Contents

[How to RUN ansible module in Python 1](#_Toc505591410)

[Iterating Through STDOUT 2](#_Toc505591411)

[JSON parsing 3](#_Toc505591412)

[Ansible-string-split-filter 4](#_Toc505591413)

[Setup 4](#_Toc505591414)

[Usage 5](#_Toc505591415)

[Examples 5](#_Toc505591416)

[Ansible Logging 6](#_Toc505591417)

[Ansible return correct result via exit\_json 6](#_Toc505591418)

Judging from [this example](http://docs.ansible.com/ansible/dev_guide/developing_modules_general.html#building-a-simple-module), an Ansible module is expected to send its output to stdout.

So this should work:

import sys

... your existing code here ...

result.json().dumps(sys.stdout)

Or, if you don't need to alter the JSON in any way, you can print result.content directly.

# How to RUN ansible module in Python

Here is my code: I am not able to run a ansible module using python. How to pass a inventory file for which this command is running. I am not able to run it for my inventory. Do I need to do something else ? Here is my ansible command:

ansible all -i /home/ubuntu/extra -m 'debug' -a 'var=hostvars'

Here is my code:

import json

import ansible.runner

import ansible.playbook

import ansible.inventory

example\_inventory = ansible.inventory.Inventory('path/to/your/inventory')

pm = ansible.runner.Runner( module\_name = 'debug', module\_args = 'var=hostvars', timeout = 5, inventory = example\_inventory, subset = 'all')

out = pm.run()

print json.dumps(out, sort\_keys=True, indent=4, separators=(',', ': '))

|  |  |
| --- | --- |
| You can pass the inventory file path to ansible.runner.Runner()  And for getting group-names and host-names , you should pass var=hostvars, not vars=hostvars  Your code would look like this,  import json  import ansible.runner  import ansible.playbook  import ansible.inventory  example\_inventory = ansible.inventory.Inventory('path/to/your/inventory')  pm = ansible.runner.Runner( module\_name = 'debug', module\_args = 'var=hostvars', timeout = 5, inventory = example\_inventory, subset = 'all')  out = pm.run()  print json.dumps(out, sort\_keys=True, indent=4, separators=(',', ': '))  and your output  {'contacted': {'ip-address': {'invocation': {'module\_args': u'var=hostvars',  'module\_complex\_args': {},  'module\_name': 'debug'},  'var': {u'hostvars': {'group\_names': ['group1', 'group2', 'group3'],  'groups': {'group1': ['ip-address'],  'all': ['ip-address'],  'group2': ['ip-address'],  'group3': ['ip-address'],  'ungrouped': []},  'inventory\_hostname': 'ip/hostname',  'inventory\_hostname\_short': 'hostname-short'}},  'verbose\_always': True}},  'dark': {}}   |  | | --- | | [share](https://stackoverflow.com/a/33543150/7579266)[edit](https://stackoverflow.com/posts/33543150/edit)[flag](https://stackoverflow.com/questions/33538483/how-to-run-ansible-module-in-python) | |

# Iterating Through STDOUT

If you need to iterate over every line from all commands, use:

- debug:

msg: "Do smth for line {{ item }}"

with\_items: "{{ found | json\_query('results[].stdout\_lines[]') }}"

This will take ever element from found.results, then every element from every stdout\_lines

I am writing a playbook to locate a string pattern in a sequence of files. If I run my utility through the command module it will generate one or more strings on STDOUT. To run this across a number of systems I would like to run the command with\_items:

- command: "findstring {{ item }}"

with\_items:

- "string1"

- "string2"

register: found

failed\_when: found.rc >= 2

And then iterate over the result to post process the info:

- name: Print strings we found

debug:

var: "{{ item }}"

with\_items: found.results

--------------------------------------------------------------------------------------------------------------------------------------

I have a playbook that looks like this:

- hosts: host1

gather\_facts: false

tasks:

- name: "Loop"

command: "echo {{ item }}"

with\_items: [ 0, 2, 4, 6, 8, 10 ]

register: hello

- debug: "msg={{ hello.results }}"

Everything works correctly, and the output is returned, but there is tons and tons of output. It turns out that this:

- debug: "msg={{ hello.results.1.stdout }}"

does exactly what I want -- just grab the stdout from the command -- but only for one of the six times through the loop.

What I really want/need to do is this:

- debug: "msg={{ hello.results.\*.stdout }}"

where it goes into the hello structure, accesses the results entry, goes to each member of that array, and pulls out the stdout value.

**Solution**:

- debug: "msg={{ hello.results | map(attribute='stdout') | join('\n') }}"

**Remark**:

By default, Ansible will print visible \n two-character sequences instead of wrapping the lines, so either use a callback plugin for a human readable output ([example](https://github.com/n0ts/ansible-human_log)) or verify the method with:

- copy:

content: "{{ hello.results | map(attribute='stdout') | join('\n') }}"

dest: ./result.txt

Support for dynamic nested loops is not implemented in Ansible.

To iterate over each line, you can flatten the result:

- name: print message

debug:

msg: "{{ item + ' test' }}"

with\_items: "{{ ssh\_known\_host\_results.results | map(attribute='stdout\_lines') | list }}"

# JSON parsing

{

"hosts" : [

"xxx:<port>",

"xxx:<port>",

"xxx:<port>"

],

"setName" : "xxx",

"setVersion" : xxx,

"ismaster" : true,

"secondary" : false,

"primary" : "xxx",

"me" : "xxx",

"electionId" : ObjectId("xxxx"),

"maxBsonObjectSize" : xxx,

"maxMessageSizeBytes" : xxxx,

"maxWriteBatchSize" : xxx,

"localTime" : ISODate("xxx"),

"maxWireVersion" : 4,

"minWireVersion" : 0,

"ok" : 1

}

There are quite a bit of helpful [filters](http://docs.ansible.com/ansible/playbooks_filters.html#filters-for-formatting-data) in Ansible.

Try: when: (output\_text.stdout | from\_json).ismaster

{

"queue": {

"first": {

"car": "bmw",

"year": "1990",

"model": "x3",

"color": "blue"

},

"second": {

"car": "bmw",

"year": "2000",

"model": "318",

"color": "red"

}

}

}

tasks: print the color

set\_fact:

color1 : "{{ jsonVar | from\_json | json\_query('queue.[0].['color']')}}"

color2 : "{{ jsonVar | from\_json | json\_query('queue.[1].['color']')}}"

# Ansible-string-split-filter

A simple Ansible Jinja2 filter to split a string into a list.

## Setup

See the [plugin documentation](http://docs.ansible.com/developing_plugins.html#distributing-plugins).

## Usage

split(delimiter)

split\_regex(pattern) # regex syntax is identical to Pythons

## Examples

In a Jinja 2 template:

# hostname = dev.example.com

ldap\_server = "dc={{ hostname | split('.') | join(',dc=') }}"

Output:

ldap\_server = dc=dev,dc=example,dc=com

The same example, using split\_regex:

ldap\_server = "dc={{ hostname | split\_regex('\.') | join(',dc=') }}"

---

- hosts: localhost

tasks:

- debug:

msg: "Item: {{item}}"

with\_items: "{{ '1,2,3,4,5'.split(',') }}"

Output from ansible 2.1:

PLAY [localhost] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

ok: [localhost]

TASK [debug] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [localhost] => (item=1) => {

"item": "1",

"msg": "Item: 1"

}

ok: [localhost] => (item=2) => {

"item": "2",

"msg": "Item: 2"

}

ok: [localhost] => (item=3) => {

"item": "3",

"msg": "Item: 3"

}

ok: [localhost] => (item=4) => {

"item": "4",

"msg": "Item: 4"

}

ok: [localhost] => (item=5) => {

"item": "5",

"msg": "Item: 5"

}

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

localhost : ok=2 changed=0 unreachable=0 failed=0

# Ansible Logging

Modules normally are executed remotely, so there actually is no way to directly output anything.

You can return additional data in your module in the exit\_json call.

if module.\_verbosity >= 3:

module.exit\_json(changed=True, debug="wooha!")

else:

module.exit\_json(changed=True)

module.\_verbosity corresponds to the verbosity level (-v = 1, -vvv = 3) and is available since Ansible 2.1.

Source: [Ansible Devel-list](https://groups.google.com/d/msg/ansible-devel/s0iSb7phnqY/UB9vaLFJAwAJ) and [github](https://github.com/ansible/ansible/commit/5760f0effbebe55e7150eb06e9a6b691a0e98455" \l "diff-90085fdcec6ed8b273ba885eaee60328)

# Ansible return correct result via exit\_json

import os

def main():

module = AnsibleModule(

argument\_spec = dict()

)

(rc, uname\_os, stderr) = module.run\_command("uname -r")

(rc, rpm\_os, stderr) = module.run\_command("rpm -q --last kernel | perl -pe 's/^kernel-(\S+).\*/$1/' | head -1 | sed -e 's/^[ \t]\*//' | sed 's/ //g'")

if rpm\_os.rstrip() != uname\_os.rstrip():

out = "REBOOT"

changed = True ## now changed is changed to True

else:

out = "DO NOT REBOOT"

changed = False

module.exit\_json(changed=changed, output=out)

from ansible.module\_utils.basic import \*

main()

changed = False is always the last statement executed whether rpm\_os != (or ==) uname\_os --> as is "out =" and "module.exit" which you possibly don't want here either. I assume you return "changed" and not some other variable from the function. Try:

changed = False ## default

out = "DO NOT REBOOT"

if rpm\_os != uname\_os:

out = "REBOOT"

changed = True ## now changed is changed to True

module.exit\_json(changed=changed, ouput=out)