ANSIBLE

JOHN BRYCE
Leading in IT Education
a matrix company

What is Ansible?

JOHN BRYCE
Leading in IT Education
a matrix company

It's a **simple automation language** that can perfectly describe an IT application infrastructure in Ansible Playbooks.

It's an **automation engine** that runs Ansible Playbooks.

Ansible Tower is an **enterprise framework** for controlling, securing and managing your Ansible automation with a **UI and RESTful API**.

ANSIBLE

JOHN BRYCE
Leading in IT Education
a matrix company

Ansible’s neighbors

In the anon DevOps kit we have several tools for the software provisioning, configuration management and application deployment:

CHEF

puppet

ANSIBLE

SALTSTACK

What is Ansible?

SIMPLE

Human readable automation

No special coding skills needed

Tasks executed in order

Get productive quickly

POWERFUL

App deployment

Configuration management

Workflow orchestration

Orchestrate the app lifecycle

AGENTLESS

Agentless architecture

Uses OpenSSH & WinRM

No agents to exploit or update

More efficient & more secure

The Ansible Way

CROSS PLATFORM – Linux, Windows, UNIX
Agentless support for all major OS variants, physical, virtual, cloud and network

HUMAN READABLE – YAML
Perfectly describe and document every aspect of your application environment

PERFECT DESCRIPTION OF APPLICATION
Every change can be made by playbooks, ensuring everyone is on the same page

VERSION CONTROLLED
Playbooks are plain-text. Treat them like code in your existing version control.

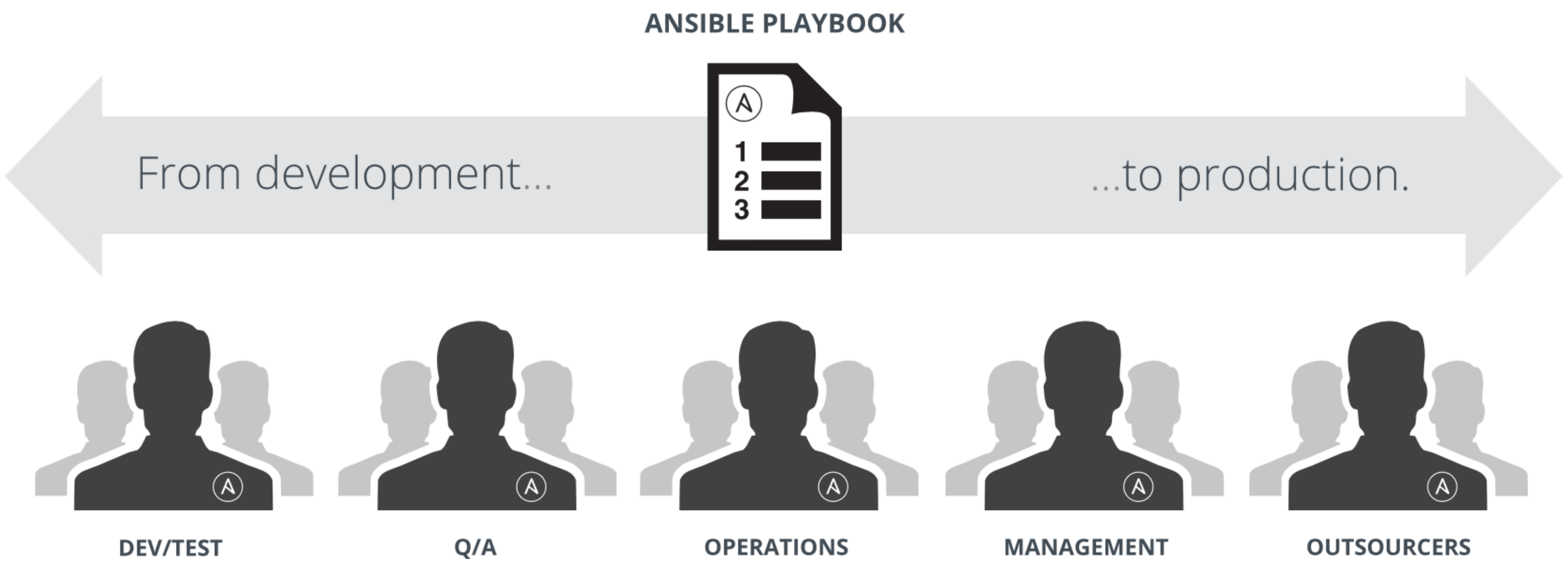
DYNAMIC INVENTORIES
Capture all the servers 100% of the time, regardless of infrastructure, location, etc.

ORCHESTRATION THAT PLAYS WELL WITH OTHERS – HP SA, Puppet, Jenkins, RHNSS, etc. Homogenize existing environments by leveraging current toolsets and update mechanisms.

Ansible: The Language of DevOps

COMMUNICATION IS THE KEY TO DEVOPS.

Ansible is the first **automation language** that can be read and written across IT. Ansible is the only **automation engine** that can automate the entire **application lifecycle** and **continuous delivery pipeline**.



Ansible included

Ansible comes bundled with hundreds of modules for a wide variety of automation tasks

- cloud
- containers
- database
- files
- messaging
- monitoring
- network
- notifications
- packaging
- source control
- system
- testing
- utilities
- web infrastructure

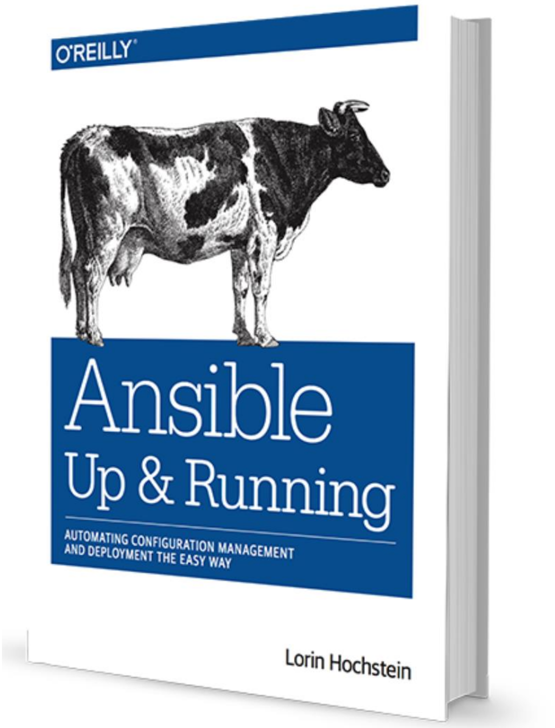
Ansible Modules control the things that you're automating. They can do everything from acting on system files, installing packages, or making API calls to a service framework. Ansible ships with over 1300 today -- and this number is always expanding with every release.

Community

JOHN BRYCE
Leading in IT Education
a matrix company

THE MOST POPULAR OPEN-SOURCE AUTOMATION COMMUNITY ON GITHUB

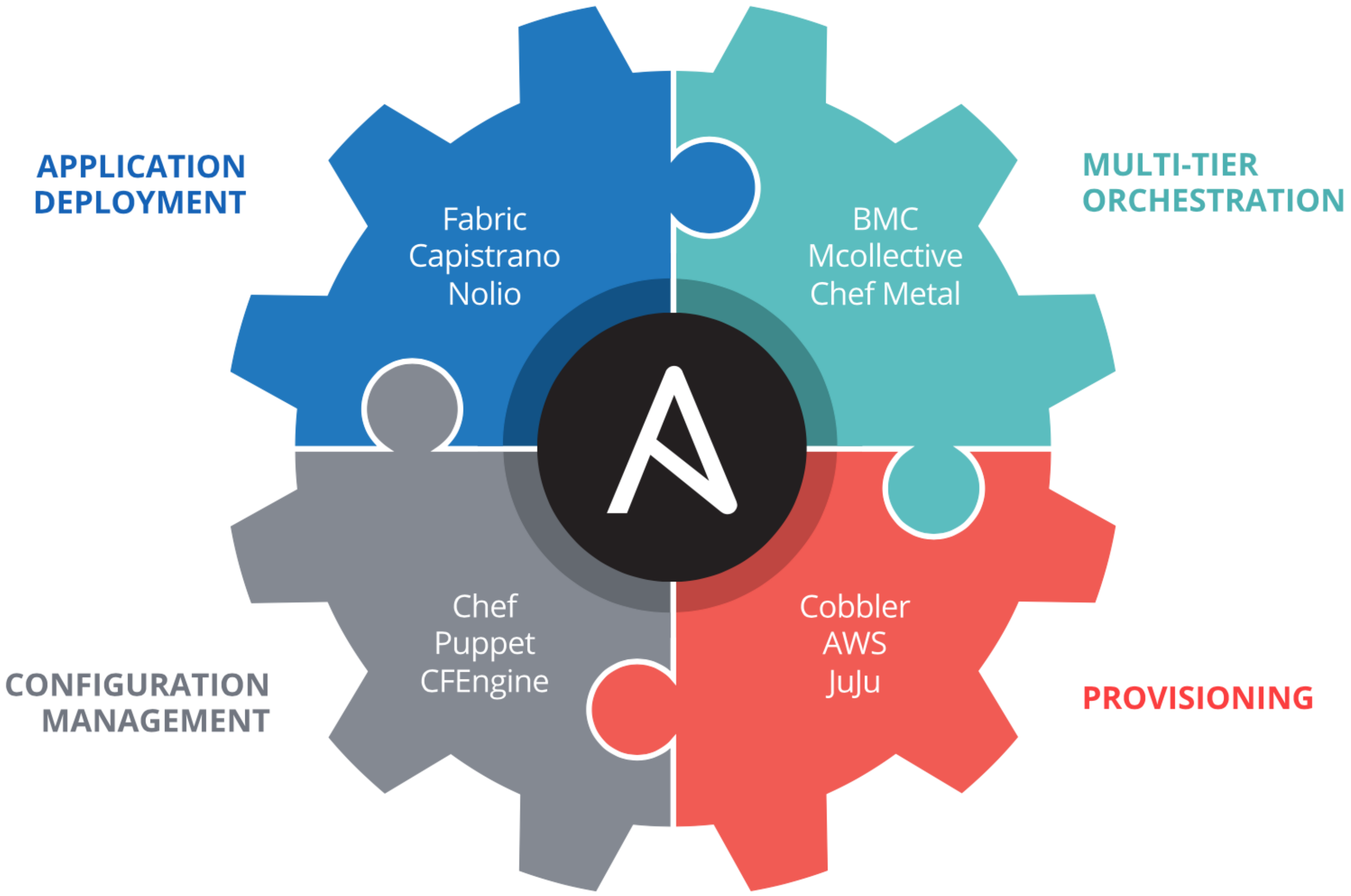
- 28,000+ stars & 10,000+ forks on GitHub
- 3200+ GitHub Contributors
- Over 1300 modules shipped with Ansible
- New contributors added every day
- 1200+ users on IRC channel
- Top 10 open source projects in 2017
- World-wide meetups taking place every week
- Ansible Galaxy: over 18,000 subscribers
- 250,000+ downloads a month
- AnsibleFest and Ansible Automates events across the globe


O'REILLY
Ansible
Up & Running
AUTOMATING CONFIGURATION MANAGEMENT AND DEPLOYMENT THE EASY WAY
Lorin Hochstein

http://ansible.com/community

JOHN BRYCE
Leading in IT Education
a matrix company

Complete Automation



Use Cases

CONFIG MANAGEMENT

Centralizing configuration file management and deployment is a common use case for Ansible, and it's how many power users are first introduced to the Ansible automation platform.

APP DEPLOYMENT

When you define your application with Ansible, and manage the deployment with Tower, teams are able to effectively manage the entire application lifecycle from development to production.

PROVISIONING

Your apps have to live somewhere. If you're PXE booting and kickstarting bare-metal servers or VMs, or creating virtual or cloud instances from templates, Ansible and Ansible Tower help streamline the process.

CONTINUOUS DELIVERY

Creating a CI/CD pipeline requires buy-in from numerous teams. You can't do it without a simple automation platform that everyone in your organization can use. Ansible Playbooks keep your applications properly deployed (and managed) throughout their entire lifecycle.

SECURITY & COMPLIANCE

When you define your security policy in Ansible, scanning and remediation of site-wide security policy can be integrated into other automated processes and instead of being an afterthought, it'll be integral in everything that is deployed.

ORCHESTRATION

Configurations alone don't define your environment. You need to define how multiple configurations interact and ensure the disparate pieces can be managed as a whole. Out of complexity and chaos, Ansible brings order.

Installing Ansible

The best way to install Ansible on CentOS, RHEL, or Scientific Linux is to configure the EPEL repository and install Ansible directly:

```
$ sudo yum install ansible
```

```
# on Debian or Ubuntu you will need the PPA repo configured
$ sudo apt-get install ansible
```

```
# on all other platforms it can be installed via pip
$ sudo pip install ansible
```

Installing Ansible

For this current course we have already created the Ansible environment on Ubuntu for you, please ask mentor to provide the credentials.

⌨ *- by this sign we would mark the practice part*

Get to the course Ansible environment using SSH CLI (command line interface):

Type the following command to get the current ansible version:

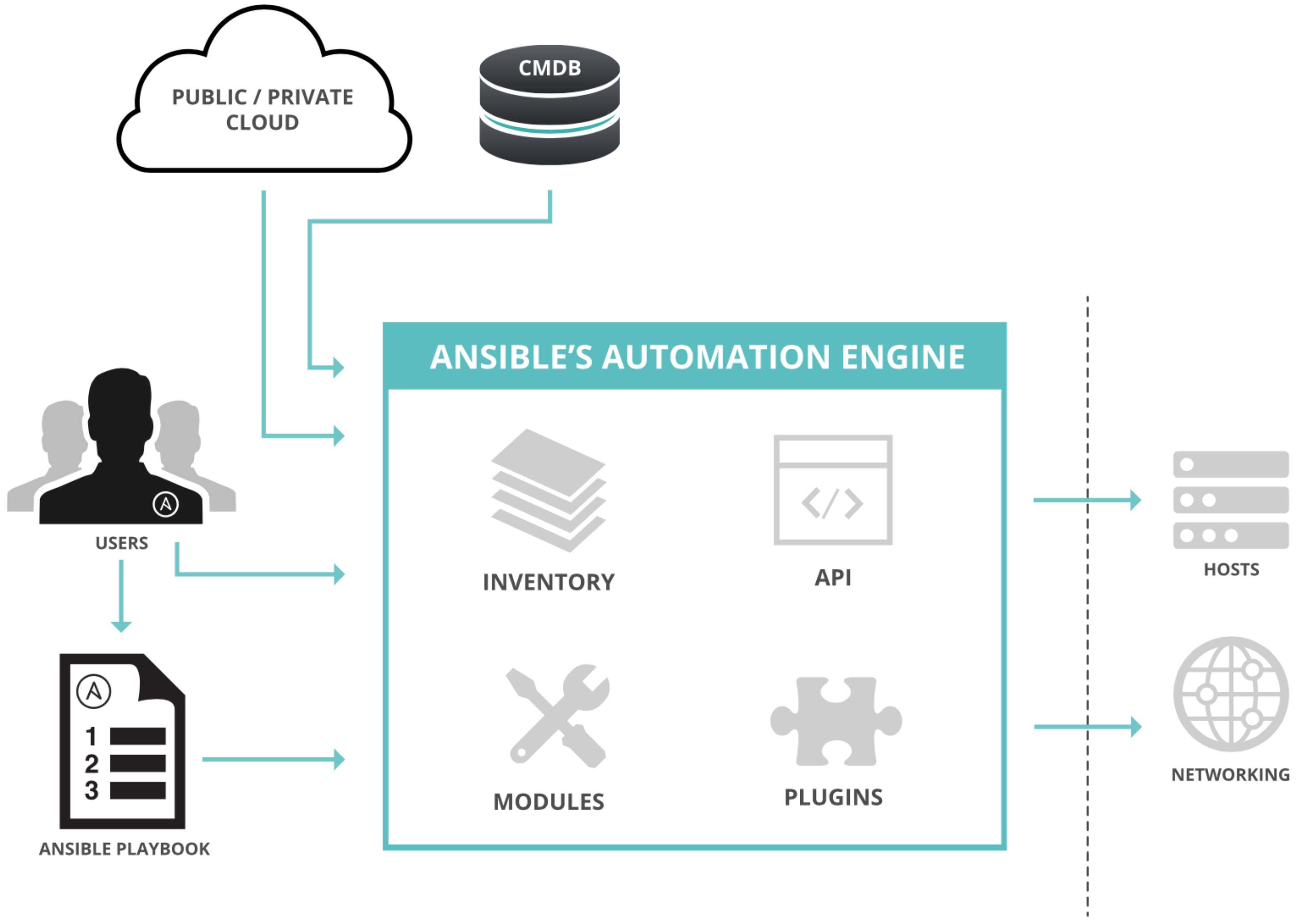
```
>ansible --version
```

The output should say like this:

```
ansible 2.6.1
config file = /etc/ansible/ansible.cfg
configured module search path = [u'/home/ubuntu/.ansible/plugins/modules',
u'/usr/share/ansible/plugins/modules’]
ansible python module location = /usr/lib/python2.7/dist-packages/ansible
executable location = /usr/bin/ansible python version = 2.7.12 (default, Dec 4 2017, 14:50:18) [GCC 5.4.0
20160609]
```

How Ansible Works

JOHN BRYCE
Leading in IT Education
a matrix company



How Ansible Works

PLAYBOOKS are written in YAML. Tasks are executed sequentially and invokes Ansible modules.

MODULES are “tools in the toolkit” Python, Powershell or any language Extended Ansible simplicity to entire stack.

PLUGINS are ”gears in the engine”. Code that plugs the core engine. Adaptability for various uses & platforms.

INVENTORY gets acquainted your Ansible core within the structured set of infrastructure to work on.

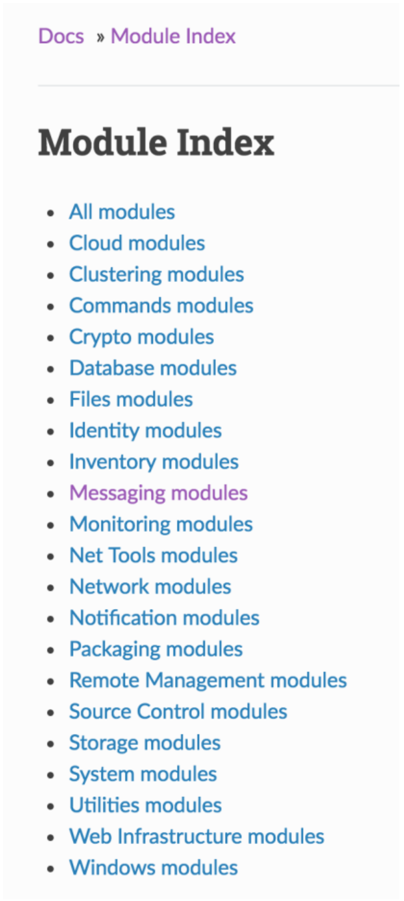
Modules

Modules are bits of code transferred to the target system and executed to satisfy the task declaration.

- apt/yum
- copy
- file
- get_url
- git
- ping
- debug
- service
- synchronize
- template
- uri
- user
- wait_for
- assert

Modules Documentation

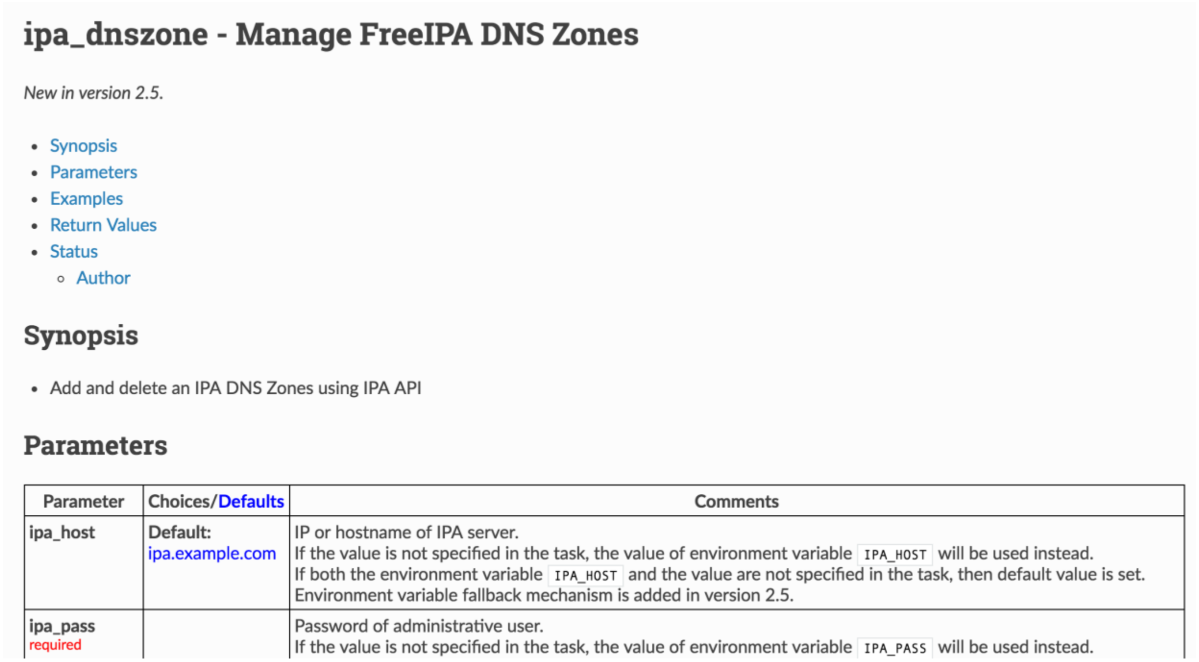
http://docs.ansible.com/



Docs » Module Index

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



ipa_dnszone - Manage FreeIPA DNS Zones

New in version 2.5.

- Synopsis
- Parameters
- Examples
- Return Values
- Status
  - Author

Synopsis

- Add and delete an IPA DNS Zones using IPA API

Parameters

Parameter	Choices/Defaults	Comments
ipa_host	Default: ipa.example.com	IP or hostname of IPA server. If the value is not specified in the task, the value of environment variable `IPA_HOST` will be used instead. If both the environment variable `IPA_HOST` and the value are not specified in the task, then default value is set. Environment variable fallback mechanism is added in version 2.5.
ipa_pass required		Password of administrative user. If the value is not specified in the task, the value of environment variable `IPA_PASS` will be used instead.

Modules

To retrieve all the ansible modules that are currently installed on the environment please run:

```
>ansible-doc -l
```

The output should say like this:

```
a10_server           Manage A10 Networks AX/SoftAX/Thunder/vThunder devices' server object.
a10_server_axapi3    Manage A10 Networks AX/SoftAX/Thunder/vThunder devices
a10_service_group    Manage A10 Networks AX/SoftAX/Thunder/vThunder devices' service
…
```

* It takes a while to get the list

Modules: Run Commands

If Ansible doesn't have a module that suits your needs there are the "run command" modules:

command: Takes the command and executes it on the host. The most secure and predictable.

shell: Executes through a shell like /bin/sh so you can use pipes etc. Be careful.

script: Runs a local script on a remote node after transferring it.

raw: Executes a command without going through the Ansible module subsystem.

NOTE: Unlike standard modules, run commands have no concept of desired state and should only be used as a last resort.

Inventory

Inventory is a collection of hosts (nodes) with associated data and groupings that Ansible can connect and manage.

- Hosts (nodes)
- Groups
- Inventory-specific data (variables)
- Static or dynamic sources

Inventory

```
[control]
control ansible_host=10.42.0.2

[web]
node-[1:3] ansible_host=10.42.0.[6:8]

[haproxy]
haproxy ansible_host=10.42.0.100

[all:vars]
ansible_user=vagrant
ansible_ssh_private_key_file=~/.vagrant.d/insecure_private_key
```

Inventory

To play ansible on the same host we configured the inventory in a following way:

```
>cat /home/ubuntu/hosts
```

The output should say like this:

```
[local]
localhost ansible_connection=local
```

Ad-Hoc Commands

An **ad-hoc command** is a single Ansible task to perform quickly, but don't want to save for later:

```
# check all my inventory hosts are ready to be managed by Ansible
>ansible all -m ping
WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'
# collect and display the discovered facts for the localhost
>ansible localhost -m setup
# run the uptime command on all hosts on the localhost
>ansible localhost -m command -a "uptime"
localhost | SUCCESS | rc=0 >> 10:22:28 up 46 min, 1 user, load average: 0.01, 0.02, 0.03
```

Sidebar: Discovered Facts

Facts are bits of information derived from examining a host systems that are stored as variables for later use in a play:

>ansible localhost -m setup

```
localhost | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "172.17.0.1",
            "172.31.44.83",
            "172.24.0.1"
        ],
        "ansible_all_ipv6_addresses": [
            "fe80::47a:67ff:fed7:c9ec"
        ],
        "ansible_apparmor": {
            "status": "enabled"
        },
```

Variables

Ansible can work with metadata from various sources and manage their context in the form of variables.

- Command line parameters
- Plays and tasks
- Files
- Inventory
- Discovered facts
- Roles

Variable Precedence

The order in which the same variable from different sources will override each other.

1. extra vars
2. task vars (only for the task)
3. block vars (only for tasks in block)
4. role and include vars
5. play vars_files
6. play vars_prompt
7. play vars
8. set_facts
9. registered vars
10. host facts
11. playbook host_vars
12. playbook group_vars
13. inventory host_vars
14. inventory group_vars
15. inventory vars
16. role defaults

Tasks

Tasks are the application of a module to perform a specific unit of work.

- **file**: A directory should exist
- **yum**: A package should be installed
- **service**: A service should be running
- **template**: Render a configuration file from a template
- **get_url**: Fetch an archive file from a URL
- **git**: Clone a source code repository

Example Tasks in a Play

```
tasks:
- name: Ensure httpd package is present
  yum:
    name: httpd
    state: latest


- name: Ensure latest index.html file is present
  copy:
    src: files/index.html
    dest: /var/www/html/


- name: Restart httpd
  service:
    name: httpd
    state: restarted
```

Handler Tasks

Handlers are special tasks that run at the end of a play if notified by another task when a change occurs:

```
tasks:
- name: Ensure httpd package is present
  yum:
    name: httpd
    state: latest
  notify: restart httpd

- name: Ensure latest index.html file is present
  copy:
    src: files/index.html
    dest: /var/www/html/

handlers:
- name: restart-httpd
  service:
    name: httpd
```

Plays & Playbooks

Plays are ordered sets of tasks to execute against host selections from your inventory. A playbook is a file containing one or more plays:

```
---
- name: Ensure apache is installed and started
  hosts: web
  become: yes
  vars:
    http_port: 80

  tasks:
  - name: Ensure httpd package is present
    yum:
      name: httpd
      state: latest

  - name: Ensure latest index.html file is present
    copy:
      src: files/index.html
```

Doing More with Playbooks

Templates

Ansible embeds the Jinja2 templating engine that can be used to dynamically:

- Set and modify play variables
- Conditional logic
- Generate files such as configurations from variables

Doing More with Playbooks

Loops

Loops can do one task on multiple things, such as create a lot of users, install a lot of packages, or repeat a polling step until a certain result is reached.

```
- yum:
    name: "{{ item }}"
    state: latest
  with_items:
  - httpd
  - mod_wsgi
```

Doing More with Playbooks

Conditionals

Ansible supports the conditional execution of a task based on the run-time evaluation of variable, fact, or previous task result.

```
- yum:
    name: httpd
    state: latest
  when: ansible_os_family == "RedHat"
```

Doing More with Playbooks

Tags

Tags are useful to be able to run a subset of a playbook on-demand.

```
- yum:
    name: "{{ item }}"
    state: latest
  with_items:
  - httpd
  - mod_wsgi
  tags:
     - packages
```

Doing More with Playbooks

Blocks

Blocks cut down on repetitive task directives, allow for logical grouping of tasks and even in play error handling.

```
- block:
  - yum:
      name: "{{ item }}"
      state: latest
    with_items:
    - httpd
    - mod_wsgi


  - template:
      src: templates/httpd.conf.j2
      dest: /etc/httpd/conf/httpd.conf
  when: ansible_os_family == "RedHat"
```

Roles

Roles are packages of closely related Ansible content that can be shared more easily than plays alone.

- Improves readability and maintainability of complex plays
- Eases sharing, reuse and standardization of automation processes
- Enables Ansible content to exist independently of playbooks, projects -- even organizations
- Provides functional conveniences such as file path resolution and default values

Ansible Galaxy

Ansible Galaxy is a hub for finding, reusing and sharing Ansible content including roles.

Jump-start your automation project with content contributed and reviewed by the Ansible community.

http://galaxy.ansible.com

Ansible Galaxy

To install your first ansible role, please run:

```
>ansible-galaxy install jdauphant.nginx
```

**- This role installs and configures the nginx web server*

The output should say like this:

```
- downloading role 'nginx', owned by jdauphant
- downloading role from https://github.com/jdauphant/ansible-role-nginx/archive/v2.18.1.tar.gz
- extracting jdauphant.nginx to /home/ubuntu/.ansible/roles/jdauphant.nginx
- jdauphant.nginx (v2.18.1) was installed successfully
```

See the role there: ***/home/ubuntu/.ansible/roles***

Ansible Galaxy

To use the just installed role, please create the following playbook:

```
>vi /home/ubuntu/my_first_playbook.yml
```

```
- hosts: localhost
  roles:
  - role: jdauphant.nginx
    nginx_http_params:
      - sendfile "on"
      - access_log "/var/log/nginx/access.log"
    nginx_sites:
      default:
        - listen 8089
```

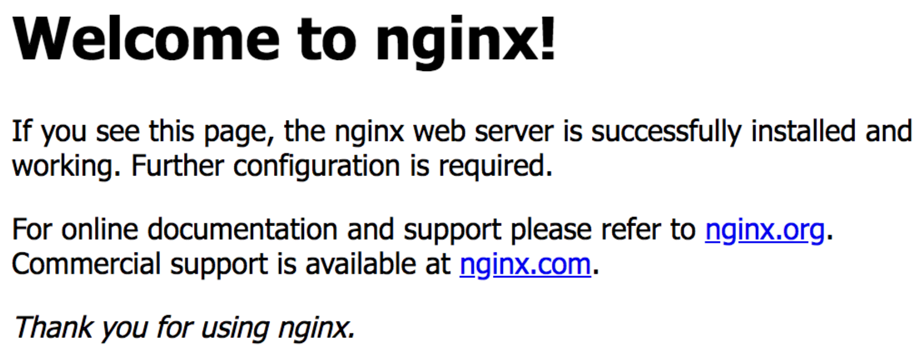
```
>cd /home/ubuntu
> sudo ansible-playbook my_first_playbook.yml -i ./hosts
```

It installs the nginx web server to listen to the ***8089*** *port*

Ansible Galaxy

Please check the installation by drilling down the link:

http://<your server IP>:8089/



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Stop and remove the nginx by:

>sudo service nginx stop

>sudo apt-get remove -y nginx

Next Steps

- It's easy to get started

  ansible.com/get-started

- Join the Ansible community

  ansible.com/community

- Would you like to learn a lot more?

  redhat.com/en/services/training/do407-automation-ansible

Thanks!!!

JOHN BRYCE
Leading in IT Education
a matrix company