ANSIBLE

# ANSIBLE ESSENTIALS & BEST PRACTICES

Phil Griffiths & Patrick Harrison
Domain Solutions Architects

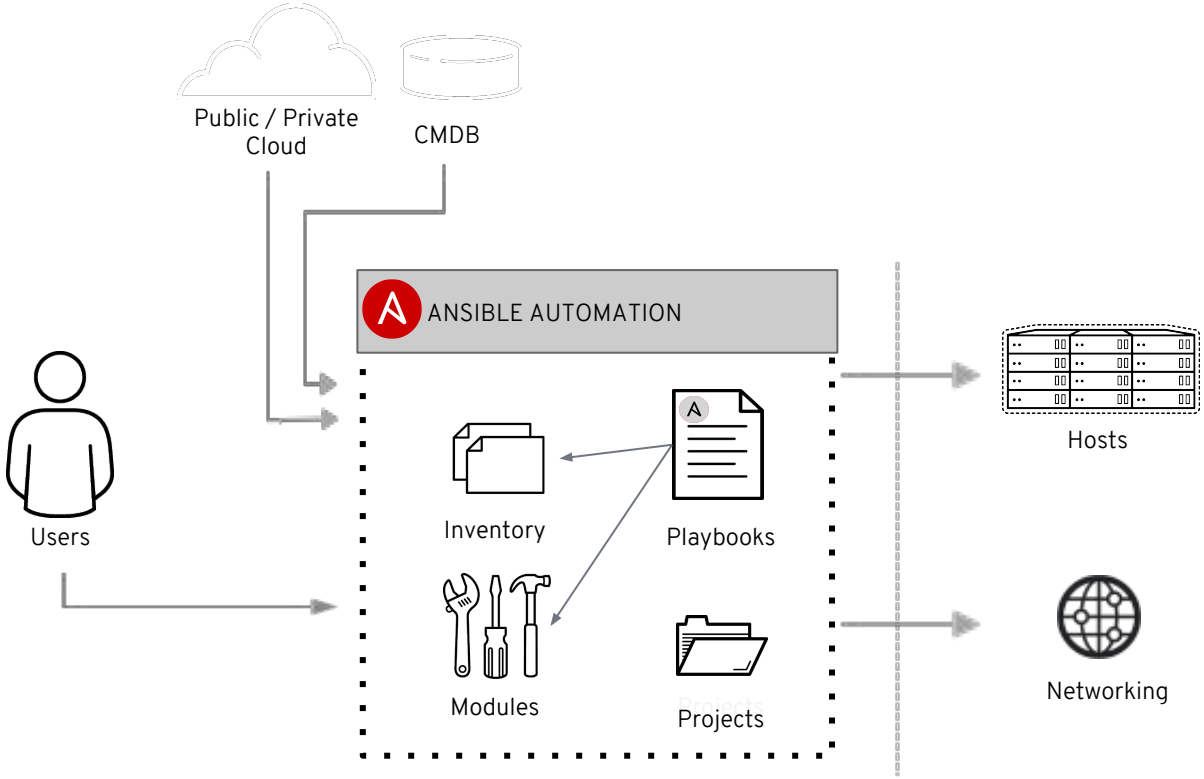
January 28, 2019

redhat

# What is Ansible?

# How Does It Work?





## Example of an ad-hoc ansible orchestration task

- **Module**: yum
- **Arguments**: name=bash state=installed

```
$ ansible localhost -m yum -a "name=bash state=installed"
localhost | SUCCESS => {
    "changed": false,
    "msg": "Nothing to do"
}
```

- What if I wanted to do more than one thing? **Playbooks**!

# Plays & Tasks



```
- hosts: web
  name: install and start apache
  tasks:
    - name: install apache packages
      yum:
        name: httpd
        state: latest

    - name: start apache service
      service:
        name: httpd
        state: started
        enabled: yes
```

```
PLAY [install and start apache]
*********************************

TASK [setup]
*********************************
ok: [web1]

TASK [install apache packages]
*********************************
ok: [web1]

TASK [start apache service]
*********************************
ok: [web1]
```

# What Can Ansible Do?

# Cross Platform

Agentless support for all Linux and Windows variants, physical, virtual, cloud and network devices

**Do this...**

| | | | | | |
|---|---|---|---|---|---|
| Provisioning | Configuration Management | Application Deployment | Orchestration | Continuous Delivery | Security and Compliance |

**On these...**

| | | | | |
|---|---|---|---|---|
| Firewalls | Load Balancers | Middleware | Applications | Containers |
| Servers | Infrastructure | Storage | Network Devices |  |

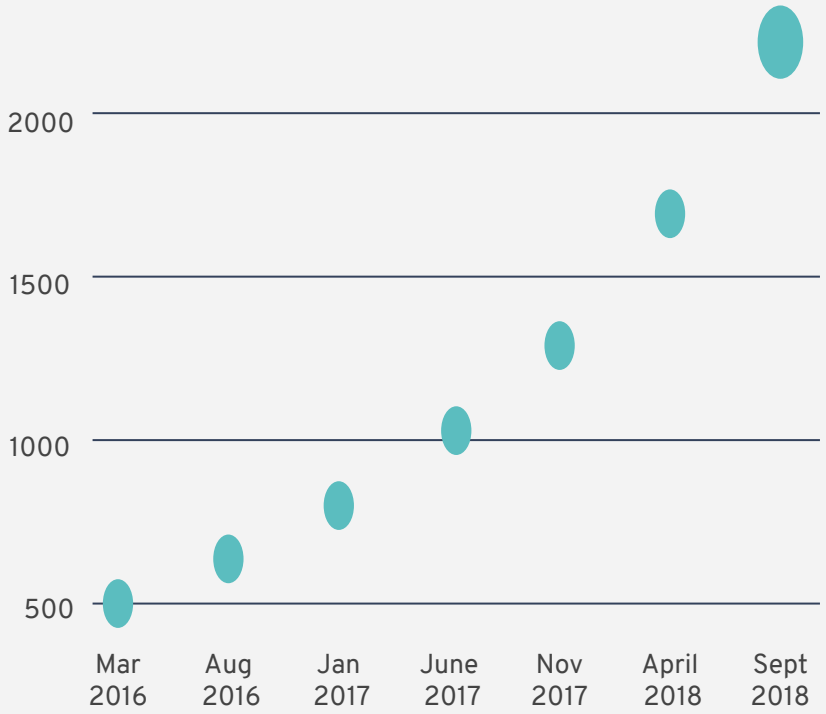
# The Mighty Module

```
$ ansible --version
```

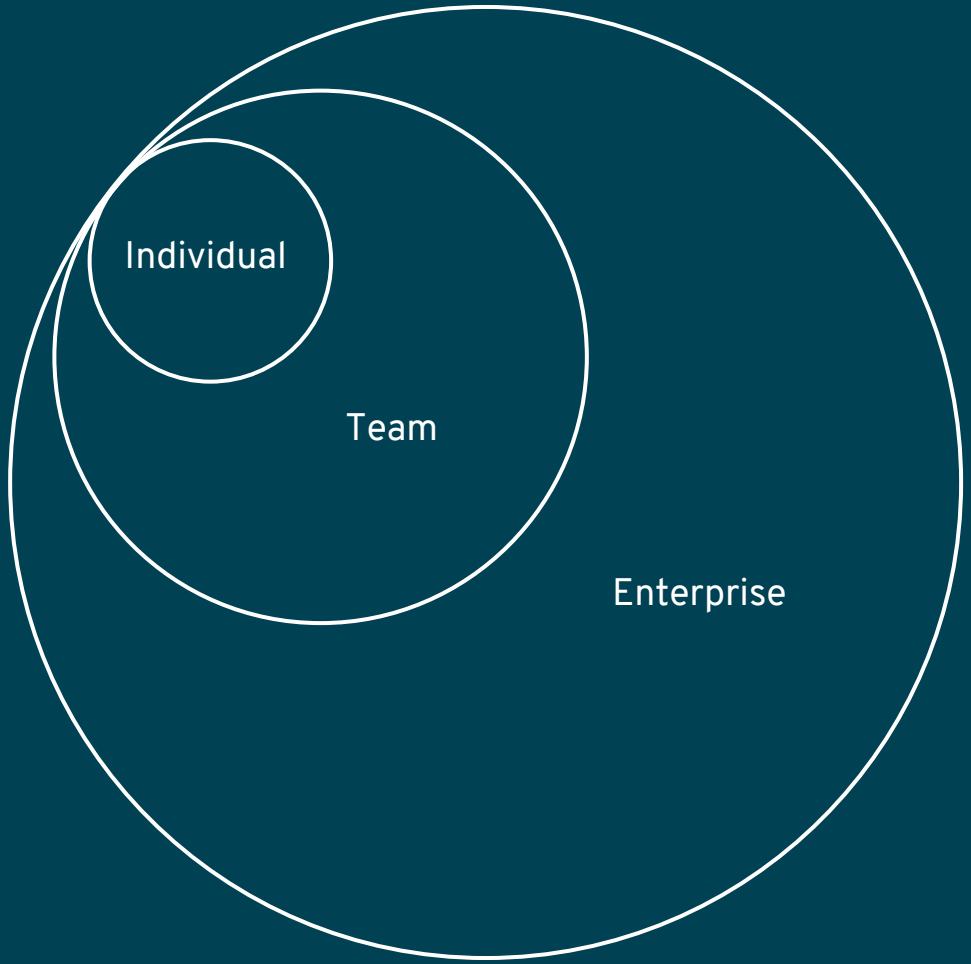
**2.7.5**

```
$ ansible-doc -l | wc -l
```

# 2146



# Successful Automation




**An enterprise-wide automation strategy** must benefit individuals first.

*“COST SAVINGS”*
*“DRIVE EFFICIENCY”*

# TIME

**Days -> Minutes**
**Repeatability**
**Error Reduction**

# START THE GROUNDSWELL



**Starting with the BIG picture is not the best path to enlightenment**

**Start the revolution from your desk**

Solving smaller problems in repeatable fashion is easier to unify

Look for quick wins, current gaps

Make easy but noticeable progress

Map out orchestration, workflows etc

# Team work


The Path To Enlightenment

# COMPLEXITY KILLS PRODUCTIVITY

That's not just a marketing slogan. We really mean it and believe that. We strive to reduce complexity in how we've designed Ansible tools and encourage you to do the same. **Strive for simplification in what you automate.**

# OPTIMIZE FOR READABILITY

If done properly, it can be the documentation of your workflow automation.

# THINK DECLARATIVELY

Ansible is a desired state engine by design. If you're trying to "write code" in your plays and roles, you're setting yourself up for failure. Our YAML-based playbooks were never meant to be for programming.

# Think of those following

# Good Karma #1

## KISS

- Keep plays and playbooks focused. Multiple simple ones are better than having a huge single playbook full of conditionals
- Once a playbook gets long or you're repeating tasks, use **roles**

# Good Karma #2

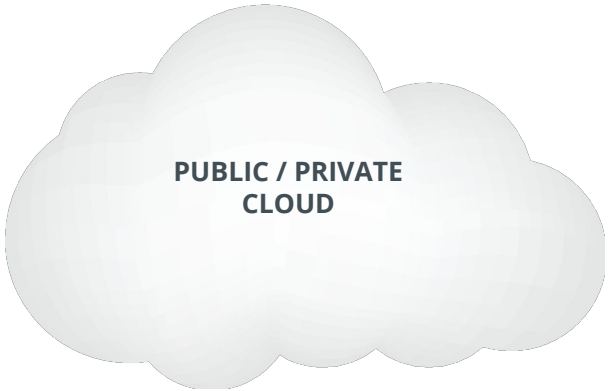
## Treat your Ansible content like code

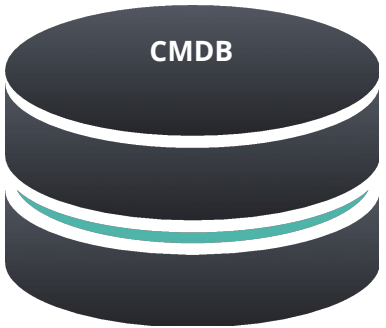
- Version control your Ansible content
- Use SCM as the control point
- Start as simple as possible and iterate
  - Start with a basic playbook and static inventory
  - Small steps, big gains

# Good Karma #3

## Use a single source of truth if you have it -- even if you have multiple sources, Ansible can unify them.

- Stay in sync automatically
- Reduce human error



## Clean up your debugging tasks

- Make them optional with the verbosity parameter so they're only displayed when they are wanted.

```
- debug:
    msg: "This always displays"

- debug:
    msg: "This only displays with ansible-playbook -vv+"
    verbosity: 2
```

## NO!

```
- name: install telegraf
  yum: name=telegraf-{{ telegraf_version }} state=present update_cache=yes disab
  notify: restart telegraf

- name: configure telegraf
  template: src=telegraf.conf.j2 dest=/etc/telegraf/telegraf.conf

- name: start telegraf
  service: name=telegraf state=started enabled=yes
```

## Yes!

```
- name: install telegraf
  yum:
    name: telegraf-{{ telegraf_version }}
    state: present
    update_cache: yes
    disable_gpg_check: yes
    enablerepo: telegraf
  notify: restart telegraf

- name: configure telegraf
  template:
    src: telegraf.conf.j2
    dest: /etc/telegraf/telegraf.conf
  notify: restart telegraf

- name: start telegraf
  service:
    name: telegraf
    state: started
    enabled: yes
```

## Don’t just start services -- use smoke tests

```
- name: check for proper response
  uri:
    url: http://localhost/myapp
    return_content: yes
  register: result
  until: '"Hello World" in result.content'
  retries: 10
  delay: 1
```

## Separate provisioning from deployment and configuration tasks

```
acme_corp/
├── configure.yml
├── provision.yml
└── site.yml
```

```
$ cat site.yml
---
- import_playbook: provision.yml
- import_playbook: configure.yml
```

## Use command modules sparingly

- Use the *run* command modules like **`shell`** and **`command`** as a last resort
- The **`command`** module is generally safer
- The **`shell`** module should only be used for I/O redirect

**Still using command a lot. Develop your own modules**

# Good Karma #9

## Jinja2 is powerful but you needn't use all of it

- Templates should be simple:
  - Variable substitution
  - Conditionals
  - Simple control structures/iterations
- Things to avoid:
  - Anything that can be done directly in Ansible
  - Managing variables in a template
  - Extensive and intricate conditionals
  - Complex nested iterations

# Good Karma #10

## Careful when mixing manual and automated configuration (Or even different automation frameworks...)

- Label template output files as being generated by Ansible

```
{{ ansible_managed | comment }}
```

```
#
# Ansible managed
#
search example.com
nameserver 192.168.122.1
```

# Good Karma #11

## Checking your playbooks

- Checking syntax:

```
ansible-playbook --syntax-check playbook.yml
```

- Checking best practices

```
ansible-lint playbook.yml
```

- Automate Testing - CI/CD

## Command line tools have their limitations

- Coordination across a distributed teams & organization…
- Controlling access to credentials…
- Track, audit and report automation and management activity…
- Provide self-service or delegation…
- Integrate automation with enterprise systems…

# Good Karma #13

## Moving from Core to Tower

Many things will work directly

Some things will need re-factoring:

vars_prompt -> surveys
includes -> workflow
vault -> Tower Credentials


Thank you