Functional programming design patterns in Ansible code

Kirill Satarin
Principal Software Engineer, Red Hat*
Config Management Camp, Ghent, Belgium, 3-4-5 February 2025
*presenting personal opinion

Who this talk is for?

You know what Ansible is

You use Ansible

You create Ansible collections and roles

You would like to improve your Ansible content creation and testing

What this talk is about

What is functional programming

Why Ansible is functional programming

How to use functional programming to your advantage

What is functional programming?

- Functions
  - Pure functions
  - Functions with side effects
- Immutable variables
- Declarative
- Lazy evaluations

Functions and Side effects

Functions have inputs and outputs, everything but output is a side effect.

Pure functions are functions without any side effects

Printing is a side effect, saving information on disk is a side effect, changing host configuration is a side effect

(Pure) Functions in Ansible

Filters - https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_filters.html

- Default
- Mandatory
- Ternary
- Dict2items
- Data conversions: bool, int, str
- List

Jinja2 filters
https://jinja.palletsprojects.com/en/stable/templates/

- Map
- Select / reject
- Selectattr / rejectattr

Variables = (Pure) Functions in Ansible?

% cat vars/main.yml

```
function1: "{{ argument }}"
function2: "{{ argument | lower }}"
function3: "{{ argument | upper }}"
function4: "{{ argument | reverse }}"
function5: "{{ argument | sort }}"
```

% cat tasks/main.yml

```
- name: Use different functions
  ansible.builtin.debug:
    msg:
      - "{{ function1 }}"
      - "{{ function2 }}"
      - "{{ function3 }}"
      - "{{ function4 }}"
      - "{{ function5 }}"
    vars:
      argument: “Hello, CfgMgmtCamp”
```

Functions with side effects in Ansible

Any action is a function with side effects:

```
- name: Print variable
  ansible.builtin.debug:
    var: hello
  register: hello_debug_output
  vars:
    hello: "Hello CfgMgmtCamp 2025!"
```

```
TASK [functional.ansible.side_effect : Print variable] **********
ok: [localhost] => {
    "hello": "Hello CfgMgmtCamp 2025"
}

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

Immutable variables

Imperative

a = 5

…

print(a)

Functional

let a = 5

…

print a

What will it print?

Immutable variables are good because they simplify reasoning about the program

Immutable variables in Ansible

```
- name: Print 'hello' variable
  ansible.builtin.debug:
    var: hello
  register: hello
  vars:
    hello: "Hello CfgMgmtCamp 2025!"
```

https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_variables.html

Read <<<<<<<<<<

1. command line values (for example, `-u my_user`, these are not variables)
2. role defaults (as defined in Role directory structure) [1]
3. inventory file or script group vars [2]
4. inventory group_vars/all [3]
5. playbook group_vars/all [3]
6. inventory group_vars/* [3]
7. playbook group_vars/* [3]
8. inventory file or script host vars [2]
9. inventory host_vars/* [3]
10. playbook host_vars/* [3]
11. host facts / cached set_facts [4]
12. play vars
13. play vars_prompt
14. play vars_files
15. role vars (as defined in Role directory structure)
16. block vars (only for tasks in block)
17. task vars (only for the task)
18. include_vars
19. set_facts / registered vars <<< Change
20. role (and include_role) params
21. include params
22. extra vars (for example, `-e "user=my_user"`)(always win precedence)

Declarative programming

Imperative programming

```
for x in range(1,n):
    If x % 2 ==0:
        print(x)
```

Functional programming

```
map(print, filter(lambda x: x % 2 == 0, range(1, n)))
```

Declarative programming in Ansible

Ansible is declarative!

~~If not is_installed(package_name):~~

~~install (package_name)~~

```
- name: Install package
  ansible.builtin.package:
    name: “{{ package_name }}”
    state: present
```

Lazy evaluations in Ansible

Being lazy - do not calculate value unless you definitely need it

All the variables are lazy evaluated in Ansible
cat vars/main.yml

- - -

lazy_not_accessed: this variable is never accessed that is why it can have any value

lazy_not_accessed_bool: "{{ lazy_not_accessed | bool }}"

lazy_not_accessed_but_mandatory: "{{ does_not_exist | ansible.builtin.mandatory }}"

How is this beneficial to Ansible content creators

Functions

- Pure functions - try to make all your functions pure
- Functions with side effects - control all the Ansible actions

Immutable variables - “never” use set_facts

Declarative - you do not have to understand how it is done, only the result

Lazy evaluations - use *vars/main.yml* and *defaults/main.yml* in your roles

How to use it?

DEMO time