Ansible Filters

a short introduction

Frank Jung

frankhjung@linux.com

August 9, 2018




Topics

What we will cover . . .

Topics

What we will cover . . .

- what are Ansible filters?




Topics

What we will cover . . .

- what are Ansible filters?
- some typical use cases




Topics

What we will cover . . .

- what are Ansible filters?
- some typical use cases
- getting help




Topics

What we will cover . . .

- what are Ansible filters?
- some typical use cases
- getting help
- how to roll your own?



What are Ansible Filters?

Filters are from a fast, flexible, Python template engine called . . .




What are Ansible Filters?

Filters are from a fast, flexible, Python template engine called . . .

overview ~ documentation ~ community

Welcome

Jinja2 is a full featured template engine for Python. It has full unicode support, an optional integrated sandboxed execution environment, widely used and BSD licensed.

Jinja is Beautiful

```
{% extends "layout.html" %}
{% block body %}
  <ul>
  {% for user in users %}
    <li><a href="{{ user.url }}">{{ user.username }}</a></li>
  {% endfor %}
  </ul>
{% endblock %}
```

And Powerful

Jinja2 is one of the most used template engines for Python. It is inspired by Django's templating system but extends it with an expressive language that gives template authors a more powerful set of tools. On top of that it adds sandboxed

jinja.pocoo.org



What are Ansible Filters?

Filters are used to transform data . . .




What are Ansible Filters?

Filters are used to transform data . . .

1. they can be used inside a template expression




What are Ansible Filters?

Filters are used to transform data . . .

1. they can be used inside a template expression
2. they can be used to manipulate local data




What are Ansible Filters?

Filters are used to transform data . . .

1. they can be used inside a template expression
2. they can be used to manipulate local data




What are Ansible Filters?

Some important features of filters . . .




What are Ansible Filters?

Some important features of filters . . .

1. they are executed on the Ansible controller

What are Ansible Filters?

Some important features of filters . . .

1. they are executed on the Ansible controller
   - **not** on the task's target host

What are Ansible Filters?

Some important features of filters . . .

1. they are executed on the Ansible controller
   - **not** on the task's target host
2. Ansible ships with its own filters




What are Ansible Filters?

Some important features of filters . . .

1. they are executed on the Ansible controller
   - **not** on the task's target host
2. Ansible ships with its own filters
   - or use standard filters from Jinja2




What are Ansible Filters?

Some important features of filters . . .

1. they are executed on the Ansible controller
   - **not** on the task's target host
2. Ansible ships with its own filters
   - or use standard filters from Jinja2
   - or you can write your own!




What are Ansible Filters?

Some important features of filters . . .

1. they are executed on the Ansible controller
   - **not** on the task's target host
2. Ansible ships with its own filters
   - or use standard filters from Jinja2
   - or you can write your own!
3. can be sandboxed




What are Ansible Filters?

Some important features of filters . . .

1. they are executed on the Ansible controller
   - **not** on the task's target host
2. Ansible ships with its own filters
   - or use standard filters from Jinja2
   - or you can write your own!
3. can be sandboxed
   - so can be used to evaluate untrusted code




Typical Filter Use Cases

For the following pages I will use this example structure . . .

```
vars:
  martin:
    name: Martin D'veloper
    job: Developer
    skills:
      - ansible
      - python
      - perl
      - ruby
    url: https://github.com/frankhjung/ansible-filters
```




Typical Filter Use Cases

Use filters to manipulate local data . . .




Typical Filter Use Cases

Use filters to manipulate local data . . .

- create new facts

```
- name: count the items in martin.skills
  set_fact:
    skills_length: "{{ martin.skills | length }}"
```




Typical Filter Use Cases

Use filters to manipulate local data . . .

- create new facts
- subset or filter lists

```
- name: show skills starting with 'p'
  debug:
    var: item
  with_items: "{{ martin.skills }}"
  when: item | match('p.*')

- name: show skills except 'perl'
  debug:
    var: item
  with_items: "{{ martin.skills }}"
  when: not item | match('perl')
```




Typical Filter Use Cases

Use filters to manipulate local data . . .

- create new facts
- subset or filter lists
- manipulate strings

```
- name: sort skills
  debug:
    var: item
  with_items: "{{ martin.skills | sort(reverse=True) }}"

- name: change job to lowercase
  debug:
    msg: "{{ martin.job | lower }}"

- name: capitalize skills
  debug:
    msg: "{{ item | capitalize }}"
  with_items: "{{ martin.skills }}"
```




Pipes

Filters are piped . . .




Pipes

Filters are piped . . .

```
- debug:
    msg: "{{ martin.name | lower }}"
```




Pipes

Filters are piped . . .

```
- debug:
    msg: "{{ martin.name | lower }}"
```

Which results in . . .

```
TASK [debug] ********************************************************************
ok: [localhost] => {
    "msg": "martin d'veloper"
}
```



Pipes

Pipes can be chained . . .




Pipes

Pipes can be chained . . .

```
- name: show name
  debug:
    var: martin.name

- debug:
    msg: "{{ martin.name | list | length }}"
```



Pipes

Pipes can be chained . . .

```
- name: show name
  debug:
    var: martin.name

- debug:
    msg: "{{ martin.name | list | length }}"
```

Which results in . . .

```
TASK [show name] ***************************************************************
ok: [localhost] => {
    "martin.name": "Martin D'veloper"
}

TASK [debug] *******************************************************************
ok: [localhost] => {
    "msg": "16"
}
```

Getting help when things go wrong

New in Ansible 2.3 is `type_debug`:

Getting help when things go wrong

New in Ansible 2.3 is `type_debug`:

```
- name: type_debug var
  debug:
    msg: "{{ martin | type_debug }}"
```




Getting help when things go wrong

New in Ansible 2.3 is `type_debug`:

```
- name: type_debug var
  debug:
    msg: "{{ martin | type_debug }}"
```

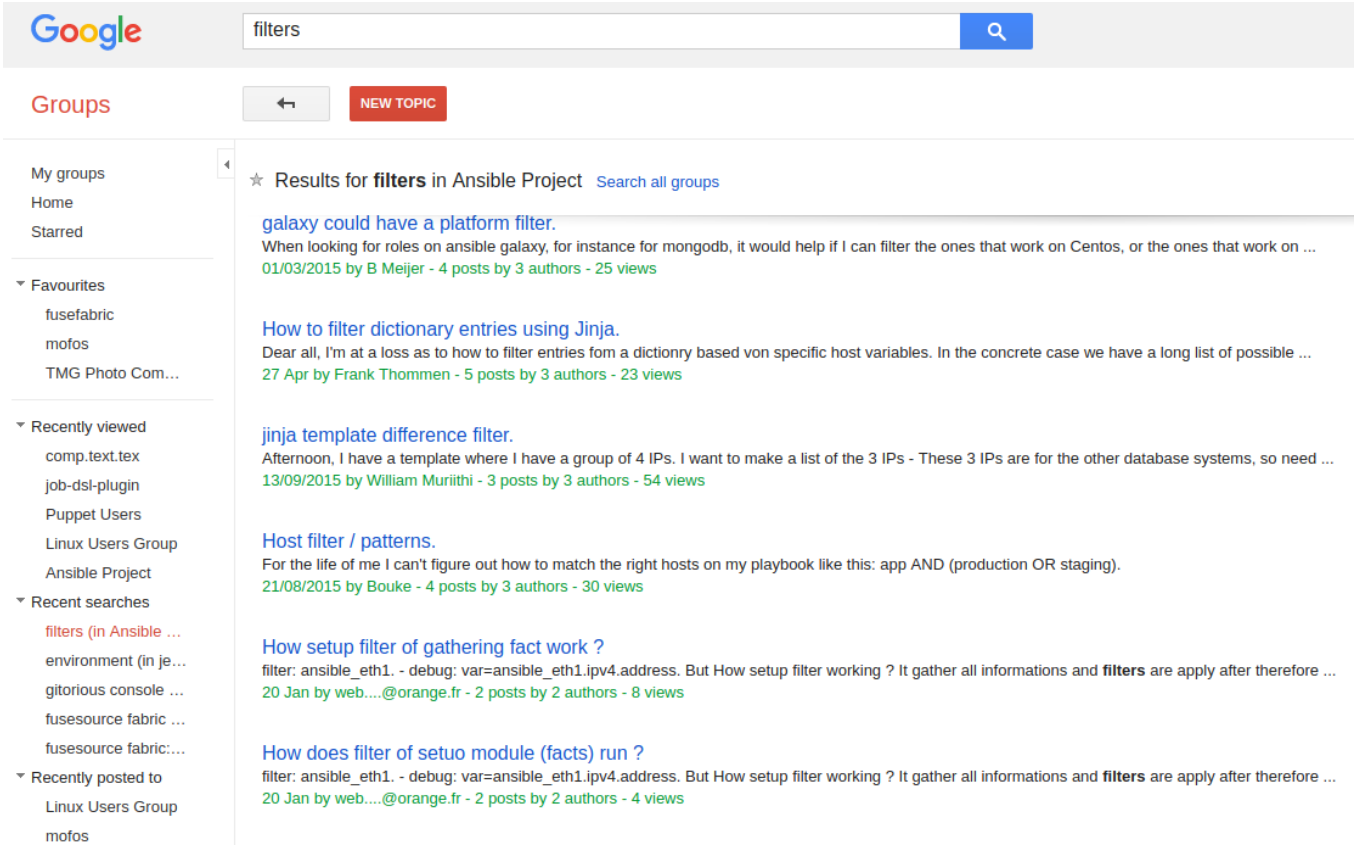
Which gives variable **type** information . . .

```
TASK [type_debug var] ***************************
ok: [localhost] => {
    "msg": "dict"
}
```




Getting help when things go wrong

Ansible Project - Google Groups



https://groups.google.com/d/forum/ansible-project

Getting help when things go wrong

Read the source, for example Jinja has a sort method:

```
def do_sort(environment, value, reverse=False,
            case_sensitive=False, attribute=None):
    """Sort an iterable.  Per default it sorts ascending,
    if you pass it true as first argument it will reverse
    the sorting.
    ...
```

Getting help when things go wrong

Read the source, for example Jinja has a sort method:

```
def do_sort(environment, value, reverse=False,
            case_sensitive=False, attribute=None):
    """Sort an iterable.  Per default it sorts ascending,
    if you pass it true as first argument it will reverse
    the sorting.
    ...
```

Where `do_sort` maps to ...




Getting help when things go wrong

Read the source, for example Jinja has a sort method:

```
def do_sort(environment, value, reverse=False,
            case_sensitive=False, attribute=None):
    """Sort an iterable.  Per default it sorts ascending,
    if you pass it true as first argument it will reverse
    the sorting.
    ...
```

Where `do_sort` maps to . . . `sort`:

```
FILTERS = {
    'sort':                 do_sort,
      ...
}
```



Getting help when things go wrong

Read the source, for example Jinja has a sort method:

```
def do_sort(environment, value, reverse=False,
            case_sensitive=False, attribute=None):
    """Sort an iterable.  Per default it sorts ascending,
    if you pass it true as first argument it will reverse
    the sorting.
    ...
```

Where `do_sort` maps to . . . `sort`:

```
FILTERS = {
    'sort':                 do_sort,
      ...
}
```

Which gives a hint on how to use this filter:

```
- name: sort skills
  debug:
    var: item
  with_items: "{{ martin.skills | sort(reverse=True) }}"
```

https://github.com/pallets/jinja/blob/master/jinja2/filters.py



Writing your own filter

You may think to start with the Ansible documentation . . .

- Ansible dev guide to developing plugins




Writing your own filter

You may think to start with the Ansible documentation . . .

- Ansible dev guide to developing plugins

But, that only redirects you to . . .

- lib/ansible/plugins/filter




Writing your own filter

You may think to start with the Ansible documentation . . .

- Ansible dev guide to developing plugins

But, that only redirects you to . . .

- lib/ansible/plugins/filter

A better place to start is . . .

- Creating your own Ansible filter plugins by *Jon Langemak*
  `http://www.dasblinkenlichten.com/creating-ansible-filter-plugins/`



Writing your own filter

You may think to start with the Ansible documentation . . .

- Ansible dev guide to developing plugins

But, that only redirects you to . . .

- lib/ansible/plugins/filter

A better place to start is . . .

- Creating your own Ansible filter plugins by *Jon Langemak*
  `http://www.dasblinkenlichten.com/creating-ansible-filter-plugins/`

Let's go through an example in detail . . .



Writing your own filter

The steps to create a custom filter are . . .




Writing your own filter

The steps to create a custom filter are . . .

- write a test for your plugin




Writing your own filter

The steps to create a custom filter are . . .

- write a test for your plugin
- create a directory called `filter_plugins`




Writing your own filter

The steps to create a custom filter are . . .

- write a test for your plugin
- create a directory called `filter_plugins`
- add a Python script into this directory



Writing your own filter

The steps to create a custom filter are . . .

- write a test for your plugin
- create a directory called `filter_plugins`
- add a Python script into this directory
  - add a function implementing your filter




Writing your own filter

The steps to create a custom filter are . . .

- write a test for your plugin
- create a directory called `filter_plugins`
- add a Python script into this directory
  - add a function implementing your filter
  - implement class `FilterModule` overriding `filters` method

Writing your own filter

The steps to create a custom filter are . . .

- write a test for your plugin
- create a directory called `filter_plugins`
- add a Python script into this directory
  - add a function implementing your filter
  - implement class `FilterModule` overriding `filters` method
- test your new filter

Now the code . . .




Writing your own filter

Writing a custom filter in detail . . .




Writing your own filter

Writing a custom filter in detail . . .

- write a test for your plugin `filter_examples.yaml`

```
- name: count number of occurrences of 'r' in name
  set_fact:
    counter: "{{ martin.name | count('r') }}"

- name: test there are exactly 2 occurrences
  assert:
    that: counter | int == 2
    msg: "expect counter to be 2 got {{ counter }}"
```




Writing your own filter

Writing a custom filter in detail . . .

- write a test for your plugin `filter_examples.yaml`
- create a directory called `filter_plugins`

  ```
  mkdir filter_plugins
  ```




Writing your own filter

Writing a custom filter in detail . . .

- write a test for your plugin `filter_examples.yaml`
- create a directory called `filter_plugins`
- add a Python script into this directory

```
edit filter_plugins/custom.py
```




Writing your own filter

Writing a custom filter in detail . . .

- write a test for your plugin `filter_examples.yaml`
- create a directory called `filter_plugins`
- add a Python script into this directory
  - add a function implementing your filter

```
def count(word, char):
    ''' Count occurrences of character in string. '''
    return word.count(char)
```




Writing your own filter

Writing a custom filter in detail . . .

- write a test for your plugin `filter_examples.yaml`
- create a directory called `filter_plugins`
- add a Python script into this directory
  - add a function implementing your filter
  - implement class `FilterModule` overriding `filters` method

```
class FilterModule(object):
    ''' Custom Jinja2 filters. '''
    def filters(self):
        return {'count': count}
```




Writing your own filter

Writing a custom filter in detail . . .

- write a test for your plugin `filter_examples.yaml`
- create a directory called `filter_plugins`
- add a Python script into this directory
  - add a function implementing your filter
  - implement class `FilterModule` overriding `filters` method
- test your new filter

```
TASK [debug] *******************************************************************
ok: [localhost] => {
    "martin.name": "Martin D'veloper"
}

TASK [count number of occurrences of 'r' in name] ******************************
ok: [localhost]

TASK [test there are exactly 2 occurrences] ************************************
ok: [localhost] => {
    "changed": false,
    "msg": "All assertions passed"
}
```



Writing your own filter

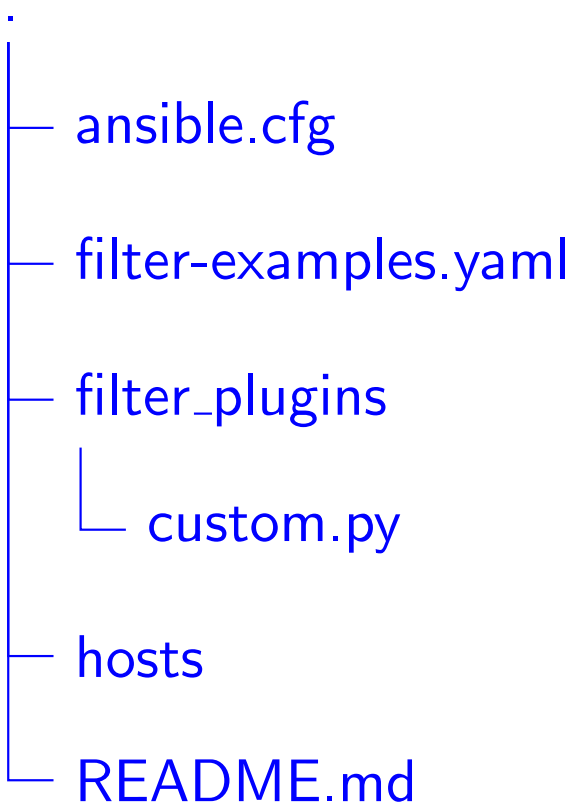
The full Python code for custom filter:

```
#!/usr/bin/python

def count(word, char):
    ''' Count occurrences of character in string. '''
    return word.count(char)

class FilterModule(object):
    ''' Custom Jinja2 filters. '''
    def filters(self):
        return {'count': count}
```




Writing your own filter

Ansible project structure with filters . . .



What we covered . . .




What we covered . . .

- what filters are




What we covered . . .

- what filters are
- typical places you will use them




What we covered . . .

- what filters are
- typical places you will use them
- getting help when things go wrong




What we covered . . .

- what filters are
- typical places you will use them
- getting help when things go wrong
- writing your own filter



List of Ansible Filters

b64decode
b64encode
basename
bool
change
changed
checksum
combine
comment
dirname
expanduser
extract
failed
failure
fileglob
from_json
from_yaml
groupby
hash
mandatory
md5
password_hash
quote
random
realpath
regex_escape
regex_findall
regex_replace
regex_search
relpath
sha1
shuffle
skip
skipped
splitext
strftime
succeeded
success
ternary
to_datetime
to_json
to_nice_json
to_nice_yaml
to_uuid
to_yaml
type_debug
win_basename
win_dirname
win_splitdrive

List of Jinja Filters

abs
attr
batch
capitalize
center
default
dictsort
escape
filesizeformat
first
float
forceescape
format
groupby
indent
int
join
last
length
list
lower
map
pprint
random
reject
rejectattr
replace
reverse
round
safe
select
selectattr
slice
sort
string
striptags
sum
title
trim
truncate
upper
urlencode
urlize
wordcount
wordwrap
xmlattr

Using Python String methods

If the Ansible variable is of type string, then you can use Python's string methods. For example:

```
- name: split name using python
  debug:
    msg: "{{ martin.name.split(' ') }}"
```



Using Python String methods

If the Ansible variable is of type string, then you can use Python's string methods. For example:

```
- name: split name using python
  debug:
    msg: "{{ martin.name.split(' ') }}"
```

Which gives you back a list:

```
TASK [debug] *******************************************************************
ok: [localhost] => {
    "martin.name": "Martin D'veloper"
}

TASK [split name using python] *************************************************
ok: [localhost] => {
    "msg": [
        "Martin",
        "D'veloper"
    ]
}
```




Resources

Presentation and sample code

- https://github.com/frankhjung/ansible-filters

www.ansible.com

- http://docs.ansible.com/ansible/playbooks_filters.html
- https://github.com/ansible/ansible

Ansible Project - Google Groups

- https://groups.google.com/d/forum/ansible-project

Creating your own Ansible filter plugins

- http://www.dasblinkenlichten.com/creating-ansible-filter-plugins/

jinja.pocoo.org

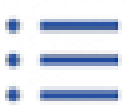
- http://jinja.pocoo.org/
- http://jinja.pocoo.org/docs/2.9/templates
- https://github.com/pallets/jinja

Addendum

Ansible

Association for Computing Machinery

The Linux Foundation



The Marlo Group



Frank Jung
frankhjung@linux.com
github.com/frankhjung/ansible-filters