

ANSIBLE

# Writing Modules For Ansible

**Sergey Goncharov**

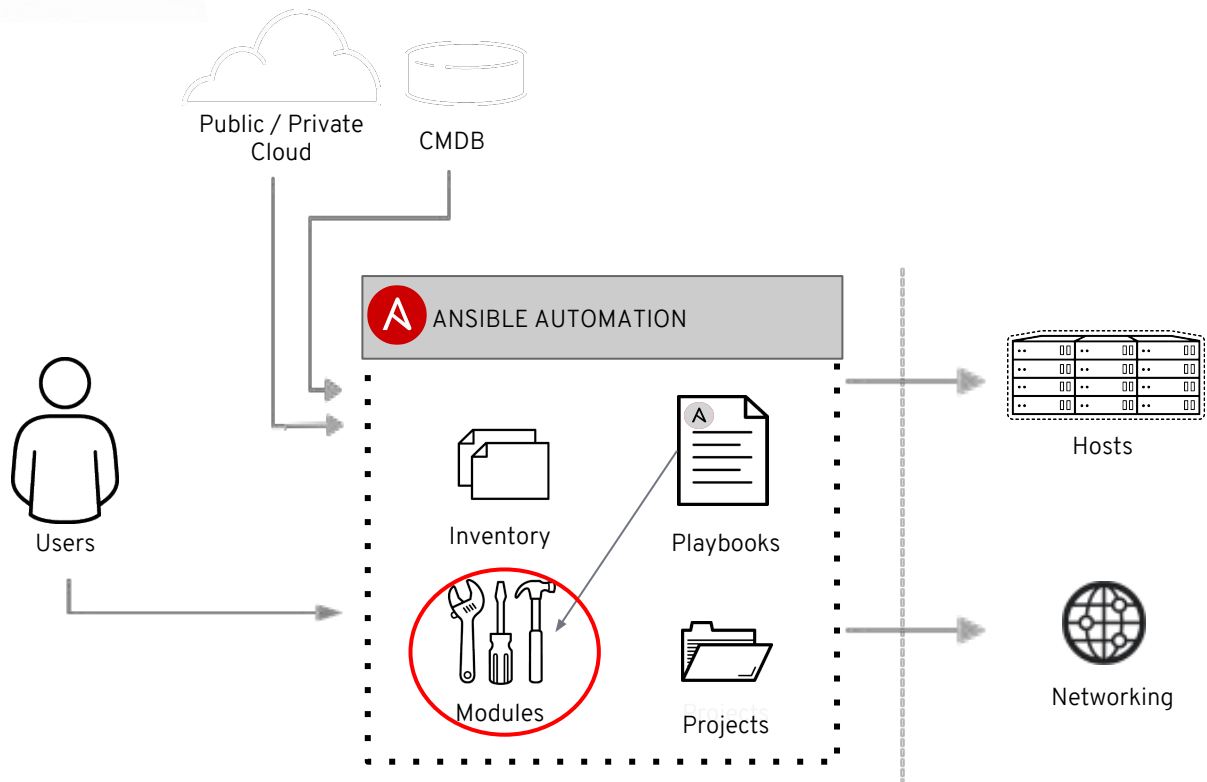
Senior Solution Architect

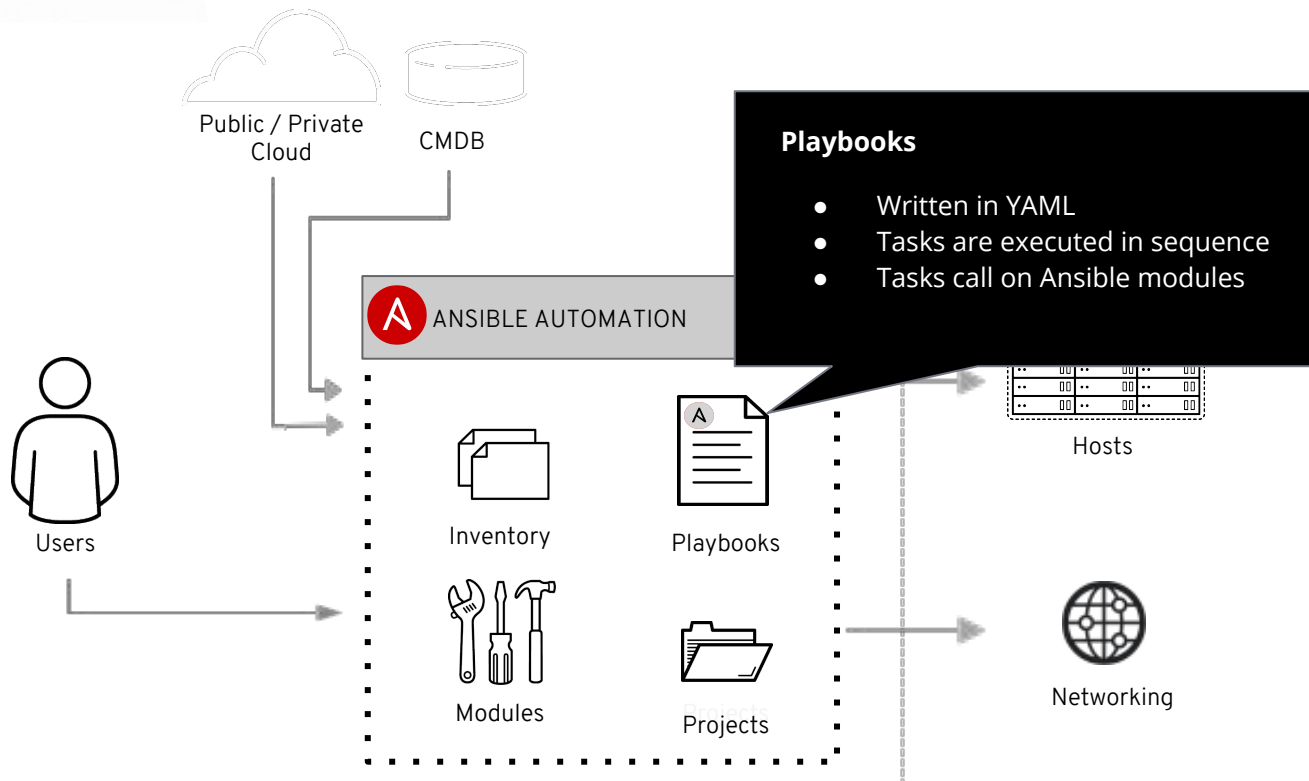
Cloud Infrastructure

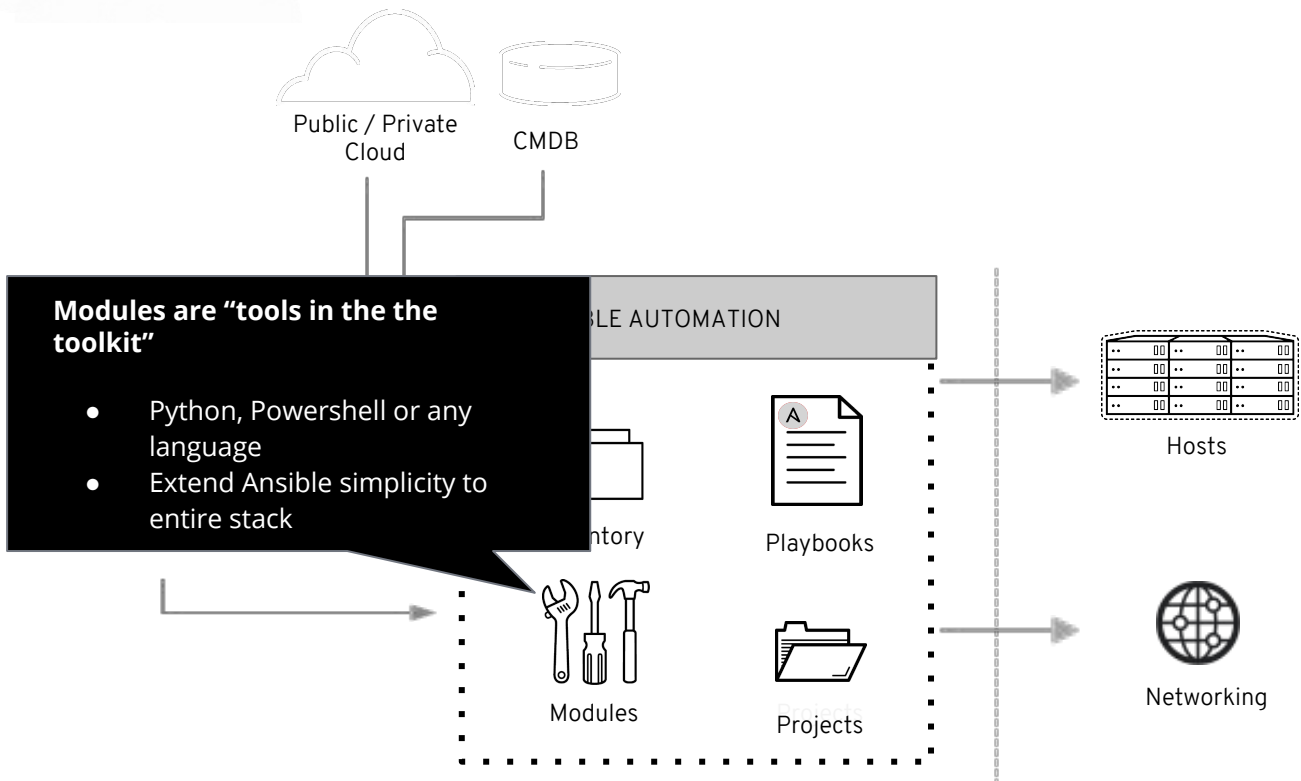
**Red Hat**

[sgonchar@redhat.com](mailto:sgonchar@redhat.com)

# What Are Modules?







# What Are Modules?

## Using shipped modules

Ansible 2.2 contained ~900 modules  
Ansible 2.5 contains ~1650 modules

## Ansible 2.5

For previous versions, see the [documentation archive](#).

### INSTALLATION, UPGRADE & CONFIGURATION

- ▣ Installation Guide
- ▣ Configuring Ansible
- ▣ Ansible Porting Guides

### USING ANSIBLE

#### ▣ **User Guide**

- Ansible Quickstart
- ▣ Getting Started
- ▣ Working with Command Line Tools
- ▣ Introduction To Ad-Hoc Commands
- ▣ Working with Inventory
- ▣ Working With Dynamic Inventory

## Module Index

- [All modules](#)
- [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](#)

# Should you develop a module?

Before starting development, answer these questions

- **Does a** similar module exist?
  - [http://docs.ansible.com/ansible/latest/modules/modules\\_by\\_category.html](http://docs.ansible.com/ansible/latest/modules/modules_by_category.html)
- **Is there** development already ongoing for a similar module?
  - [https://github.com/ansible/ansible/labels/new\\_module](https://github.com/ansible/ansible/labels/new_module)
  - <https://github.com/ansible/ansible/labels/module>
  - <https://ansible.sivel.net/pr/byfile.html>
- **Should you** use or develop an action plugin instead?
- **Should you** use a role instead? (14 000 exists on Galaxy)
- **Should you** write one or multiple modules?
  - Remember that complexity kills productivity



## Philips Hue Lightbulbs - Why Write a Module?

### Complexity:

- You must register your app with the base station.
- The API is REST based and expects JSON for updates.
- The API returns JSON, which is complex to parse and validate in a playbook.
- Some actions require multiple API calls, which translate into individual tasks in a playbook (even when looping).
- RGB colors are not used by the API - you must use Hue/Saturation/Luminance or one of two other options.



# How Modules Work

There are different types of modules, for example:

- **Action plugins** - always execute server-side and are sometimes able to do all work there (example: *debug*, *template*)
- **New-style modules** - all that ship with Ansible. Arguments embedded in module instead of separate file, more efficient.
- **Python** - New-style Python modules use the Ansiballz framework for constructing modules. These modules use imports from `ansible.module_utils` in order to pull in boilerplate module code, such as argument parsing, formatting of return values as JSON and various file operations.
- **Powershell** - use the Module Replacer framework for constructing modules.

There are different types of modules, for example:

- **JSONARGS** - Scripts that arrange for an argument string to be placed within them using special string:  
jsonargs = "<<INCLUDE\_ANSIBLE\_MODULE\_JSON\_ARGS>>"
- **Non-native WANT\_JSON modules** - If a module has the string WANT\_JSON in it anywhere, Ansible treats it as a non-native module that accepts a filename as its only command line parameter, the format of the argument file will then be in JSON. Otherwise it will be key=value.
- **Binary module** - compiled small program, works like a WANT\_JSON module.
- **And more**  
[http://docs.ansible.com/ansible/latest/dev\\_guide/developing\\_program\\_flow\\_modules.html](http://docs.ansible.com/ansible/latest/dev_guide/developing_program_flow_modules.html)

## Modules written in other languages than Python/Powershell

- Modules can be written in any language an author wishes, they just need to specify certain special strings in their code.
- If not, a file containing the module args will be uploaded, and the path to that file is the first argument to the module. If your module contains the string `WANT_JSON`, that args file will be formatted as JSON (otherwise they're **key=value** pairs).
- As of Ansible version 2.0, modules must output JSON (key=value output is no longer allowed).

## ./library/my\_new\_test\_module

```
#!/bin/sh
set -e
source ${1} # Note, not using WANT_JSON
echo {"changed": true, "msg": "${msg}"}
exit 0
```

## ./test.yml

```
- hosts: localhost
  gather_facts: no
  tasks:
    - my_new_test_module:
      msg: "hello world"
```

```
$ ansible-playbook -vv test.yml
Using /etc/ansible/ansible.cfg as config file
1 plays in test.yml

PLAY *****

TASK [my_new_test_module] *****
changed: [localhost] => {"changed": true, "msg": "hello world"}

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

## How are modules executed?

- **task\_executor** - TaskExecutor decides if it's an action plugin or a module. If module, it loads 'Normal Action Plugin' and passes info about what's to be done.
- **Normal Action Plugin** - Inits connection. Pushes module to host. Executes the module on the remote host. *Primary coordinator.*
- **module\_common.py** - Identifies module type, selects preprocessor.
- **Module Replacer/Ansiballz** - Preprocessors which does substitutions of specific substring patterns in the module file. [Read more.](#)
- **Passing arguments** - module arguments are turned into a JSON-ified string and passed to the module.
- **Internal arguments** - parameters which implements global features. Often you do not need to know about these.

# How to Write Modules



## Example development environment

- Clone the Ansible repository:
  - `$ git clone https://github.com/ansible/ansible.git`
- Change directory into the repository root dir:
  - `$ cd ansible`
- Create a virtual environment:
  - `$ python3 -m venv venv` (or for Python 2 `$ virtualenv venv`.  
Note, this requires you to install the virtualenv package:
  - `$ pip install virtualenv`
- Activate the virtual environment:
  - `$ . venv/bin/activate`
- Install development requirements:
  - `$ pip install -r requirements.txt`
- Run the environment setup script for each new dev shell process:
  - `$ . hacking/env-setup`

## Example module development

- Navigate to the directory that you want to develop your new module in.  
Example:
  - `$ cd ~/ansible/lib/ansible/modules/cloud/my_new_test_module`
- Create your new module file:
  - `$ touch my_new_test_module`
- Create test playbook
  - `$ vi test.yml`
- Copy module to module path
  - `$ cp my_new_test_module /usr/share/ansible/plugins/modules/`
- Test your first module
  - `$ ansible-playbook -vv ./test.yml`

I deserve yet another demo.



## Module Writing Methods/Strategies

1. Wrap a CLI command (what was just demoed)
2. Use a 3rd party library
3. Interact with the API directly

(sometimes modules use more than one of these methods)

## Wrapping CLI Commands

### Pros:

- Easy to write, low learning curve.
- Protects users from complexity

### Cons:

- Output/results have to be scraped out of the CLI output, which is very fragile and prone to error

```
cli_command | awk '{ print $6 } | cut -d"/" -f2 | sed 's/old/new/'
```

- Depending on use-case - only slightly more useful than using command/shell modules.

## Using 3rd Party Libraries

### Pros:

- Also very easy to get started with, since someone else has done the hard work for you.

### Cons:

- Extra dependencies for users running your module remotely (the library must be installed everywhere you run the module).
- Modules may not cover API features you need (especially new features).
- Bugs and abandonment (don't forget to evaluate!)

## Interacting With the API Directly

### Pros:

- No extra dependencies (Ansible provides helper code in `module_utils/urls.py` to make HTTP calls).
- New features are accessible immediately without having to wait.

### Cons:

- Having to know the API and maintain the module.

# Module Testing and Debugging

## Tips for Testing and Debugging

Run your module with our module test wrapper:

```
$ hacking/test-module --help
```

Useful Options:

```
-m MODULE_PATH, --module-path=MODULE_PATH  
-a MODULE_ARGS, --args=MODULE_ARGS  
-I INTERPRETER_TYPE=INTERPRETER_PATH,  
--interpreter=INTERPRETER_TYPE=INTERPRETER_PATH  
-c, --check          run the module in check mode  
-n, --noexecute      do not run the resulting module
```



## Tips for Testing and Debugging

Run your module with `ANSIBLE_KEEP_REMOTE_FILES=1` and `-vvv`:

```
127.0.0.1 EXEC /bin/sh -c 'LANG=en_US.UTF-8 LC_ALL=en_US.UTF-8
LC_MESSAGES=en_US.UTF-8 /usr/bin/python
/home/user/.ansible/tmp/ansible-tmp-1455631745.37-113407429292636/hue; rm -rf
"/home/user/.ansible/tmp/ansible-tmp-1455631745.37-113407429292636/" > /dev/null
2>&1'
```

```
/home/user/.ansible/tmp/ansible-tmp-1455631745.37-113407429292636/hue explode
```

# Best Practices

## For Modules You Want to Contribute

- GPLv3+
- Use Python.
- Include the DOCUMENTATION string in the module (especially make sure you set the version\_added field).
- Include the EXAMPLES string in the module, and make sure the examples use the expanded YAML format (not key=value options).

## General purpose best practices

1. [http://docs.ansible.com/ansible/developing\\_modules.html](http://docs.ansible.com/ansible/developing_modules.html)
2. [http://docs.ansible.com/ansible/latest/dev\\_guide/developing\\_modules\\_best\\_practices.html](http://docs.ansible.com/ansible/latest/dev_guide/developing_modules_best_practices.html)

**Thank you**



#ANSIBLEAUTOMATES



redhat.

ANSIBLE