Ansible Playbooks

- ~~Ansible Introduction~~
- ~~Ansible Concepts~~
- Ansible Playbooks
- Plugins & Dynamic inventories
- Playbooks Advanced
- Ansible Roles
- AWX / Ansible Tower Overview

©2019 edc4it B.V.

(c) 2019 edc4it B.V.

Playbook Introduction

(c) 2019 edc4it B.V.

YAML

A Playbook is typically a YAML file

Slide Deck

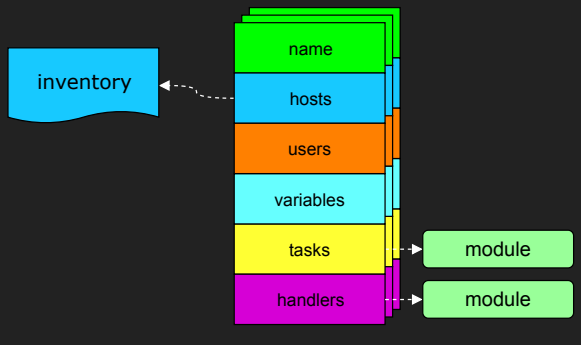
(c) 2019 edc4it B.V.

A book with plays

Each play consists of tasks

(c) 2019 edc4it B.V.

Play structure



(c) 2019 edc4it B.V.

Breakdown of sections

Example

```yaml
---
- name: Configure Apache httpd
  hosts: tv-series-web-servers
  become: true
  tasks:
  - name: ensure https is at the latest version
    package:
      name: httpd
      state: present
  …
  - name: ensure apache is running
    service:
      name: httpd
      state: started
```


(c) 2019 edc4it B.V.

Multiple plays

```
---
- name: Configure Apache httpd
  hosts: tv-series-web-servers
  become: true
  tasks:
    ...
- name: Configure Postgres
  hosts: tv-series-db-servers
  become: true
  become_user: postgres
  tasks:
    ...
```


(c) 2019 edc4it B.V.

Hosts

(c) 2019 edc4it B.V.

Example

```
---
- hosts: db-servers
  ...
```

Will be ordered by *inventory*

(c) 2019 edc4it B.V.

Setting ordering

```
---
- hosts: db-servers
  order:
  ...
```

`inventory`, `reverse_inventory`, `sorted`, `reverse_sorted`, `shuffle`

(c) 2019 edc4it B.V.

Gathering facts

```
---
- hosts: db-servers
  gather_facts: false
  ...
```

By default `true/yes`

(c) 2019 edc4it B.V.

Tasks

(c) 2019 edc4it B.V.

A play contains one or more tasks

a named call to a module with arguments

(c) 2019 edc4it B.V.

Task list

- in order
- one-at-a-time for all machines
- then proceed to next task

This behaviour can be changed through strategies (explained later)

(c) 2019 edc4it B.V.

Failures

machines with failed tasks are taken out of rotation

Errors can be managed (explained later)

(c) 2019 edc4it B.V.

Example

```
tasks:
  - name: Ensure docker is installed
    package:
      name: {{docker_package_name}}
  - name: make sure docker is restarted and enabled
    service:
      name: docker
      state: restarted
      enabled: true
```


(c) 2019 edc4it B.V.

Modules overview

(c) 2019 edc4it B.V.

Categories

- Cloud modules
- Clustering modules
- Commands modules
- Crypto modules
- Database modules
- Files modules
- Identity modules
- Inventory modules
- Messaging modules
- Monitoring modules
- Net Tools modules
- Network modules
- Notification modules
- Packaging modules
- Remote Management modules
- Source Control modules
- Storage modules
- System modules
- Utilities modules
- Web Infrastructure modules
- Windows modules

(c) 2019 edc4it B.V.

Getting help on a module

ansible-doc

```
$ ansible-doc …
```


(c) 2019 edc4it B.V.

command

```
- name: Ensure the database is initialised
  command: "/usr/bin/initdb -u scott -encoding UTF-8 -n recipesdb"
  args:
    chdir: /var/db/
    creates: /var/db/recipesdb # to make this task idempotent
```

Uses *free form* syntax for the command (and hence the `args`)

(c) 2019 edc4it B.V.

shell

runs in a shell (`/bin/sh`)

```
- name: Ensure security tokens are removed
  shell: "$HOME/cleanup.sh"
  args:
    removes: /etc/credentials/
```

Also uses *freeform*

(c) 2019 edc4it B.V.

Installing software

calls yum, dnf, apt, …

```
- name: Ensure docker is installed
  package:
    name: {{docker_package_name}}
    state: present # (set to absent to uninstall)
```

Also available: apt, yum, yumdnf …

(c) 2019 edc4it B.V.

Installing software

using a list

```
- name: Ensure docker is installed
  package:
    name:
      - docker
      - docker-compose
    state: present
```


(c) 2019 edc4it B.V.

service module

```
- name: make sure docker is running and enabled
  service:
    name: docker
    state: started
    enabled: true
```

Uses **`ansible_service_mgr`** fact (systemd, systemctl, service, …)

(c) 2019 edc4it B.V.

Non OS package modules

- bower
- bundler
- composer
- cpanm
- easy_install
- gems
- maven_artifact
- npm/yarn
- pear
- pip

(c) 2019 edc4it B.V.

File modules overview

- archive
- blockinfile
- copy
- fetch
- file
- find
- ini_file
- lineinfile
- patch
- replacein
- stat
- synchronize
- tempfile
- template
- unarchive
- xml
- ...

(c) 2019 edc4it B.V.

Copy files/directories

```yaml
- name: ensure index page is there
  copy:
    src: resources/index.html
    dest: /var/www/html
    owner: www
    group: www
    mode: 0644
```


(c) 2019 edc4it B.V.

Ensuring lines in files

```yaml
tasks:
- lineinfile:
    path: /etc/httpd/conf/httpd.conf
    regexp: '^#Listen '
    line: 'Listen 8080'
- lineinfile:
    path: /etc/sudoers
    state: absent
    regexp: '^%wheel
```


(c) 2019 edc4it B.V.

Block in files

```yaml
- name: Ensure proxy set for npm
  blockinfile:
    path: "{{ ansible_env.HOME }}/.npmrc"
    block: |
      proxy={{proxy_address}}
      https-proxy={{proxy_address}}
    marker: '# {mark} proxy settings'
    create: yes
    insertbefore: BOF
```

Often in conjunction with yaml literal style multi-line scalar (|)

(c) 2019 edc4it B.V.

Creating users/groups

```yaml
tasks:
  - name: Ensure the jboss group exists
    group:
      name: postgres
      state: present
  - name: Ensure jboss user is available
    user:
      name: jboss
      group: jboss
      home: /opt/jboss/
      password: $6$7S8WGecOpqPvO22p$yxwVjGZpAk8G4xQ6DWDtSaevEc4luXQcBhlIZ8YuW.qewY8yXkmuNG.u3ZU3taccKDOHKQLj2hiibdaq7Yj1
      append: true
```

Password needs to be salted (see next slide)

(c) 2019 edc4it B.V.

Password hasing

```
$ mkpasswd --method=sha-512
Password:
$6$78BWGecUgqFvOS2$p7yxwVjGZpAk8G6rQ6DNDtSsmvKr4luXQrBh1Z18YuW.qxw7ZyXkmoNG.u3ZU3tzccKDDHKQLj2hiibdaqJYj1
```

or use python (example is python3)

```
$ $ python -c 'import crypt,getpass; \
           print(crypt.crypt(getpass.getpass(), \
              crypt.mksalt(crypt.METHOD_SHA512)))'
```

For python make sure **passlib** is installed

(c) 2019 edc4it B.V.

Identity & Privilege escalation

(c) 2019 edc4it B.V.

overview

key	description
remote_user	SSH user
become	set to `yes` to escalate privilege
become_user	which user to *become* (root)
become_method	`su`, `sudo`, `pbrun`, …
become_flags	parameters to the `become_method`

might be used in conjunction with **`--ask-become-pass`**

(c) 2019 edc4it B.V.

example

```
---
- hosts: webservers
  remote_user: admin
  become: yes
  become_user: www
```


(c) 2019 edc4it B.V.

can also be set on tasks

```
---
- hosts: webservers
  tasks:
    - service:
        name: nginx
        state: started
      become: yes
      become_method: sudo
```


(c) 2019 edc4it B.V.

defaults are set using `ansible.cfg`

ini	env	default
become	ANSIBLE_BECOME	false
become_ask_pass,	ANSIBLE_BECOME_ASK_PASS	false
become_flags	ANSIBLE_BECOME_FLAGS	''
become_method	ANSIBLE_BECOME_METHOD	sudo
become_user	ANSIBLE_BECOME_USER	root

`[privilege_escalation]` section

(c) 2019 edc4it B.V.

Command Line Options

- --ask-become-pass,-K
- --become,-b
- --become-method=
- --become-user=

(c) 2019 edc4it B.V.

Variables & Facts in Playbooks

(c) 2019 edc4it B.V.

Variables in Playbook

example (recap)

```
---
- hosts: all
  user: vagrant
  vars:
    host: proxy01.example.com
    port_no: 8443
    proxy_address: http://{{host}}:{{port_no}}
  tasks:
  - name: Ensure proxy set in .npmrc
    blockinfile:
      path: "/home/vagrant/.npmrc"
      block: |
        proxy={{proxy_address}}
        https-proxy={{proxy_address}}
        …
```


(c) 2019 edc4it B.V.

External File

```
hosts: …
vars_files:
  - vars/myvars.yaml
```

vars/myvars.yaml:

```
host: proxy01.example.com
port_no: 8443
proxy_address: http://{{host}}:{{port_no}}
```


(c) 2019 edc4it B.V.

Command Line

-e/--extra-vars

```
- hosts: ""{{ env }}"

  …
```


(c) 2019 edc4it B.V.

Command Line

-e/--extra-vars

```
- hosts: ""{{ env }}"

  …
```

```
$ ansible-playbook -e "env=localhost" playbook.yml
```


(c) 2019 edc4it B.V.

Command Line

-e/--extra-vars

```
- hosts: ""{{ env }}"
  vars:
    port_no: 8443
  …
```

```
$ ansible-playbook -e "env=localhost" playbook.yml
```

Can also use yaml/json

(c) 2019 edc4it B.V.

Command Line

-e/--extra-vars

```
- hosts: ""{{ env }}"
  vars:
    port_no: 8443
  …
```

```
$ ansible-playbook -e "env=localhost" playbook.yml

$ ansible-playbook -e "env=localhost"
                   -e "port_no=8080" playbook.yml
```

Can also use yaml/json

(c) 2019 edc4it B.V.

Command Line

-e/--extra-vars

```
- hosts: ""{{ env }}"
  vars:
    port_no: 8443
  …
```

```
$ ansible-playbook -e "env=localhost" playbook.yml

$ ansible-playbook -e "env=localhost"
                   -e "port_no=8080" playbook.yml

$ ansible-playbook -e "-e "{'env':'test-machines'}"" playbook.yml
```

Can also use yaml/json

(c) 2019 edc4it B.V.

Command Line

-e/--extra-vars

```
- hosts: ""{{ env }}"
  vars:
    port_no: 8443
  …
```

```
$ ansible-playbook -e "env=localhost" playbook.yml

$ ansible-playbook -e "env=localhost"
                   -e "port_no=8080" playbook.yml

$ ansible-playbook -e "-e "{'env':'test-machines'}"" playbook.yml

$ ansible-playbook -e "@my-vars.json" playbook.yml
```

Can also use yaml/json

(c) 2019 edc4it B.V.

Prompting

```
- hosts:  localhost
  vars:
    proxy: http://{{username}}:{{password}}@{{host}}:{{port_no}}
    …
  vars_prompt:
  - name: username
    prompt: "proxy username"
    private: no
  - name: password
    prompt: "proxy password"
    private: yes
```


(c) 2019 edc4it B.V.

Register Variables

```
tasks:
- command: /usr/sbin/ip route list 10.0.2.0/24
  register: routeValue
- debug:
    var: routeValue
```


(c) 2019 edc4it B.V.

Setting facts using set_fact

```
tasks:
  - command: whoami
    register: woami_result
  - set_fact:
      user: "{{woami_result.stdout}}"
```


(c) 2019 edc4it B.V.

set_fact using Jinja2 functions

```
tasks:
  - command: /usr/sbin/ip route list 10.0.2.0/24
    register: route_result
  - set_fact:
      route_ip: "{{route_result.stdout.split()[8]}}"
```


(c) 2019 edc4it B.V.

Troubleshooting

(c) 2019 edc4it B.V.

Check syntax

--syntax-check

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


(c) 2019 edc4it B.V.

Check mode

-C/--check

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

Like a *dry-mode*

(c) 2019 edc4it B.V.

Diff

(together with check-mode)

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


(c) 2019 edc4it B.V.

DIFF
(together with check-mode)

```
$ ansible-playbook --check --diff mybook.yml
...
TASK [Configure the server] ***********************************
--- before: /etc/nginx/conf.d/server.conf (content)
+++ after: /etc/nginx/conf.d/server.conf (content)
@@ -0,0 +1,6 @@
+# BEGIN ANSIBLE MANAGED BLOCK
+server {
+       listen 80;
+       location / {
+       }
+  }
+# END ANSIBLE MANAGED BLOCK
```


(c) 2019 edc4it B.V.

verbose mode

```
$ ansible-playbook  my-book.yaml  --verbose
```

-v/--verbose, -vvv, -vvvv

(c) 2019 edc4it B.V.

debug module

```
- tasks:
  - name: Initialise the database
    shell: initdb
    register: result
  - debug:
      var: result
      verbosity: 3  # for -vvv, 0 by default
```

Or use `msg` with Jinja2 templates

(c) 2019 edc4it B.V.

stepping
one-step-at-a-time

```
$ ansible-playbook mybook.yaml --step
PLAY [mybook] task: TASK: Gathering Facts
  (N)o/(y)es/(c)ontinue:
```

At the same time open a ssh connection to investigate

(c) 2019 edc4it B.V.

stepping
one-step-at-a-time

```
$ ansible-playbook mybook.yaml --step
PLAY [mybook] task: TASK: Gathering Facts
  (N)o/(y)es/(c)ontinue: y
```

At the same time open a ssh connection to investigate

(c) 2019 edc4it B.V.

stepping
one-step-at-a-time

```
$ ansible-playbook mybook.yaml --step
PLAY [mybook] task: TASK: Gathering Facts
  (N)o/(y)es/(c)ontinue: y
ok: [db-servers]
Perform task: TASK: Ensure database is installed
(N)o/(y)es/(c)ontinue: |
```

At the same time open a ssh connection to investigate

(c) 2019 edc4it B.V.

Make error messages easier to read

```
ANSIBLE_STDOUT_CALLBACK=yaml  ansible-playbook mybook.yaml
```

Can also be set in `ansible.cfg` (`stdout_callback`)

More on this later

(c) 2019 edc4it B.V.

Lab 3

Playbooks

(c) 2019 edc4it B.V.

Handlers

(c) 2019 edc4it B.V.

Problem

- certain tasks require an action afterwards
- tasks are idempotent
- so how do we know if we need to perform the action?
- what if many tasks require the same action (e.g., reload a service?)

(c) 2019 edc4it B.V.

Handlers

- Called zero or one time
- only when notified one or more times
- at the end of the block
- in the order they are defined

(c) 2019 edc4it B.V.

Defining Handlers

```
tasks:
...
handlers:
- name: restart redis service
  service:
    name: redis
    state: restarted
- name: restart docker service
  service:
    name: docker
    state: restarted
```


(c) 2019 edc4it B.V.

Triggering Handlers by their *name*

```
tasks:
   - name: configure redis authentication
     ...
     notify: restart redis service
   - name: enable redis store
     ...
     notify: restart redis service
handlers:
...
```


(c) 2019 edc4it B.V.

Triggering Handlers using *listen*

```
tasks:
  - name: configure redis authentication
    ...
    notify: RESTART_REDIS
handlers:
  - name: restart redis service
    service:
      name: redis
      state: restarted
    listen: RESTART_REDIS
```

Decouples from handler's name

(c) 2019 edc4it B.V.

Triggering multiple handlers;

```
tasks:
  - name: configure redis authentication
    ...
    notify: RESTART_REDIS
handlers:
  - name: restart redis service
    ...
    listen: RESTART_REDIS
  - name: restart redis proxy
    ...
    listen: RESTART_REDIS
```


(c) 2019 edc4it B.V.

Triggering all handlers before block-end using a *meta* statement

```
tasks:
   - name: ...
     ...
   - name: ...
     ...
   - meta: flush_handlers
```

Triggers all notified handlers inside block so far

(c) 2019 edc4it B.V.

Lab 4

Handlers

(c) 2019 edc4it B.V.

Done!

- ~~Ansible Introduction~~
- ~~Ansible Concepts~~
- ~~Ansible Playbooks~~
- Plugins & Dynamic inventories
- Playbooks Advanced
- Ansible Roles
- AWX / Ansible Tower Overview

(c) 2019 edc4it B.V.