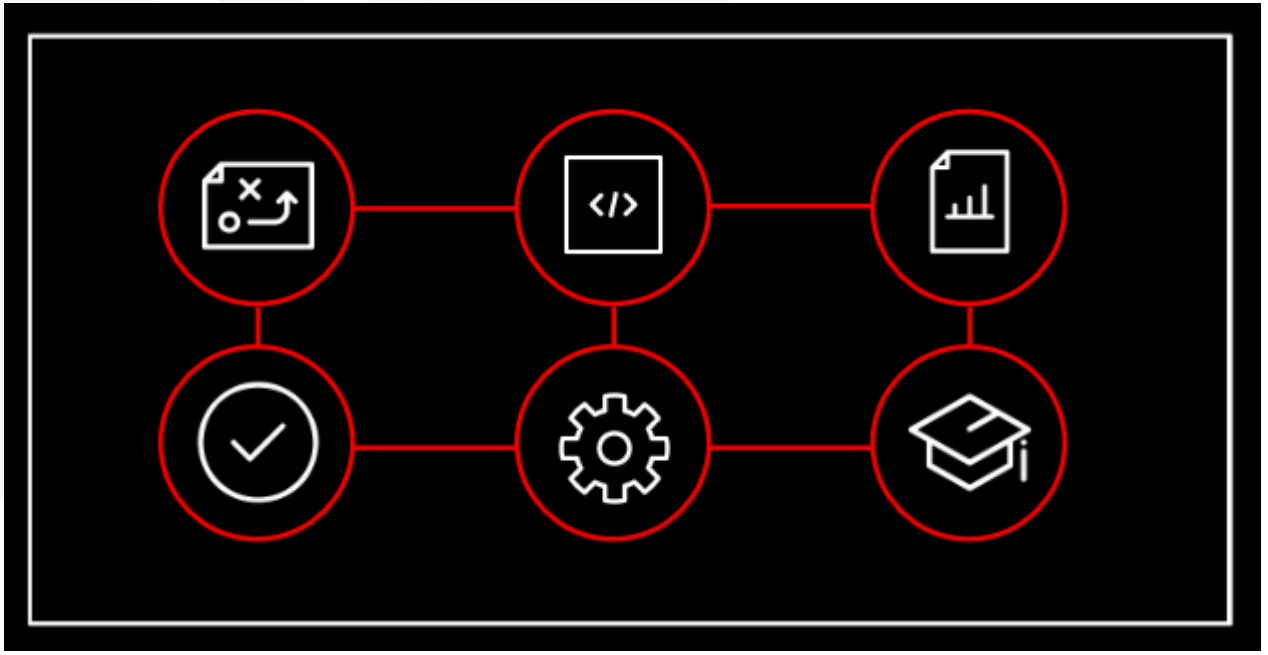


The Ansible basics



Collections

- Simplified and consistent content delivery
- Collections contains:
  - Modules
  - Playbooks
  - Roles
  - Plugins
  - Docs
  - Tests



Source: https://github.com/ansible/workshops/blob/devel/decks/ansible_rhel.pdf

Collections

```
checkmk
└── general
    ├── ansible.cfg
    ├── CHANGELOG.rst
    ├── changelogs
...
    │   │   └── 4.3.1
    │   │       ├── fix_rule_conditions_missing.yml
    │   │       └── release_summary.yml
    ├── docs
    │   ├── activation_module.rst
...
    ├── meta
    │   └── runtime.yml
    ├── playbooks
    │   ├── demo
    │   │   ├── downtimes.yml
...
    ├── plugins
    │   ├── doc_fragments
    │   │   ├── common.py
...
```

Collections

```
...
│   ├── lookup
│   │   └── bakery.py
...
│   ├── modules
│   │   └── activation.py
...
│   └── module_utils
│       └── api.py
...
├── README.md
└── roles
    ├── agent
    │   ├── defaults
    │   │   └── main.yml
    │   ├── handlers
    │   │   └── main.yml
    │   ├── meta
    │   │   └── main.yml
    │   ├── README.md
    │   ├── tasks
    │   │   └── Debian.yml
...
```

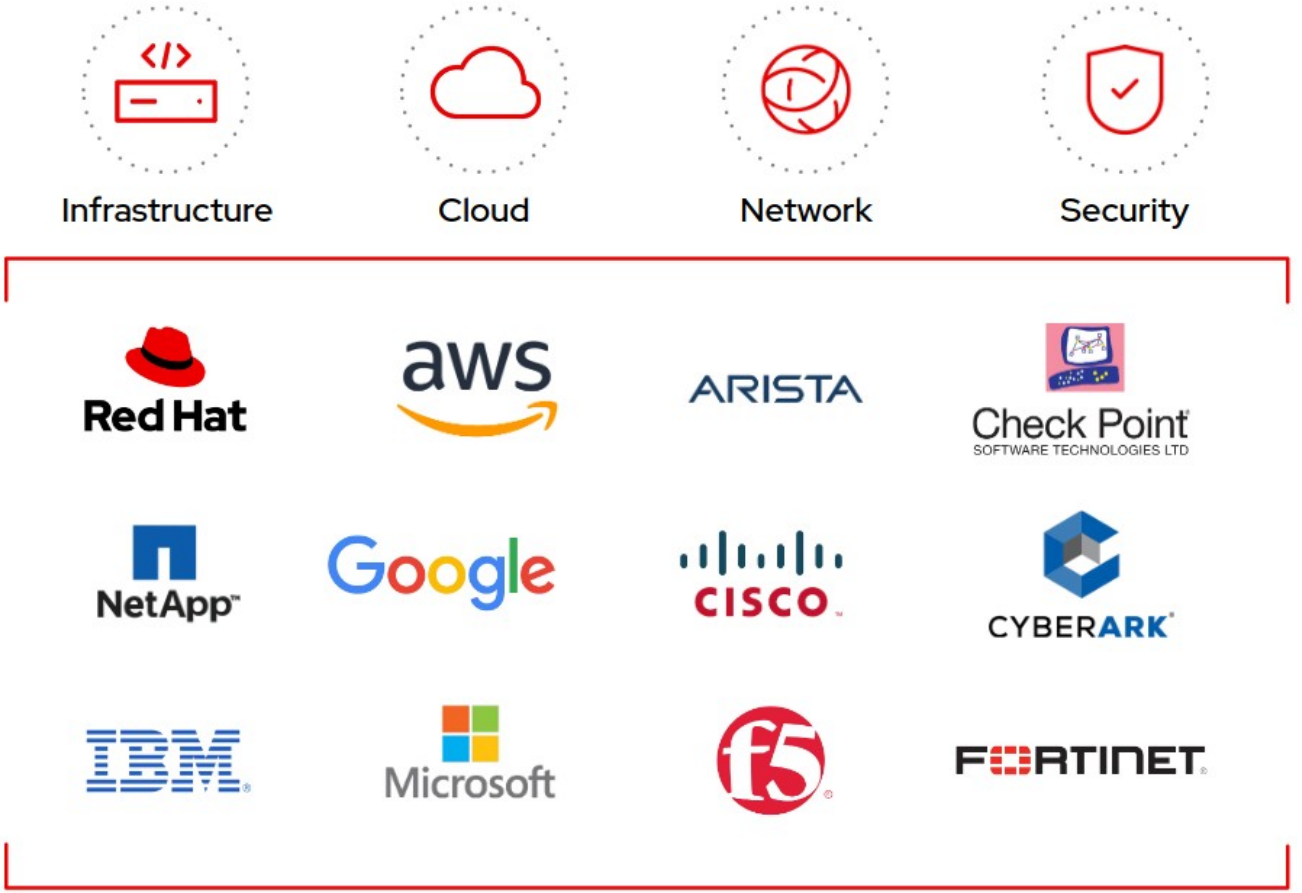
Use Collections in Playbooks

```
---

- name: Configure nginx webserver
  hosts: webservers
  become: true
  tasks:
    - name: Enable epel repository
      ansible.builtin.yum:
        name: epel-release
        state: present

    - name: Install nginx
      ansible.builtin.yum:
        name: nginx
        state: present

    - name: Copy static.html
      ansible.builtin.copy:
        src: files/static.html
        dest: /usr/share/nginx/html/static.html
```

Collections

90+

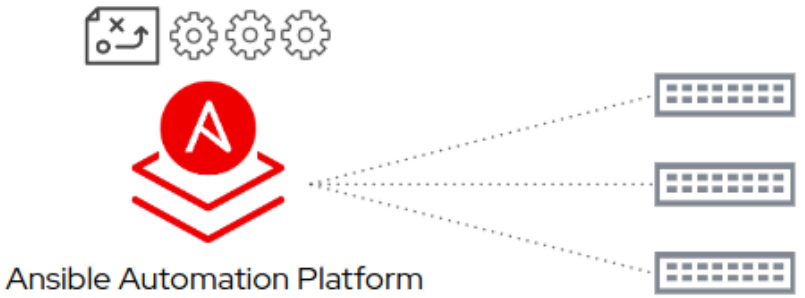
certified platforms



Source: https://github.com/ansible/workshops/blob/devel/decks/ansible_rhel.pdf

How Ansible Works

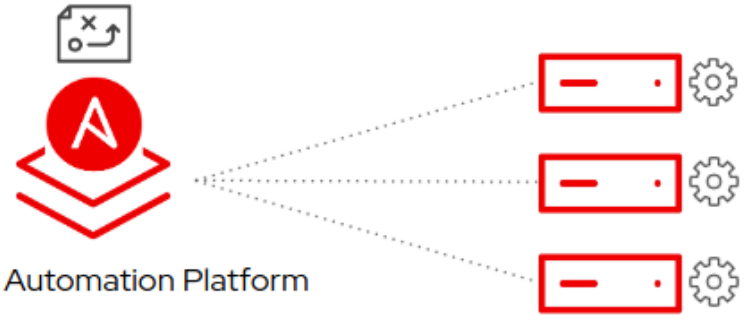
Module code is executed locally on the control node



Local Execution

Network Devices / API Endpoints

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



Remote Execution

Linux / Windows Hosts

Source: https://github.com/ansible/workshops/blob/devel/decks/ansible_rhel.pdf

Preparing the LAB environment

Preparing the LAB environment

- Create LAB environment
  - Control node == target node
  - Use local connection
  - Configure sudo for privileged access

Preparing the LAB environment

- Available VMs (if you don’t have your own)
- Pwds: **AnsibleW0rkshop2024!**

Hostname	IP Address	Userrname	Participant
ansible-ws-1	116.203.227.120	ansible	
ansible-ws-2	162.55.51.64	ansible	
ansible-ws-3	188.245.36.142	ansible	
ansible-ws-4	5.75.175.200	ansible	
ansible-ws-5	94.130.57.183	ansible	
ansible-ws-6	195.201.130.43	ansible	
ansible-ws-7	116.203.17.200	ansible	
ansible-ws-8	128.140.41.17	ansible	
ansible-ws-9	159.69.216.194	ansible	
ansible-ws-10	94.130.181.155	ansible	

Preparing the LAB environment

- Configure sudo for privileged access

```
# <EDITOR> /etc/sudoers.d/ansible
<USERNAME> ALL=(ALL) NOPASSWD: ALL
```

Use user root to run this command!

Replace <EDITOR> with vi, nano or your favorite editor

Replace <USERNAME> with your system user

LAB 1: Configure sudo

LAB 1: Configure sudo

- Configure sudo for privileged access

```
# <EDITOR> /etc/sudoers.d/ansible
<USERNAME> ALL=(ALL) NOPASSWD: ALL
```

Use user root to run this command!

Replace <EDITOR> with vi, nano or your favorite editor

Replace <USERNAME> with your system user

Install Ansible

Install Ansible

- Most Linux distributions ship 2 versions:
  - **ansible-core**: Ansible binary + minimal set of collections
  - **ansible**: Ansible and selected collections

Install Ansible - RHEL

- Install Ansible on RHEL 9

```
# subscription-manager repos --enable
codeready-builder-for-rhel-9-$(arch)-rpms
# dnf install
https://dl.fedoraproject.org/pub/epel/epel
-release-latest-9.noarch.rpm
# dnf install ansible
```

Install Ansible – RHEL derivates

- Install Ansible on AlmaLinux, Rocky Linux 9

```
# dnf config-manager --set-enabled crb
# dnf install epel-release
# dnf install ansible
```

Source: https://commons.wikimedia.org/wiki/File:AlmaLinux_Icon_Logo.png

Install Ansible – openSUSE

- Install Ansible on openSUSE 15.5

```
# zypper install ansible
```

Source: http://kuboosoft.blogspot.com/2012/12/opensuse-123-milestone-2-esta-aqui.html

Install Ansible – Ubuntu

- Install Ansible on Ubuntu 24.04

```
# apt update
# apt install ansible
```

Source: https://windytheplaneh.deviantart.com/art/with-speedvideo-Ubuntu-logo-vector-1-599350984

Install Ansible – pip

- Install Ansible with pip

```
$ pip install ansible
```

Source: https://sefiks.com/2020/03/21/publishing-python-packages-on-pip-and-pypi/

LAB 2: Install Ansible

LAB 2: Install Ansible

- Ensure Ansible is installed

```
$ ansible --version
ansible [core 2.14.14]
```

Depending on your operating system the Ansible version can be different

Prepare Ansible inventory

Prepare Ansible inventory

- Ansible needs to be aware of target machines
- As there is no agent, an inventory is used
- Inventory defines:
  - Available hosts
  - How to connect to these hosts
- More details about inventories will follow later in this workshop

Prepare Ansible inventory

- Create folder for your Ansible code

```
$ mkdir playbooks
$ cd playbooks
$ <EDITOR> hosts

testserver ansible_connection=local
```

Replace <EDITOR> with vi, nano or your favorite editor

Test connection to target server

- Make sure your target server is reachable

```
$ ansible testserver -i hosts -m ping
testserver | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Ansible does not send an ICMP ping to test the connection, it tries to log into the target machine.

LAB 3: Add target to Ansible inventory

LAB 3: Add target to Ansible inventory

- Make sure your target server is reachable

```
$ ansible testserver -i hosts -m ping
testserver | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Ad-hoc Commands

Ad-hoc Commands

- Easy way to execute ~~commands~~ modules on multiple machines
- Get list of available modules in latest Ansible package: https://docs.ansible.com/ansible/latest/collections/index_module.html

Ad-hoc Commands

- Get uptime

```
$ ansible testserver -i hosts -m command -a uptime
testserver | CHANGED | rc=0 >>
 14:42:44 up 39 min,  1 user,  load average: 0.00, 0.01, 0.01
```

- Show last 10 lines of messages
- `$ ansible testserver -i hosts -a "tail /var/log/messages"`

```
testserver | FAILED | rc=1 >>
tail: cannot open '/var/log/messages' for reading: Permission deniednon-zero return code
```

Ad-hoc Commands

- Show last 10 lines of messages with root permissions
- `$ ansible testserver -i hosts -a "tail /var/log/messages" -b`
- Install Apache package
- `$ ansible testserver -i hosts -m yum -a "name=httpd state=installed" -b`

LAB 4: Create user

LAB 4: Create user

- Create an user with the following settings:
  - Name: testuser
  - Gecos/Comment: Test User
  - Shell: /bin/bash
  - Ensure that the home directory /home/testuser will be created automatically

https://docs.ansible.com/ansible/latest/collections/index_module.html#ansible-builtin

LAB 4: Create user

- Solution
- **$ ansible testserver -i hosts -m ansible.builtin.user -a "name=testuser comment=\"Test User\" shell=/bin/bash state=present" -b**

```
testserver  | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "comment": "Test User",
    "create_home": true,
    "group": 1001,
    "home": "/home/testuser",
    "name": "testuser",
    "shell": "/bin/bash",
    "state": "present",
    "system": false,
    "uid": 1001
}
```

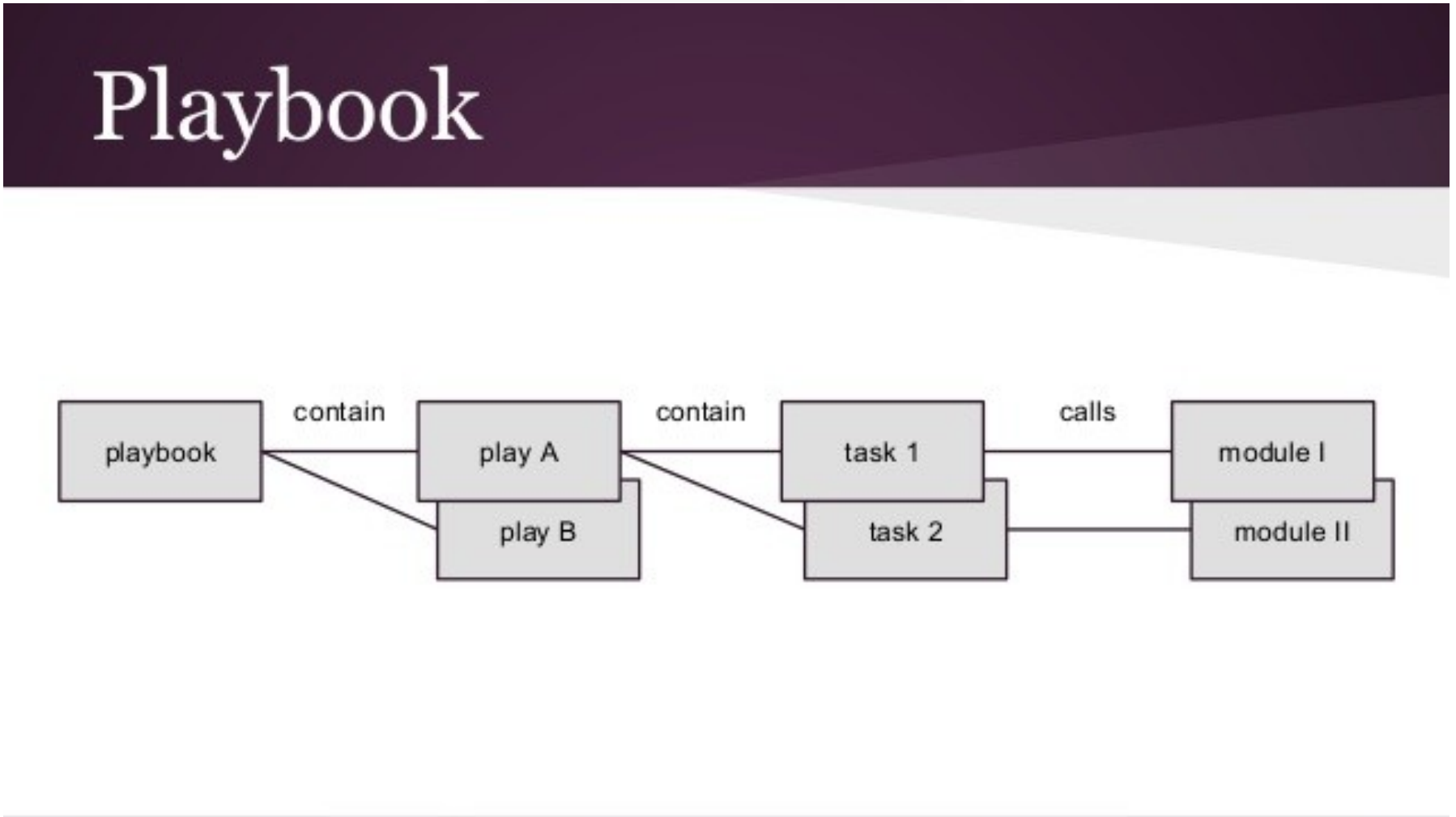
Playbooks

Playbooks

- Recurring way of executing Ansible code
- Collection of Ansible tasks
- Dependencies between tasks

- In this workshop we are going to create a webserver (Apache) with a default web page.

Playbooks



Source: *https://image.slidesharecdn.com/ansible-150925121447-lva1-app6891/95/ansible-101-16-638.jpg?cb=1443183393*

Playbooks

- A playbooks consists of 1 or multiple **plays**

```
- name: Configure Apache webserver
```

- Each play consists of 1 or multiple **tasks**

```
- name: Enable epel repository
- name: Install Apache
```

Playbooks

- Each tasks executes a **module**

```
- ansible.bultin.yum
- ansible.builtin.copy
- ansible.builtin.template
- ansible.builtin.service
```

Simple Playbook

- Install Apache webserver with Ansible
- Use the correct sample code matching your operating system – either Red Hat based, Debian based or SUSE based

```
$ <EDITOR> web-notls.yml
```

Ansible playbooks are written in YAML – make sure that the YAML-syntax is correct. Never use tab, always use spaces!

Simple Playbook – Red Hat

```
---
- name: Configure Apache webserver
  hosts: webservers
  become: true

  tasks:
    - name: Install Apache
      ansible.builtin.dnf:
        name: httpd
        state: present
```

Simple Playbook – Debian

```
---
- name: Configure Apache webserver
  hosts: webservers
  become: true

  tasks:
    - name: Install Apache
      ansible.builtin.apt:
        name: apache2
        update_cache: true
        state: present
```

Simple Playbook – SUSE

```
---
- name: Configure Apache webserver
  hosts: webservers
  become: true

  tasks:
    - name: Install Apache
      community.general.zypper:
        name: apache2
        state: present
```

Simple Playbook

- It would be possible to use the generic **ansible.bultin.package** module, but imho it's easier to understand the code if the os specific package module is used

Note the different syntax between Ad-hoc command arguments (=) and a structured playbooks. In Playbooks, each argument is in a seperate line (:)

Simple Playbook

- This playbooks should be executed only for group **webservers**. No ssh connection is required for the test host

```
$ <EDITOR> hosts
[webservers]
restserver ansible_connection=local
```

Simple Playbook

- Playbooks are executed with command **ansible-playbook** instead of ansible

```
$ ansible-playbook -i hosts web-notls.yml

PLAY [Configure Apache webserver] ************************************************

TASK [setup] *********************************************************************

ok: [testserver]

TASK [Install Apache] ************************************************************

changed: [testserver]

PLAY RECAP ***********************************************************************

testserver                 : ok=2    changed=1    unreachable=0    failed=0
```

Simple Playbook

- Static files can be copied with **ansible.builtin.copy** module
- Ensure to deploy **static.html** page on your host

```
$ <EDITOR> web-notls.yml
```

Note: don’t add the placeholder „...“ to your playbook – it just tells you to append the new code before or after the existing one!

Simple Playbook – Red Hat/Debian

```
...

    - name: Copy static.html
      ansible.builtin.copy:
        src: files/static.html
        dest: /var/www/html/static.html
```

Simple Playbook – SUSE

```
...

    - name: Copy static.html
      ansible.builtin.copy:
        src: files/static.html
        dest: /srv/www/htdocs/static.html
```

Simple Playbook

- Static files should be placed in folder **files**

```
$ mkdir files
$ <EDITOR> files/static.html

<html>
  <head>
    <title>Ansible workshop</title>
  </head>
  <body>
    <h1>Ansible workshop</h1>
    <p>This is our first web application, created in Ansible
workshop!</p>
  </body>
</html>
```

Simple Playbook

- Deploy static webpage

```
$ ansible-playbook -i hosts web-notls.yml

PLAY [Configure Apache webserver]
**********************************************

TASK [setup]
*******************************************************************

ok: [testserver]

TASK [Install Apache]
***********************************************

ok: [testserver]

TASK [Copy static.html] ****************************************

changed: [testserver]

PLAY RECAP
*********************************************************************

testserver                 : ok=3    changed=1    unreachable=0    failed=0
```

Simple Playbook

- Deploy static webpage

```
$ ansible-playbook -i hosts web-notls.yml

TASK [Install Apache]
**********************************************************

ok: [testserver]
```

If there are no changes to the system – e.g. Apache is already installed, Ansible marks this task with **ok** instead of **changed**.

Simple Playbook

- Dynamic files can include e.g.:
  - variables
  - loop
  - conditions
- These files are copied with the **ansible.builtin.template** module
- Ensure to deploy **index.html** page on your host

```
$ <EDITOR> web-notls.yml
```

Simple Playbook – Red Hat/Debian

```
...

    - name: Copy index.html
      ansible.builtin.template:
        src: templates/index.html.j2
        dest: /var/www/html/index.html
        mode: 0644
```

Simple Playbook – SUSE

```
...

    - name: Copy index.html
      ansible.builtin.template:
        src: templates/index.html.j2
        dest: /srv/www/htdocs/index.html
        mode: 0644
```

Simple Playbook

- Templates should be placed in folder **templates**

```
$ mkdir templates
$ <EDITOR> templates/index.html.j2

<html>
  <head>
    <title>Ansible workshop</title>
  </head>
  <body>
    <h1>Ansible workshop</h1>
    <p>This is our first web application, created in
Ansible workshop!</p>
    <p>{{ ansible_managed }}</p>
  </body>
</html>
```

Simple Playbook

- Templates should be placed in folder **templates**
- File ending **.j2** is optional but strongly recommended
- More information about Jinja2 templating engine: https://jinja.palletsprojects.com/en/3.0.x/

It’s not required to study full Jinja documentation as only a couple of options are required.

Simple Playbook

- Deploy dynamic webpage

```
$ ansible-playbook -i hosts web-notls.yml

...
TASK [Copy index.html]
**********************************************************

changed: [testserver]
...
```

Simple Playbook

- Services are started with **ansible.builtin.service** module

```
$ <EDITOR> web-notls.yml
```

ansible.builtin.service is a proxy for specific service manager modules like ansible.builtin.systemd or ansible.builtin.sysvinit

Simple Playbook – Red Hat

```
...

    - name: Start and enable Apache
      ansible.builtin.service:
        name: httpd
        state: started
        enabled: true
```

Simple Playbook – Debian/SUSE

```
...

    - name: Start and enable Apache
      ansible.builtin.service:
        name: apache2
        state: started
        enabled: true
```

LAB 5: Configure firewall

LAB 5: Configure firewall

- Ensure to properly configure firewall by enabling the following ports:
  - 22
  - 80
- Alternativly configure the following services:
  - ssh
  - http

LAB 5: Configure firewall – Red Hat / SUSE

- RHEL / CentOS / AlmaLinux / RockyLinux as well as SLES 15 / openSUSE uses firewalld per default
- Ensure that firewalld is installed
- Ensure that firewalld is running
- There’s a firewalld module available in collection ansible.posix – not in ansible.builtin!

LAB 5: Configure firewall - Ubuntu

- Debian / Ubuntu uses ufw per default
- Ensure that ufw is installed
- Ensure that ufw is running
- There’s a ufw module available in collection community.general – not in ansible.builtin!

LAB 5: Configure firewall – Red Hat

Solution

```
...
    - name: Install firewalld
      ansible.builtin.dnf:
        name: firewalld
        state: present


    - name: Start and enable firewalld
      ansible.builtin.service:
        name: firewalld
        state: started
        enabled: true
```

LAB 5: Configure firewall – Red Hat

- Solution

```
...
    - name: Open service ssh in firewalld
      ansible.posix.firewalld:
        service: ssh
        state: enabled
        permanent: true
        immediate: true

    - name: Open service http in firewalld
      ansible.posix.firewalld:
        service: http
        state: enabled
        permanent: true
        immediate: true
```

LAB 5: Configure firewall – Ubuntu

Solution

```
...
    - name: Install ufw
      ansible.builtin.apt:
        name: ufw
        update_cache: true
        state: present

    - name: Start and enable ufw
      ansible.builtin.service:
        name: ufw
        state: started
        enabled: true
```

LAB 5: Configure firewall – Ubuntu

- Solution

```
...
    - name: Enable service ssh in ufw
      community.general.ufw:
        port: ssh
        proto: tcp
        rule: allow

    - name: Enable service http in ufw
      community.general.ufw:
        port: http
        proto: tcp
        rule: allow

    - name: Enable ufw and reject everything
      community.general.ufw:
        state: enabled
        policy: reject
```

LAB 5: Configure firewall – SUSE

Solution

```
...
    - name: Install firewalld
      community.general.zypper:
        name: firewalld
        state: present

    - name: Start and enable firewalld
      ansible.builtin.service:
        name: firewalld
        state: started
        enabled: true
```

LAB 5: Configure firewall – SUSE

- Solution

```
...
    - name: Open service ssh in firewalld
      ansible.posix.firewalld:
        service: ssh
        state: enabled
        permanent: true
        immediate: true

    - name: Open service http in firewalld
      ansible.posix.firewalld:
        service: http
        state: enabled
        permanent: true
        immediate: true
```

LAB 5: Configure firewall

Solution

Playbooks
YAML Syntax

YAML syntax

- YAML is a human-friendly data serialization language for all programming languages
- This chapter provides more information about YAML

YAML syntax

- Playbook header
  - A YAML file should start with **---** as a header
  - Ansible doesn’t really care about this :)

```
---

- name: My first play
...
```

YAML syntax

- Comments
  - Like many other languages, YAML uses the # sign for comments
  - Most of the time no comments are required, as all tasks should be named

```
# I am a comment :)
```

YAML syntax

- Quoting / String
  - Ansible interprets nearly everything as a string, so no quoting is required except for special cases like variables (more later)

```
- name: "Install {{ package_name }}"
  ansible.builtin.dnf:
    name: "{{ package_name }}"
    state: present
```

YAML syntax

- Booleans
  - Ansible accepts broad range for booleans, where matching strings are case insensitive

```
True | 'true' | 't' | 'yes' | 'y' | 'on'
| '1' | 1 | 1.0 => true

False | 'false' | 'f' | 'no' | 'n' |
'off' | '0' | 0 | 0.0 => false
```

YAML syntax

- Booleans
  - Ansible accepts broad range for booleans, where matching strings are case insensitive

Tip: use only **true** and **false**, as these values are expected by yamllint for booleans.

YAML syntax

- Lists
  - Lists start with a **-**
  - You’re already using lists of tasks and lists of plays
  - Install a list of packages (if the module supports it):

```
ansible.builtin.dnf:
  name:
    - httpd
    - mod_ssl
  state: present
```

YAML syntax

- Lists
  - Consult module documentation if a list is supported

name aliases: pkg list / elements=string	A package name or package specifier with version, like `name-1.0`. When using state=latest, this can be '*' which means run: dnf -y update. You can also pass a url or a local path to an rpm file. To operate on several packages this can accept a comma separated string of packages or a list of packages. Comparison operators for package version are valid here `>`, `<`, `>=`, `<=`. Example - `name >= 1.0`. Spaces around the operator are required. You can also pass an absolute path for a binary which is provided by the package to install. See examples for more information. **Default:** `[]`

YAML syntax

- Lists
  - Lists are indexed by numbers

```
ansible.builtin.debug:
  var: my_list[0]
vars:
  my_list: ['foo', 'bar']

...
TASK [ansible.builtin.debug]
**************************************************

ok: [testserver] => {
    "my_list[0]": "foo"
}
```

YAML syntax

- Dictionaries
  - Equivalent to hashes
  - Differ from a list as they are keyed using a string instead of a number

```
vars:
  package_name: httpd
```

YAML syntax

- Dictionaries

```
ansible.builtin.debug:
  var: my_list.foo
vars:
  my_list:
    foo: bar


...
TASK [ansible.builtin.debug]
**********************************************************

ok: [testserver] => {

    "my_list.foo": "bar"
}
```

YAML syntax

- Line breaks
  - Literal with **|**
  - Folded with **>**

YAML syntax

- Line breaks - literal

```
ansible.builtin.debug:
  msg: |

    First line
    Second line
    Third line


...
TASK [ansible.builtin.debug]
*******************************************************
ok: [testserver] => {

    "msg": "First line\nSecond line\nThird line\n"

}
```

YAML syntax

- Line breaks - folded

```
ansible.builtin.debug:
  msg: >

    One long
    line with
    line breaks


...
TASK [ansible.builtin.debug]
**********************************************************
ok: [testserver] => {

    "msg": "One long line with line breaks"

}
```

Playbooks
Loops and conditions

Loops and conditions

- Often required to run tasks under certain conditions
- Repeat multiple actions – like configuring firewall ports
- Loops are realized with **loop** option
- https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_loops.html

Earlier versions of Ansible used with_* instead of loop, but this is deprecated now.

Loops and conditions

- Simplest way is a loop over lists
- Each element of the list is referenced by **item** keyword
- As item is a variable, it needs to be declared as a variable
- Install Apache and mod_ssl extension

Loops and conditions – Red Hat

```
...

  tasks:
    - name: Install Apache and mod_ssl
      ansible.builtin.dnf:
        name: "{{ item }}"
        state: present
      loop:
        - httpd
        - mod_ssl
```

Loops and conditions – Debian

```
...

  tasks:
    - name: Install Apache and mod_ssl
      ansible.builtin.apt:
        name: "{{ item }}"
        update_cache: true
        state: present
      loop:
        - apache2
        - libapache2-mod-ssl
```

Loops and conditions – SUSE

```
...

  tasks:
    - name: Install Apache and mod_ssl
      community.general.zypper:
        name: "{{ item }}"
        state: present
      loop:
        - apache2
```

Loops and conditions

- These examples are just to demonstrate loops – if a module supports lists natively, use it!
- Better code for installing packages:

```
- name: Install Apache and mod_ssl
  ansible.builtin.dnf:
    name:
      - httpd
      - mod_ssl
    state: present
```

LAB 6: Improve firewall configuration

LAB 6: Improve firewall configuration

- Ensure to properly configure firewall by enabling the following ports:
  - 22
  - 80
- Alternativly configure the following services:
  - ssh
  - http
- Use a list to avoid multiple tasks

LAB 6: Improve firewall configuration – Red Hat / SUSE

- Solution

```
...
    - name: Open service ssh in firewalld
      ansible.posix.firewalld:
        service: "{{ item }}"
        state: enabled
        permanent: true
        immediate: true
      loop:
        - ssh
        - http
```

LAB 6: Improve firewall configuration – Ubuntu

- Solution

```
...
    - name: Enable service ssh in ufw
      community.general.ufw:
        port: "{{ item }}"
        proto: tcp
        rule: allow
      loop:
        - ssh
        - http

    - name: Enable ufw and reject everything
      community.general.ufw:
        state: enabled
        policy: reject
```

Loops and conditions

- Often required to loop over dictionaries, as simple lists aren’t sufficient
- Use loop with **dict2items** filter

Loops and conditions

- Dictionary example

```
...
    - name: Dictionary example
      ansible.builtin.debug:
        msg: "{{ item.key }} - {{ item.value }}"
      loop: "{{ tag_data | dict2items }}"
      vars:
        tag_data:
          environment: dev
          application: web
```

Loops and conditions

- Conditions are realized with **when** clause
- Possible to combine conditions with **and** keywork or creating a **list**
- Use **or** if conditions should not be combined
- When expects variables, so don’t declare them, but quote strings!

Loops and conditions

- When example

```
...
    - name: Install VMware tools
      ansible.builtin.dnf:
        name: open-vm-tools
        state: present
      when: ansible_virtualization_type == "VMware"
```

More information about (special) variables can be found in the next chapter...

Loops and conditions

- When **and** example

```
...
    - name: Install VMware tools
      ansible.builtin.dnf:
        name: open-vm-tools
        state: present
      when:
        - ansible_virtualization_type == "VMware"
        - ansible_os_family == "RedHat"
```

More information about (special) variables can be found in the next chapter...

Loops and conditions

- When **or** example

```
...
    - name: Open service ssh in firewalld
      ansible.posix.firewalld:
        service: "{{ item }}"
        state: enabled
        permanent: true
        immediate: true
      loop:
        - ssh
        - http
      when: >
        ansible_os_family == "Suse" or
        ansible_os_family == "RedHat"
```

Variables and facts

Variables and facts

- Many possibilities to define custom variables
- Ansible has built-in variables
- Facts are specific variables for a host
- Always quote variables except in when clause
- https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_variables.html

Variables and facts

- Facts are gathered as first task of playbooks
- Disable facts gathering:

```
---
- name: Playbook without facts
  hosts: webservers
  become: true
  gather_facts: false
  tasks:
...
```

Variables and facts

- If facts are enabled, a task is shown:

```
PLAY [Configure Apache webserver] **********************************************

TASK [setup] *******************************************************************

ok: [testserver]
```

Variables and facts

- Gather facts with Ad-hoc command

```
$ ansible -i hosts -m setup testserver
```

```
testserver | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "192.168.121.65"
        ],
        "ansible_all_ipv6_addresses": [
            "fe80::5054:ff:fede:7d69"
        ],
...
```

LAB 7: Facts

LAB 7: Facts

- Find out which facts are available for your test system

LAB 7: Facts

- Solution

```
$ ansible -i hosts -m setup testserver
```

```
testserver | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "192.168.121.65"
        ],
        "ansible_all_ipv6_addresses": [
            "fe80::5054:ff:fede:7d69"
        ],
...
```

Variables and facts

- Variables in Ansible tasks need to be quoted with **"{{ variable_name }}"** or **'{{ variable_name }}'**
- There are many possibilities for declaring variables (inventory, play, task,...)
- All variables are global
- Example: play variable

Variables and facts

- Define variable for name of static.html file

```
---
- name: Configure Apache webserver
  hosts: webservers
  become: true

  vars:
    static_file: static.html

  tasks:
...
```

Variables and facts

- Define variable for name of static.html file

```
---
- name: Configure Apache webserver
  hosts: webservers
  become: true

  vars:
    static_file: static.html

  tasks:
...
```

Variables and facts – Red Hat / Debian

- Define variable for name of static.html file

```
...
    - name: Copy static.html
      ansible.builtin.copy:
        src: "files/{{ static_file }}"
        dest: "/var/www/html/{{ static_file }}"
...
```

Variables and facts – SUSE

- Define variable for name of static.html file

```
...
    - name: Copy static.html
      ansible.builtin.copy:
        src: "files/{{ static_file }}"
        dest: "/srv/www/htdocs/{{ static_file }}"
...
```

Variables and facts

- Possible to set variable during runtime with **ansible.builtin.set_fact** module

```
...
    tasks:
      - name: Define webroot for RHEL/Debian
        ansible.builtin.set_fact:
          web_root: /var/www/html
        when: >
          ansible_os_family == "Debian" or
          ansible_os_family == "RedHat"
...
```

Variables and facts

- Possible to set variable during runtime with **ansible.builtin.set_fact** module

```
...
    - name: Define webroot for RHEL/Debian
      ansible.builtin.set_fact:
        web_root: /var/www/html
      when: >
        ansible_os_family == "Debian" or
        ansible_os_family == "RedHat"

    - name: Define webroot for SUSE
      ansible.builtin.set_fact:
        web_root: /srv/www/htdocs
      when: ansible_os_family == "Suse"
...
```

Variables and facts

- Possible to set variable during runtime with **ansible.builtin.set_fact** module

```
...
    - name: Copy static.html
      ansible.builtin.copy:
        src: "{{ web_root }}/{{ static_file }}"
        dest: "{{ web_root }}/{{ static_file }}"
...
```

Variables and facts

- Ansible has some built-in variables which can be very helpful
- See https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_vars_facts.html#information-about-ansible-magic-variables

If you try to override one of these variables, no error is displayed, but variable want be overridden!

Variables and facts

- Magic variables:
  - hostvars
  - groups
  - group_names
  - inventory_hostname
  - ansible_play_hosts
  - ansible_play_batch
  - ansible_playbook_python
  - inventory_dir
  - inventory_file
  - playbook_dir
  - role_path
  - ansible_check_mode
  - ansible_version

Variables and facts

- Variables in Templates aren’t quoted, but double brackets **{{ variable_name }}** are still required!
- We already used special variable **ansible_managed** in index.html.j2

LAB 8: Variables

LAB 8: Variables

- Add inventory hostname to yout index.html template and make sure it’s displayed by your webserver.

```
$ <EDITOR> templates/index.html.j2
```

LAB 8: Variables

- Add inventory hostname to index.html

```
$ <EDITOR> templates/index.html.j2

<html>
  <head>
    <title>Ansible workshop</title>
  </head>
  <body>
    <h1>Ansible workshop</h1>
    <p>This is our first web application, created in
Ansible workshop!</p>
    <p>{{ ansible_managed }}</p>
    <p>Hostname: {{ inventory_hostname }}</p>
  </body>
</html>
```

Variables and facts

- Variables can be defined at 22 possible locations
- Don’t use all possibilities :)
- Preceedence can’t be changed
- See https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_variables.html#variable-precedence-where-should-i-put-a-variable

Variable preceedence

1) Command line (-u user)
2) Role defaults
3) Inventory file or script group vars
4) Inventory group_vars/all
5) Playbook group_vars/all
6) Inventory group_vars/*
7) Playbook group_vars/*
8) Inventory file or script host vars
9) Inventory host_vars/*
10) Playbook host_vars/*
11) Host facts / cached set_facts
12) Play vars
13) Play vars_prompt
14) Play vars_files
15) Role vars
16) Block vars
17) Task vars
18) include_vars
19) Set facts / registered vars
20) Role params
21) Include params
22) Extra vars (-e „user=user“)

Inventories

Inventories

- Ansible uses no agent, so an inventory is required to define possible target
- 2 types of inventories:
  - Static inventories
  - Dynamic inventories
- Static inventories are most commonly in format:
  - INI
  - YAML

Inventories

- Inventory is used for:
  - Defining hosts
  - Defining groups
  - Adding hosts to groups
  - Adding connection parameters for hosts

Inventories

- Sample inventory in INI format

```
$ <EDITOR> hosts

[webservers]
testserver ansible_connection=local
```

Inventories

- Sample inventory in YAML format

```
$ <EDITOR> hosts.yml
```

```
webservers:
  hosts:
    testserver:
      ansible_connection: local
```

Inventories

- A host can be in mulitple groups
- Groups can be nested
- Special group **all** doesn’t need to be defined in inventory
- Use as many groups as you like!

Inventories

- Adding development group

```
$ <EDITOR> hosts

[development:children]
webservers

[webservers]
testserver ansible_connection=local
```

Inventories

- Adding development group

```
$ <EDITOR> hosts.yml


development:
  children:
    webservers
webservers:
  hosts:
    testserver:
      ansible_connection: local
```

Inventories

- Possible to use wildcards for hostnames
- Imho no real practical use case...

```
[webservers]
testserver[01:09].example.com
```

```
webservers:
  hosts:
    testserver[01:09].example.com
```

Inventories

- Aliases for human readable hostnames can be used
- Connection parameters can be overriden
- Per default Ansible uses Linux hostname resolution:
  - /etc/hosts
  - DNS
  - systemd-resolved
  - ssh_config
  - ...

Inventories

- Aliases / Connection Parameters

```
[webservers]
testserver1 ansible_host=192.168.0.1
testserver2 ansible_host=192.168.0.2
testserver3
testserver4 ansible_user=root
testserver5 ansible_connection=local
testserver6 ansible_host= 192.168.0.6 ansible_user=root
```

Inventories

- Host- and group vars can be defined in inventory
- Best practice is using host_vars and group_vars directory
- Example: define static_file variable in group_vars/webservers.yml file

Inventories

- Example: define static_file variable in group_vars/webservers.yml file

```
$ mkdir group_vars
$ <EDITOR> group_vars/webservers.yml
```

```
---
static_file: static.html
```

Inventories

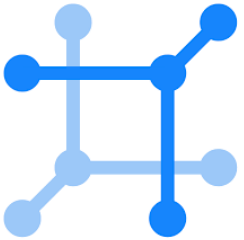
- Example: define static_file variable in group_vars/webservers.yml file

```
$ <EDITOR> web-notls.yml

---
- name: Configure Apache webserver
  hosts: webservers
  become: true

  vars:
    static_file: static.html

  tasks:
```

Inventories

- Dynamic inventory for bigger setups
- Use inventory plugin to fetch information

```
$ ansible-doc -t inventory -l
```

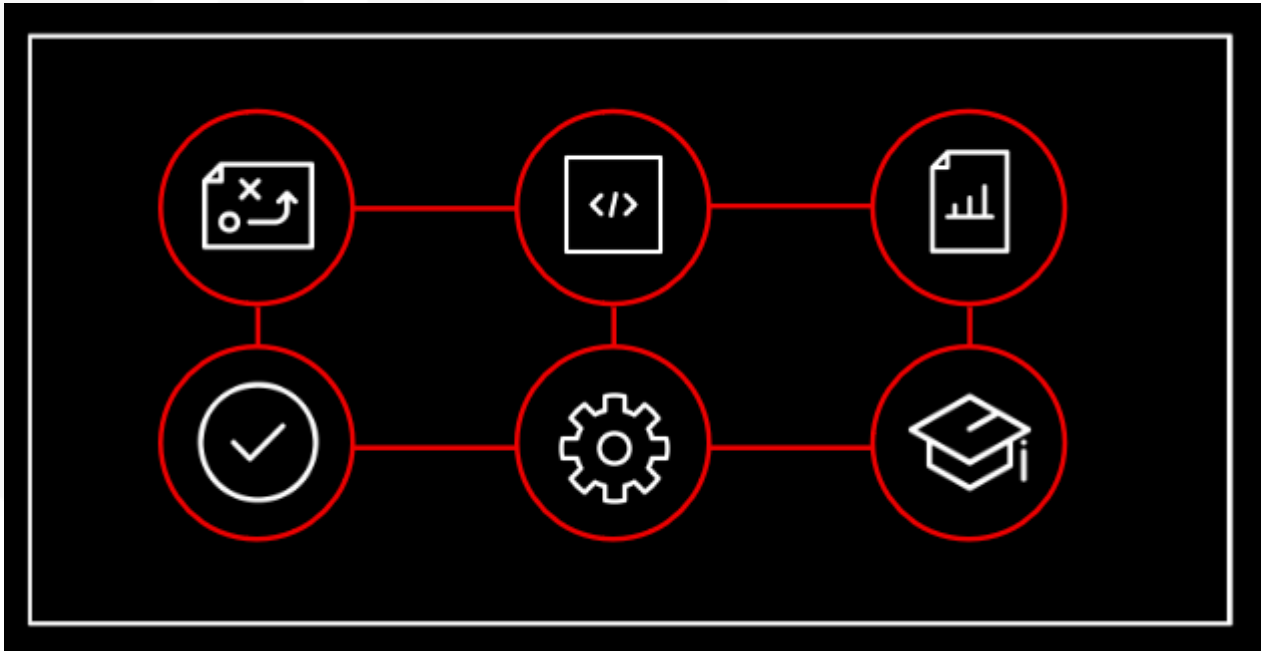


vmware®
by Broadcom

Checkmk collection

Collections

- Simplified and consistent content delivery
- Collections contains:
  - Modules
  - Playbooks
  - Roles
  - Plugins
  - Docs
  - Tests



Source: https://github.com/ansible/workshops/blob/devel/decks/ansible_rhel.pdf

Checkmk Collection

- Developed by Checkmk team + community
- Documentation:
  https://galaxy.ansible.com/ui/repo/published/checkmk/general/
- Git Repository:
  https://github.com/Checkmk/ansible-collection-checkmk.general

Checkmk Collection

- Install collection with ansible-galaxy

```
$ ansible-galaxy collection install checkmk.general
$ ansible-galaxy collection list | grep checkmk

checkmk.general          4.4.1
```

ansible-galaxy will download collection from galaxy.ansible.com if no private Automation hub is configured.

Checkmk Collection

- Sample playbooks are located in playbooks/demo folder

```
├── downtimes.yml
├── full.yml
├── groups.yml
├── hosts-and-folders.yml
├── lookup.yml
├── README.md
├── rules.yml
└── users-and-contacts.yml
```

Playbooks in collections are not intended pro production use!

Checkmk Collection

- Using the collection

```
---
- hosts: all
  tasks:
    - name: Activate changes
      checkmk.general.activation:
        server_url: "https://my_server"
        site: "my_site"
        automation_user: "my_user"
        automation_secret: "my_secret"
        sites: "my_site"
```

Checkmk Collection

- Roles for installing and configuring agents and servers:
  - agent
  - Server

You will most likely use this code only as a reference.

```
---
- hosts: all
  roles:
     - checkmk.general.server
```

checkmk

Checkmk collection Modules

Checkmk Collection

- Module: activation

```
---
- hosts: all
  tasks:
    - name: Activate changes
      checkmk.general.activation:
        server_url: "https://my_server"
        site: "my_site"
        automation_user: "my_user"
        automation_secret: "my_secret"
        force_foreign_changes: true
        sites: "my_site"
      run_once: true
```

Checkmk Collection

- Module: bakery

```
---
- hosts: all
  tasks:
    - name: Bake and sign all agents
      checkmk.general.bakery:
        server_url: "https://my_server"
        site: "my_site"
        automation_user: "my_user"
        automation_secret: "my_secret"
        signature_key_id: 1
        signature_key_passphrase: "my_key"
        state: baked_signed
```

Checkmk Collection

- Module: contact_group

```
---
- hosts: all
  tasks:
    - name: Create a single contact group
      checkmk.general.contact_group:
        server_url: "https://my_server"
        site: "my_site"
        automation_user: "my_user"
        automation_secret: "my_secret"
        name: "my_contact_group"
        title: "My Contact Group"
        customer: "provider"
        state: present
```

Checkmk Collection

- Module: discovery

```
---
- hosts: all
  tasks:
    - name: Add newly discovered services on host
      checkmk.general.discovery:
        server_url: "https://my_server"
        site: "my_site"
        automation_user: "my_user"
        automation_secret: "my_secret"
        host_name: "my_host"
        name: "my_contact_group"
        state: new
```

Checkmk Collection

- Module: downtime

```
---
- hosts: all
  tasks:
    - name: Schedule host downtime
      checkmk.general.downtime:
        server_url: "https://my_server"
        site: "my_site"
        automation_user: "my_user"
        automation_secret: "my_secret"
        host_name: "my_host"
        name: "my_contact_group"
        start_after:
          minutes: 5
        end_after:
          days: 7
          hours: 5
```

Checkmk Collection

- Module: folder

```
---
- hosts: all
  tasks:
    - name: Create a single folder
      checkmk.general.folder:
        server_url: "https://my_server"
        site: "my_site"
        automation_user: "my_user"
        automation_secret: "my_secret"
        path: "/my_folder"
        name: "My Folder"
        state: present
```

Checkmk Collection

- Module: host

```
---
- hosts: all
  tasks:
    - name: Create a host
      checkmk.general.host:
        server_url: "https://my_server"
        site: "my_site"
        automation_user: "my_user"
        automation_secret: "my_secret"
        name: "my_host"
        folder: "/"
        state: present
```

Checkmk Collection

- Module: host_group

```
---
- hosts: all
  tasks:
    - name: Create a single host group
      checkmk.general.host_group:
        server_url: "https://my_server"
        site: "my_site"
        automation_user: "my_user"
        automation_secret: "my_secret"
        name: "my_host_group"
        title: "My Host Group"
        customer: "provider"
        state: present
```

Checkmk Collection

- Module: password

```
---
- hosts: all
  tasks:
    - name: Delete a password
      checkmk.general.password:
        server_url: "https://my_server"
        site: "my_site"
        automation_user: "my_user"
        automation_secret: "my_secret"
        name: "my_password"
        state: absent
```

Checkmk Collection

- Module: rule

```
---
- hosts: all
  tasks:
    - name: Delete the first rule
      checkmk.general.rule:
        server_url: "https://my_server"
        site: "my_site"
        automation_user: "my_user"
        automation_secret: "my_secret"
        ruleset: "checkgroup_parameters:memory_percentage_used"
        rule:
          rule_id: "{{ response.content.id }}"
        state: absent
```

Checkmk Collection

- Module: service_group

```
---
- hosts: all
  tasks:
    - name: Create a single service group
      checkmk.general.service_group:
        server_url: "https://my_server"
        site: "my_site"
        automation_user: "my_user"
        automation_secret: "my_secret"
        name: "my_service_group"
        title: "My Service Group"
        customer: "provider"
        state: present
```

Checkmk Collection

- Module: tag_group

```
---
- hosts: all
  tasks:
    - name: Create a tag group
      checkmk.general.tag_group:
        server_url: "https://my_server"
        site: "my_site"
        automation_user: "my_user"
        automation_secret: "my_secret"
        name: "datacenter"
        title: "Dataccenter"
        topic: "Tags"
        tags:
          - id: datacenter_1
            title: "Datacenter 1"
        state: present
```

Checkmk Collection

- Module: timeperiod

```
---
- hosts: all
  tasks:
    - name: Delete a timeperiod
      checkmk.general.timeperiod:
        server_url: "https://my_server"
        site: "my_site"
        automation_user: "my_user"
        automation_secret: "my_secret"
        name: "worktime"
        state: absent
```

Checkmk Collection

- Module: user

```
---
- hosts: all
  tasks:
    - name: Delete a user
      checkmk.general.user:
        server_url: "https://my_server"
        site: "my_site"
        automation_user: "my_user"
        automation_secret: "my_secret"
        name: "my_user"
        state: absent
```

Checkmk collection Plugins

Checkmk Collection

- Lookup plugins: bakery

```
---
- hosts: all
  tasks:
    - name: Show bakery status
      ansible.builtin.debug:
        msg: "Bakery status is {{ bakery }}"
      vars:
        bakery: "{{ lookup('checkmk.general.bakery',
                           server_url=https://my_server,
                           site=my_site,
                           automation_user=my_user,
                           automation_secret=my_secret
                         ) }}"
```

Checkmk Collection

- Lookup plugins: folder

```
---
- hosts: all
  tasks:
    - name: Get attributes of folder
      ansible.builtin.debug:
        msg: "Attributes of folder: {{ attributes }}"
      vars:
        attributes: "{{ lookup('checkmk.general.folder',
                        'my_folder',
                        server_url=https://my_server,
                        site=my_site,
                        automation_user=my_user,
                        automation_secret=my_secret
                      ) }}"
```

Checkmk Collection

- Lookup plugins: folders

```
---
- hosts: all
  tasks:
    - name: Get all subfolders of the main folder
      ansible.builtin.debug:
        msg: "Folder tree: {{ item.id }}"
      loop: "{{ lookup('checkmk.general.folders',
                       '~',
                       show_hosts=false,
                       recursive=true,
                       server_url=https://my_server,
                       site=my_site,
                       automation_user=my_user,
                       automation_secret=my_secret
                      ) }}"
```

Checkmk Collection

- Lookup plugins: host

```
---
- hosts: all
  tasks:
    - name: Get attributes of host
      ansible.builtin.debug:
        msg: "Attributes of host: {{ attributes }}"
      vars:
        attributes: "{{ lookup('checkmk.general.host',
                        'my_host',
                        effective_attributes=true,
                        server_url=https://my_server,
                        site=my_site,
                        automation_user=my_user,
                        automation_secret=my_secret
                      ) }}"
```

Checkmk Collection

- Lookup plugins: hosts

```
---
- hosts: all
  tasks:
    - name: Get all hosts
      ansible.builtin.debug:
        msg: "Host: {{ item.id }}, IP:
{{ item.extensions.effective_attributes.ipaddress }}"
      loop: "{{ lookup('checkmk.general.hosts',
                       effective_attributes=true,
                       server_url=https://my_server,
                       site=my_site,
                       automation_user=my_user,
                       automation_secret=my_secret
                    ) }}"
```

Checkmk Collection

- Lookup plugins: rule

```
---
- hosts: all
  tasks:
    - name: Get a rule with a particular rule id
      ansible.builtin.debug:
        msg: "Rule: {{ attributes }}"
      vars:
        attributes: "{{ lookup('checkmk.general.rule',
                        rule_id='a9285bc1-dcaf-45e0-a3ba-
ad398ef06a49',
                        server_url=https://my_server,
                        site=my_site,
                        automation_user=my_user,
                        automation_secret=my_secret
                      ) }}"
```

Checkmk Collection

- Lookup plugins: rules

```
---
- hosts: all
  tasks:
    - name: Get a list of rules of ruleset host_groups
      ansible.builtin.debug:
        msg: "Rule: {{ item.extensions }}"
      loop: "{{ lookup('checkmk.general.rules',
                       ruleset='host_groups',
                       server_url=https://my_server,
                       site=my_site,
                       automation_user=my_user,
                       automation_secret=my_secret
                     ) }}"
```

Checkmk Collection

- Lookup plugins: ruleset

```
---
- hosts: all
  tasks:
    - name: Get details about a particular ruleset
      ansible.builtin.debug:
        msg: "Ruleset: {{ item.extensions }}"
      loop: "{{ lookup('checkmk.general.ruleset',
                       ruleset='host_groups',
                       server_url=https://my_server,
                       site=my_site,
                       automation_user=my_user,
                       automation_secret=my_secret
                     ) }}"
```

Checkmk Collection

- Lookup plugins: rulesets

```
---
- hosts: all
  tasks:
    - name: Get all used rulesets
      ansible.builtin.debug:
        msg: "Ruleset: {{ item.extensions.name }} has
{{ item.extensions.number_of_rules }} rules"
      loop: "{{ lookup('checkmk.general.rulesets',
                       regex='',
                       rulesets_used=true,
                       server_url=https://my_server,
                       site=my_site,
                       automation_user=my_user,
                       automation_secret=my_secret
                      ) }}"
```

Checkmk Collection

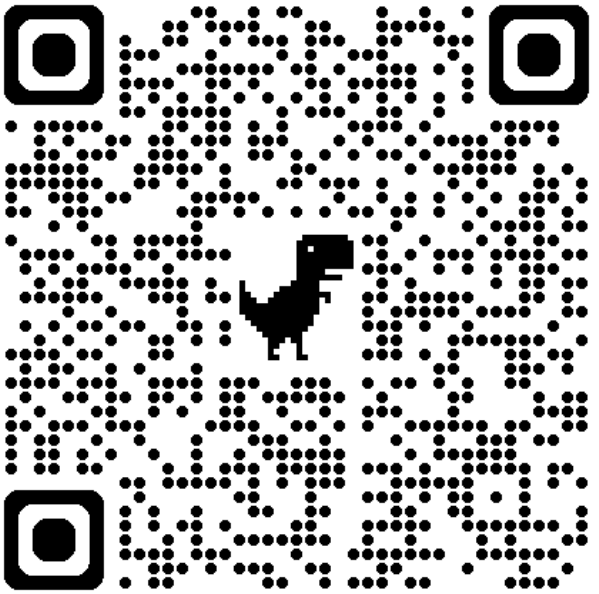
- Lookup plugins: version

```
---
- hosts: all
  tasks:
    - name: Show Checkmk version
      ansible.builtin.debug:
        msg: "Server version is {{ version }}"
      vars:
        version: "{{ lookup('checkmk.general.version',
                            server_url=https://my_server,
                            site=my_site,
                            automation_user=my_user,
                            automation_secret=my_secret
                           ) }}"
```

Feedback

Feedback

- https://forms.gle/r9CmPfSx46GKZmNZ7



Thanks a lot for your participation

DI (FH) René Koch
Freelancer
The Checkmk Conference #10, 13.06.2024