

ICA0002: IT Infrastructure Services

Ansible variables

Roman Kuchin
Juri Hudolejev
2025

Nginx + uWSGI example

#roles/nginx/files/default

```
server {  
    uwsgi_pass localhost:5000;  
}
```

#roles/nginx/tasks/main.yaml

- name: Nginx configuration
 ansible.builtin.copy:
 src: default
 dest: /etc/.../default

#roles/uwsgi/files/app.ini

```
[uwsgi]  
socket = localhost:5000;
```

#roles/nginx/tasks/main.yaml

- name: uWSGI configuration
 ansible.builtin.copy:
 src: app.ini
 dest: /etc/.../app.ini

Problems

1.

What if we need to change the port?

How to ensure that **all** needed files are changed?

Problems

1.

What if we need to change the port?

How to ensure that **all** needed files are changed?

2.

How to make this code support similar but **different** installations?

Server A: port 5001, server B: port 5002 etc.

Variables

Variable -- stored value identified by name

Generic example (not very Ansible related), set variable:

```
x = 42      # value: 42, name: x
```

Use variable:

```
print(x)    # addressed by name -- value (42) is printed
```

Ansible variables

Example, set:

```
uwsgi_addr: localhost:5000
```

Use:

```
socket = {{ uwsgi_addr }}
```

More: docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html

Why {{ ... }}?

{{ ... }} -- variable syntax used by Jinja (templating engine for Python)

Indicates that block is a variable name that should be replaced with a value:

<code>"x value is x".render(x=42)</code>	→	<code>"x value is x"</code>
<code>"x value is {{ x }}".render(x=42)</code>	→	<code>"x value is 42"</code>

We'll learn more about Jinja in the following labs

Jinja documentation: <https://jinja.palletsprojects.com>

What will Ansible do?

```
#roles/demo/files/demo.txt
```

```
x value is {{ x }}
```

```
#/opt/demo.txt
```

(What will we get here?)

```
#roles/demo/tasks/main.yaml
```

```
- name: Demo file
  ansible.builtin.copy:
    src: demo.txt
    dest: /opt/demo.txt
```


What will Ansible do?

```
#roles/demo/files/demo.txt
```

```
x value is {{ x }}
```

```
#/opt/demo.txt
```

```
x value is {{ x }}
```

```
#roles/demo/tasks/main.yaml
```

```
- name: Demo file
```

```
  ansible.builtin.copy:
```

```
    src: demo.txt
```

```
    dest: /opt/demo.txt
```

← 'copy' does NOT handle variables!

What will Ansible do?

```
#roles/demo/files/demo.txt
```

```
x value is {{ x }}
```

```
#/opt/demo.txt
```

(What will we get here?)

```
#roles/demo/tasks/main.yaml
```

```
- name: Demo file
  ansible.builtin.template:
    src: demo.txt
    dest: /opt/demo.txt
```

What will Ansible do?

```
#roles/demo/files/demo.txt
```

```
x value is {{ x }}
```

```
#roles/demo/tasks/main.yaml
```

- name: Demo file
 ansible.builtin.template:
 src: demo.txt
 dest: /opt/demo.txt

```
#/opt/demo.txt
```

(No changes)

Ansible will fail with his error:

Could not find or access 'demo.txt'
Searched in:

- roles/demo/templates/demo.txt
- ...

What will Ansible do?

```
#roles/demo/templates/demo.txt.j2    #/opt/demo.txt
```

```
x value is {{ x }}
```

```
x value is 42
```

```
#roles/demo/tasks/main.yaml
```

```
- name: Demo file
  ansible.builtin.template:
    src: demo.txt.j2
    dest: /opt/demo.txt
```

Nginx + uWSGI example

#roles/nginx/templates/default.j2

```
server {  
    uwsgi_pass {{ uwsgi_addr }};  
}
```

#roles/nginx/tasks/main.yaml

- name: Nginx configuration
 ansible.builtin.template:
 src: default.j2
 dest: /etc/.../default

#roles/uwsgi/templates/app.ini.j2

```
[uwsgi]  
socket = {{ uwsgi_addr }};
```

#roles/nginx/tasks/main.yaml

- name: uWSGI configuration
 ansible.builtin.template:
 src: app.ini.j2
 dest: /etc/.../app.ini

Files vs. templates

Ansible module **ansible.builtin.copy**:

- Uploads files from **files/** directory
- Files are uploaded as is, without modifications

Ansible module **ansible.builtin.template**:

- Uploads files from **templates/** directory
- Variables **{{ ... }}** are replaced with values on managed hosts

Ansible variables

Can be defined in multiple places:

- Command line
- Inventory file
- Separate **group_vars** files
- Separate **host_vars** files
- Plays
- Tasks
- Etc.

More: docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html

Ansible variables in the inventory file

```
elvis-1  ansible_host=193.40.157.25  ansible_port=9922  ...
```

```
{{ ansible_host }} → 193.40.157.25
```

```
{{ ansible_port }} → 9922
```


Ansible variables in this course

hosts

Host connection variables only

(`ansible_host`, `ansible_port` and `ansible_user`)

group_vars/all.yaml

All the other variables; should you need to change the service port or domain name on the defence -- only this file is modified and nothing more

No variables via command line!

Ansible variables

```
#r.../templates/my.conf.j2
```

```
x.user = {{ username }}  
x.pass = {{ password }}
```

```
#group_vars/all.yaml
```

```
username: elvis  
password: p4ssw0rd
```

```
#r.../tasks/main.yaml
```

```
...
```

```
- ansible.builtin.template:  
  src: my.conf.j2  
  dest: /path/to/my.conf
```

```
...
```

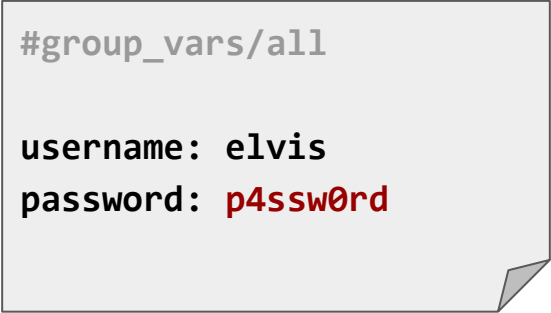
```
#/path/to/my.conf
```

```
x.user = elvis  
x.pass = p4ssw0rd
```

Problems

3.

We cannot store plain text secrets in Git!



```
#group_vars/all  
  
username: elvis  
password: p4ssw0rd
```

Ansible Vault

Command line tool to encrypt and decrypt sensitive data

Can encrypt files or separate variables

More: https://docs.ansible.com/ansible/latest/user_guide/vault.html

Ansible Vault: Vault password

Password that is used to encrypt and decrypt all other secrets

Stored in a separate file **outside** of Ansible repository, for example,

`~/.ansible/vault_password`

Configure Ansible to read the Vault password from file
(add this to `ansible.cfg`):

```
[defaults]
inventory = hosts
vault_password_file = ~/.ansible/vault_password
```

Ansible Vault: encryption

```
$ ansible-vault encrypt_string p4ssw0rd
```

```
!vault |
```

```
$ANSIBLE_VAULT;1.1;AES256
```

```
66313066386566346263353134373763333466303863656133343232623162303562643366383562  
6236373364653534376164633063663937386339333765320a316538386363663733383939626333  
66343661343564353934373239316538346666656436303035623635363661636631393261353966  
3166613234313639640a613431306235643333373164643666666437356330306166306431366662  
6161
```

```
Encryption successful
```

Ansible Vault: decryption

```
#group_vars/all.yaml
```

```
my_user: admin
```

```
my_password: !vault |
```

```
    $ANSIBLE_VAULT;1.1;AES256
```

```
    66313066386566346263353134373763333466303863656133343232623162303562643366383562
    6236373364653534376164633063663937386339333765320a316538386363663733383939626333
    66343661343564353934373239316538346666656436303035623635363661636631393261353966
    3166613234313639640a613431306235643333373164643666666437356330306166306431366662
    6161
```

```
#roles/demo/tasks/main.yaml
```

```
- name: Print credentials (Unsafe! Use for demo only!)
```

```
  ansible.builtin.debug:
```

```
    msg: "User is {{ my_user }} and password is {{ my_password }}"
```

Questions?