Introduction to Ansible

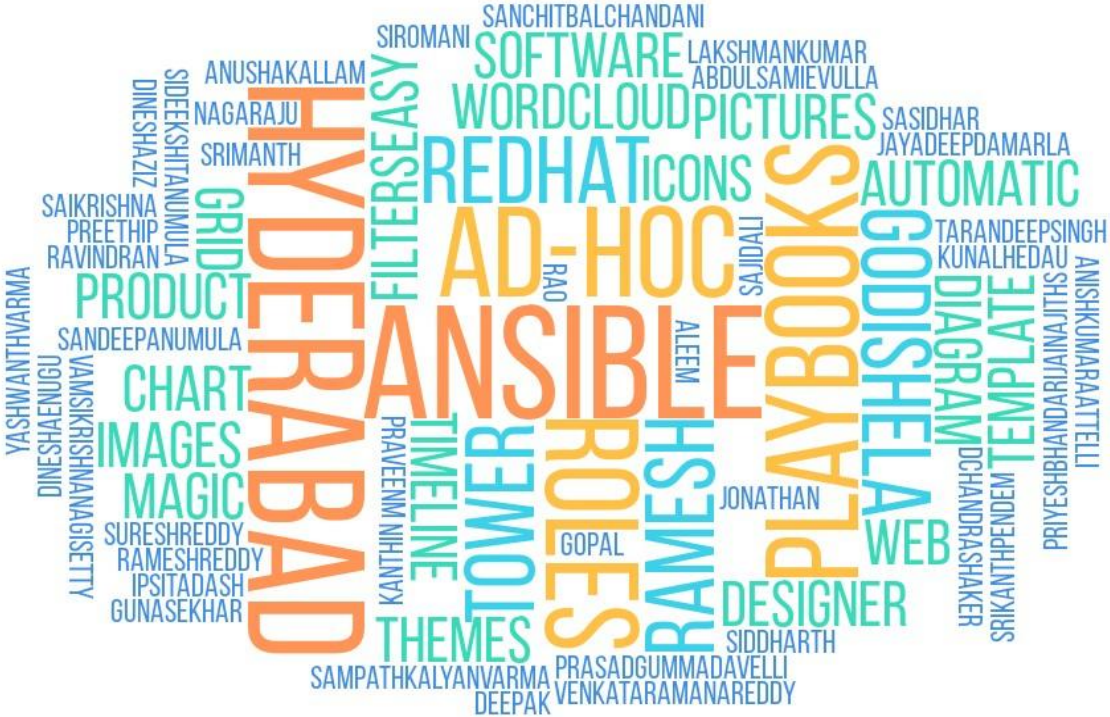
What is Ansible?

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 is an automation platform:

- Playbooks make up the automation language
- The code base is the automation engine.
- Ansible Tower manages existing automation



Who are We?

Exists from June 2015

606 members strong community

Created to serve the community with greater purpose

17 meetup's so far

Need Speakers and Successful use cases

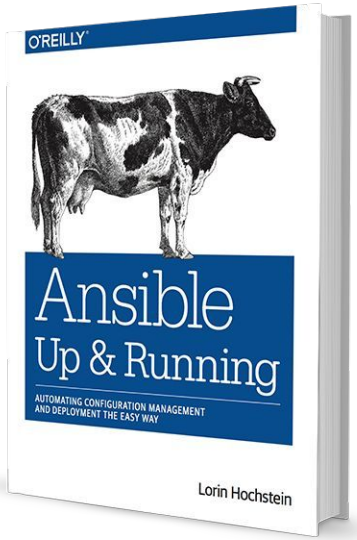
Code is available in github

www.github.com/rago-actions/ansible-automation

Community

THE MOST POPULAR OPEN-SOURCE AUTOMATION COMMUNITY ON GITHUB

- 33,500+ stars & 11,000+ forks on GitHub
- 3,300+ GitHub Contributors
- Over 3000 modules shipped with Ansible
- New contributors added every day
- 1,500+ users on IRC channel
- Top 10 open source projects in 2017
- World-wide meetups taking place every week
- Ansible Galaxy: over 18,000 subscribers
- 500,000+ downloads a month
- AnsibleFests in Austin, NYC, SF, London



http://ansible.com/community

Why Ansible? What Sets Ansible Apart?

SIMPLE

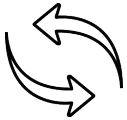
Human readable automation

No special coding skills needed

Tasks executed in order

Usable by every team

Get productive quickly



POWERFUL

App deployment

Configuration management

Workflow orchestration

Network automation

Orchestrate the app lifecycle

AGENTLESS

Agentless architecture

Uses OpenSSH & WinRM

No agents to exploit or update

Get started immediately

More efficient & more secure

WHAT CAN I DO USING ANSIBLE?

Automate the deployment and management of your entire IT footprint.

Do this...

Orchestration	Configuration Management	Application Deployment	Provisioning	Continuous Delivery	Security and Compliance

On these...

Firewalls	Load Balancers	Applications	Containers	Clouds
Servers	Infrastructure	Storage	Network Devices	And more...

ANSIBLE AUTOMATES TECHNOLOGIES YOU USE

Time to automate is measured in minutes

CLOUD

AWS
Azure
Digital Ocean
Google
OpenStack
Rackspace
+more

OPERATING SYSTEMS

RHEL and
Linux UNIX
Windows
+more

VIRT & CONTAINER

Docker
VMware RHV
OpenStack
OpenShift
+more

STORAGE

NetApp
Red Hat Storage
Infinidat
+more

WINDOWS

ACLs
Files
Packages IIS
Regedits
Shares
Services
Configs
Users
Domains
+more

NETWORK

Arista
A10
Cumulus
Bigswitch
Cisco
Cumulus
Dell
F5
Juniper
Palo Alto
OpenSwitch
+more

DEVOPS

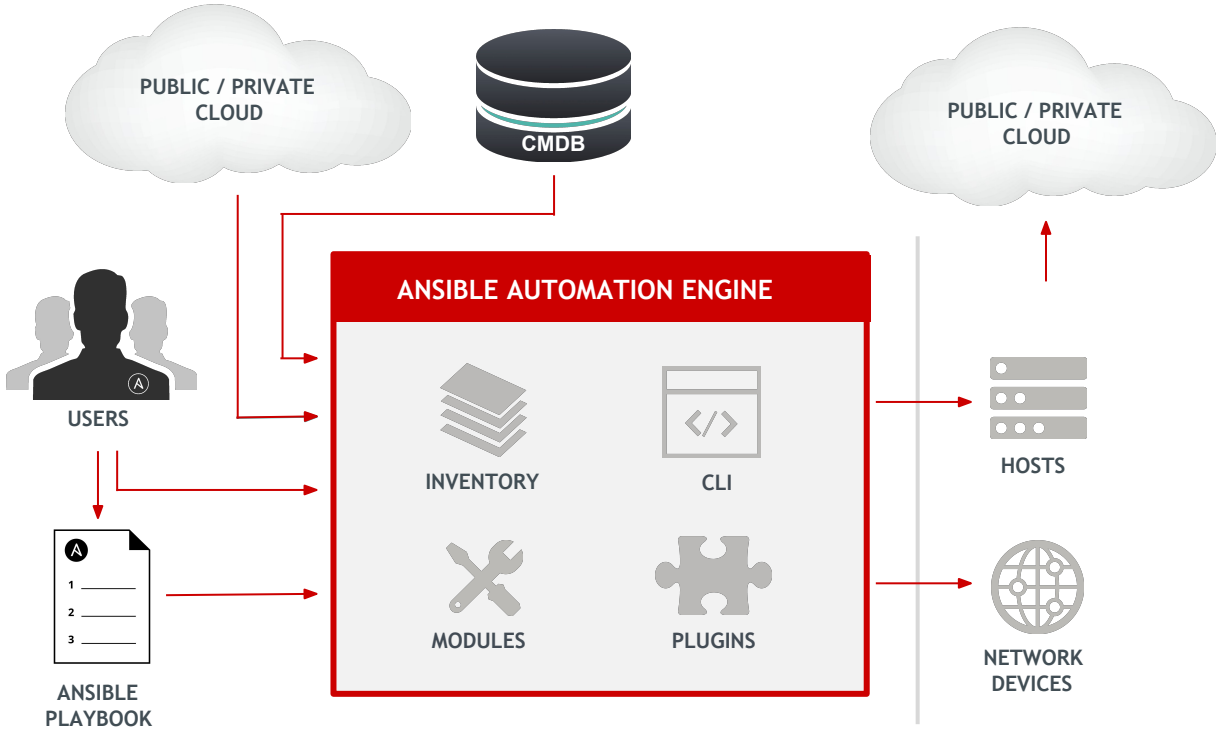
Jira
GitHub
Vagrant
Jenkins
Bamboo
Atlassian
Subversion
Slack
Hipchat
+more

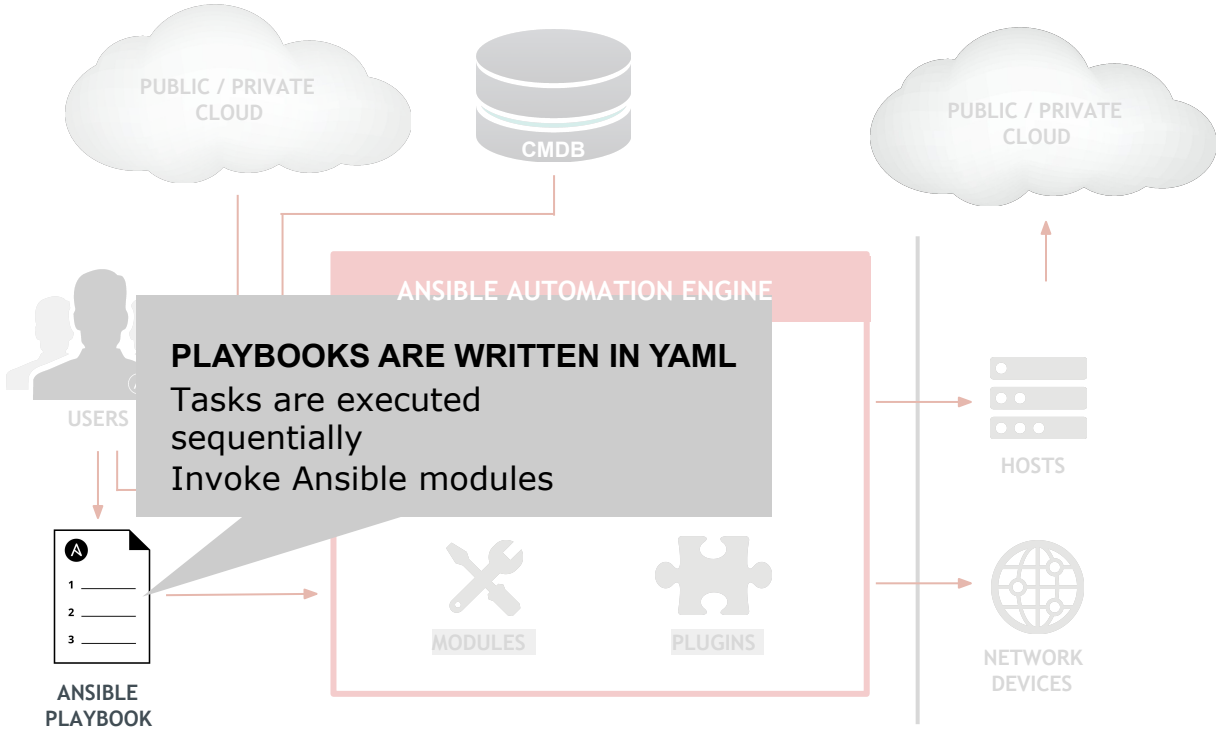
MONITORING

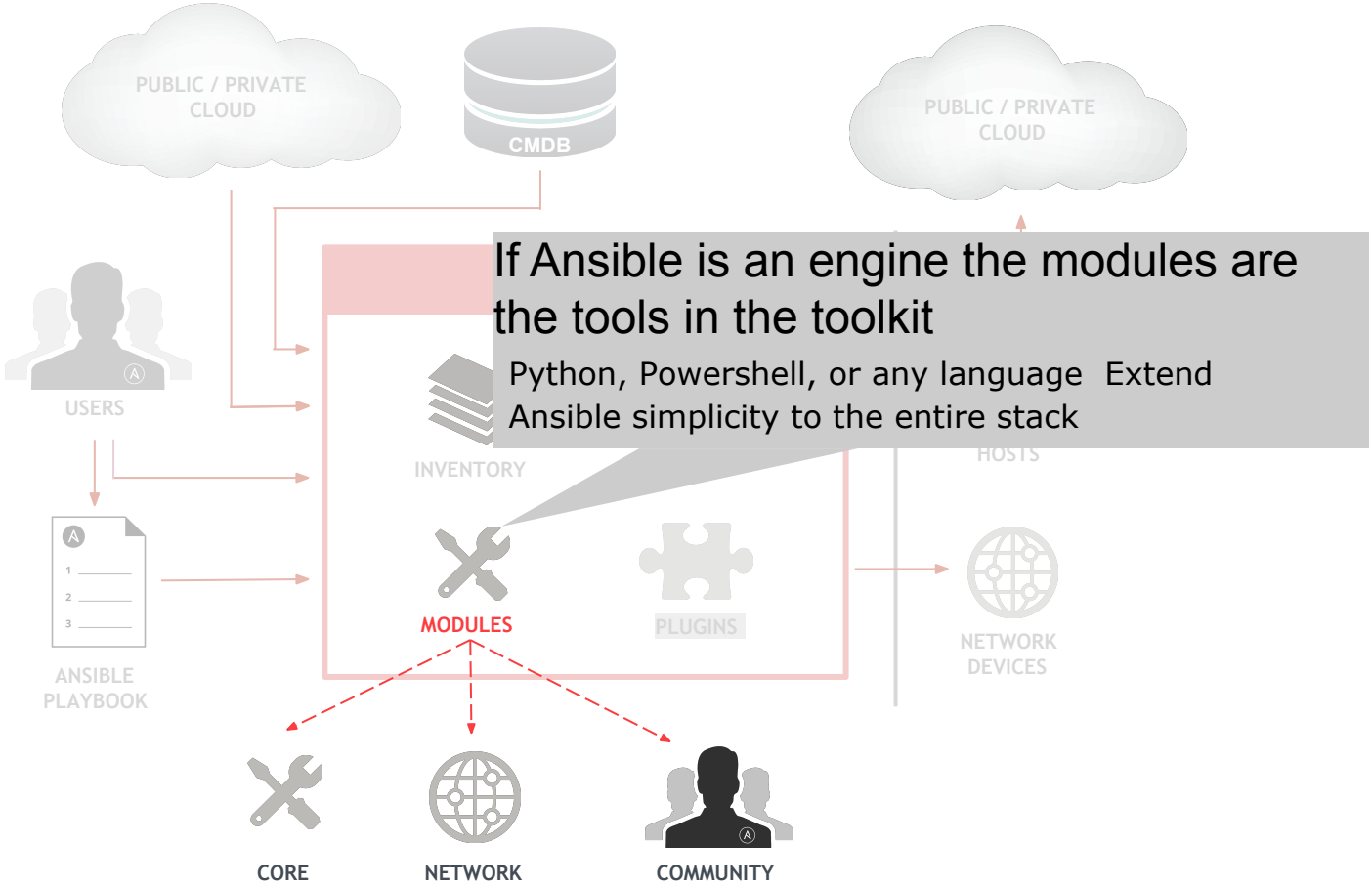
Dynatrace
Airbrake
BigPanda
Datadog
LogicMonitor
Nagios
New Relic
PagerDuty
Sensu
StackDriver
Zabbix
+more

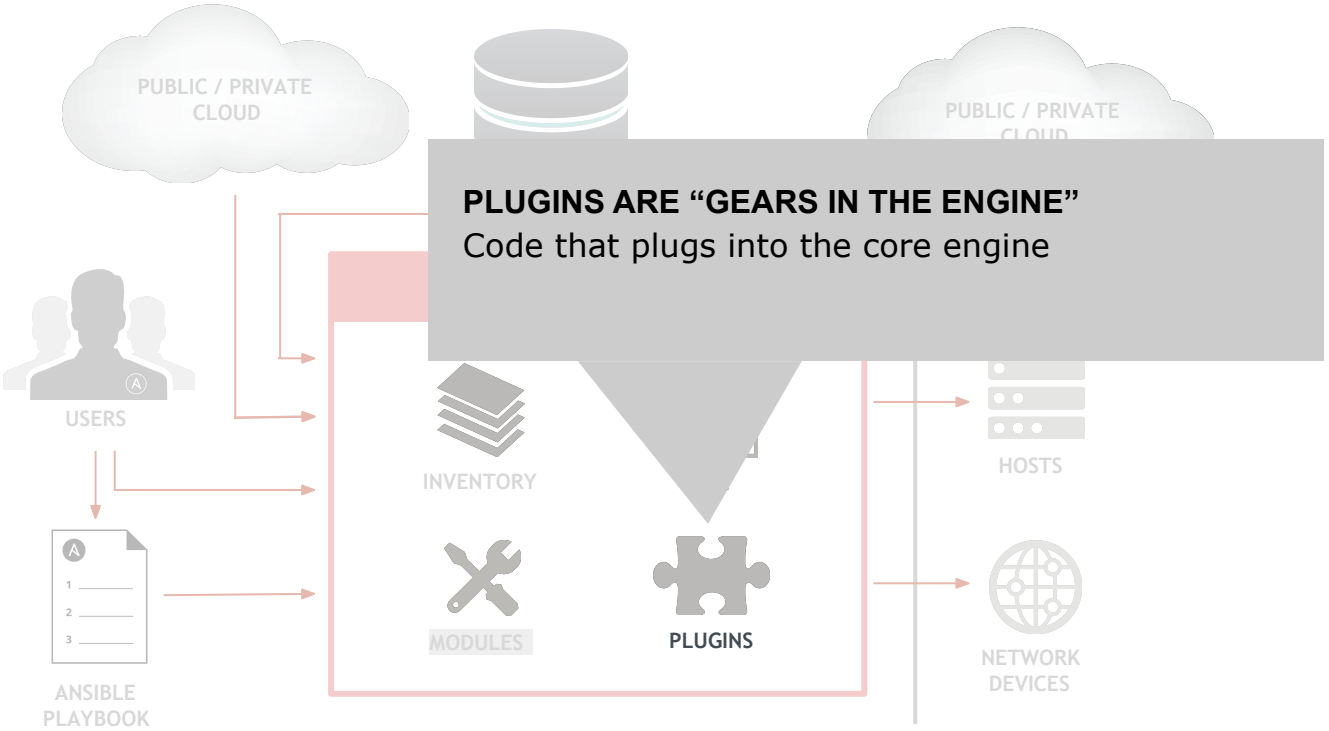
Installing Ansible

```
# you will need the extras repo configured on RHEL,
# along with the Ansible Engine repository on RHEL 7
$ sudo yum install ansible

# Ansible can be installed via “pip”, the Python
# package manager
$ sudo pip install ansible

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


PUBLIC / PRIVATE CLOUD
CMDB
PUBLIC / PRIVATE CLOUD
ANSIBLE AUTOMATION ENGINE
USERS
INVENTORY
CLI
HOSTS
MODULES
PLUGINS
ANSIBLE PLAYBOOK
NETWORK DEVICES

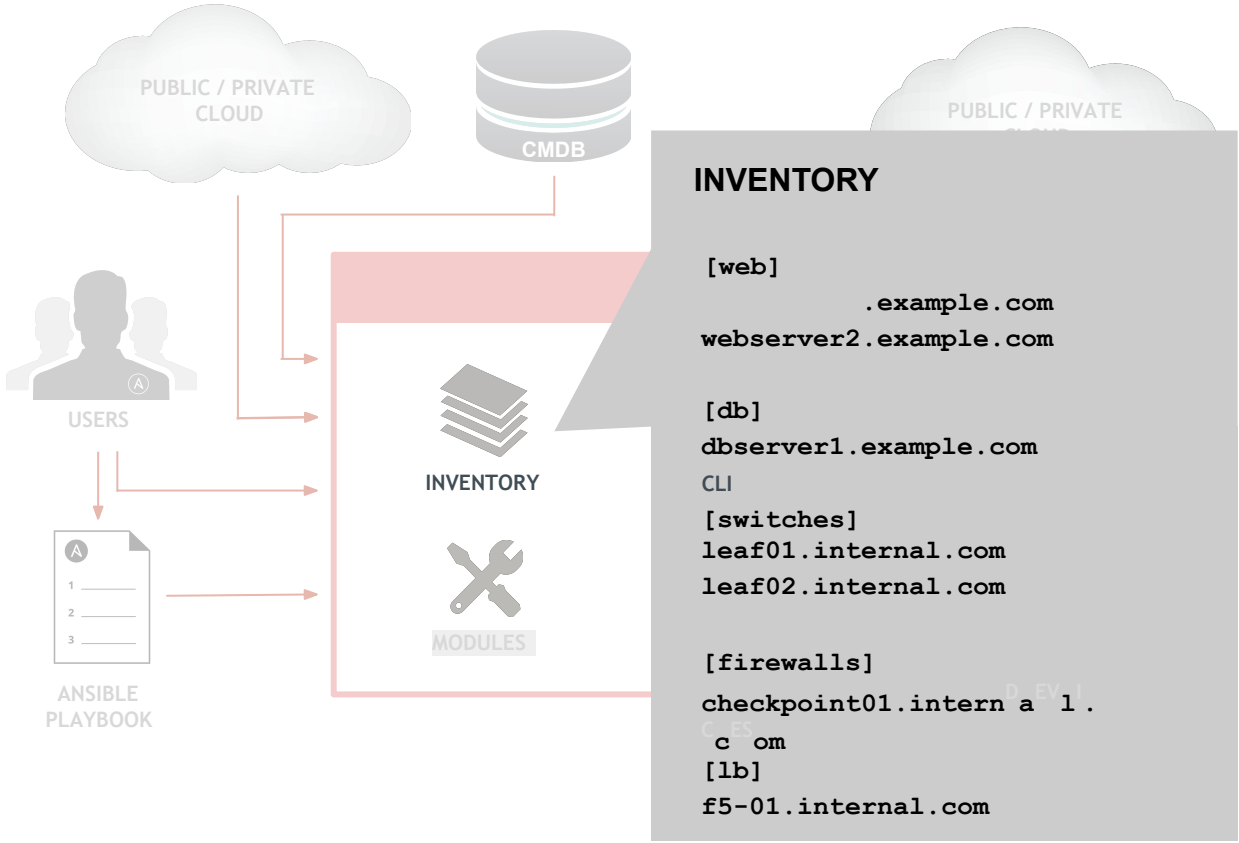

PUBLIC / PRIVATE CLOUD
CMDB
PUBLIC / PRIVATE CLOUD
ANSIBLE AUTOMATION ENGINE
USERS
PLAYBOOKS ARE WRITTEN IN YAML
Tasks are executed sequentially
Invoke Ansible modules
HOSTS
MODULES
PLUGINS
NETWORK DEVICES
ANSIBLE PLAYBOOK

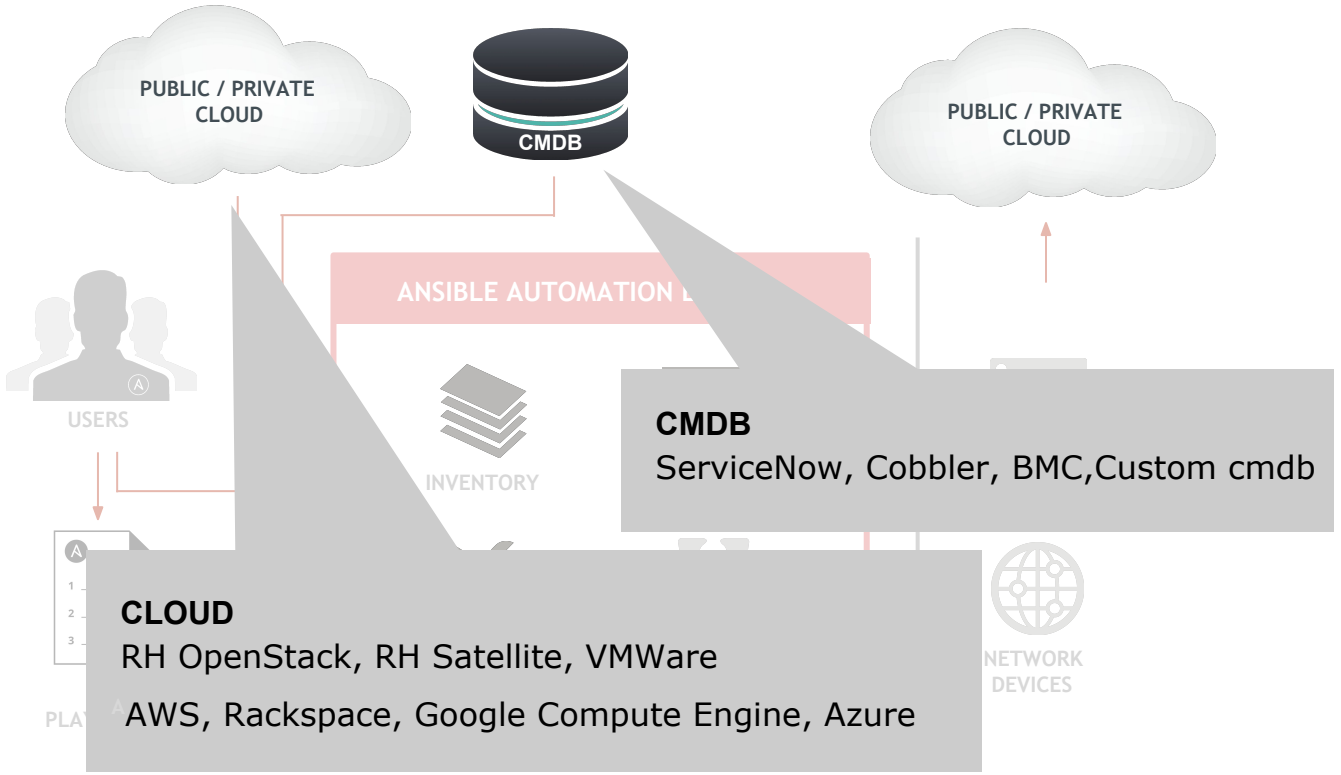

PUBLIC / PRIVATE CLOUD
CMDB
PUBLIC / PRIVATE CLOUD
If Ansible is an engine the modules are the tools in the toolkit
Python, Powershell, or any language Extend Ansible simplicity to the entire stack
USERS
INVENTORY
HOSTS
ANSIBLE PLAYBOOK
MODULES
PLUGINS
NETWORK DEVICES
CORE
NETWORK
COMMUNITY

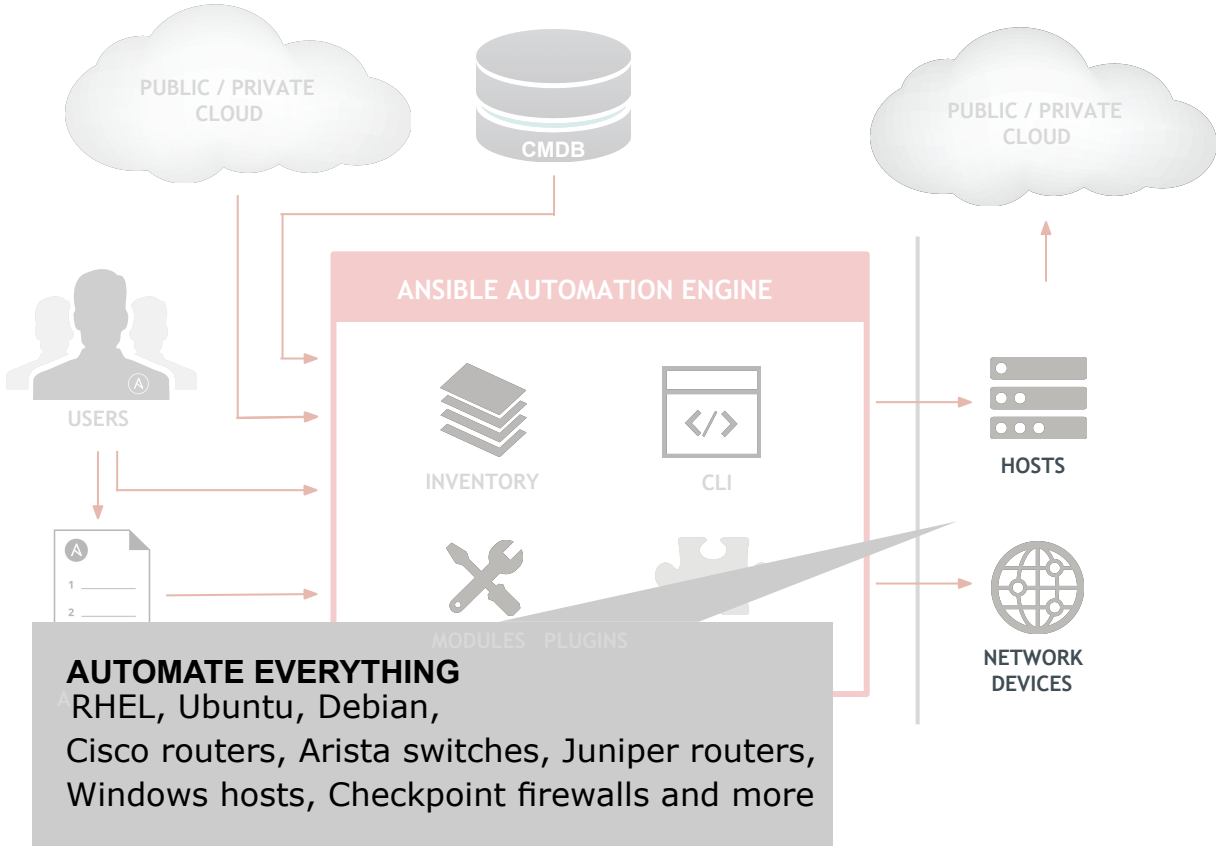


PLUGINS ARE “GEARS IN THE ENGINE”

Code that plugs into the core engine


PUBLIC / PRIVATE
CLOUD
CMDB
PUBLIC / PRIVATE
USERS
INVENTORY
MODULES
ANSIBLE
PLAYBOOK
CLI
INVENTORY
[web]
.example.com
webserver2.example.com
[db]
dbserver1.example.com
[switches]
leaf01.internal.com
leaf02.internal.com
[firewalls]
checkpoint01.intern a l.
c om
[lb]
f5-01.internal.com


PUBLIC / PRIVATE CLOUD
CMDB
PUBLIC / PRIVATE CLOUD
ANSIBLE AUTOMATION
USERS
INVENTORY
NETWORK DEVICES
CMDB
ServiceNow, Cobbler, BMC,Custom cmdb
CLOUD
RH OpenStack, RH Satellite, VMWare
AWS, Rackspace, Google Compute Engine, Azure


PUBLIC / PRIVATE CLOUD
CMDB
PUBLIC / PRIVATE CLOUD
ANSIBLE AUTOMATION ENGINE
USERS
INVENTORY
CLI
HOSTS
MODULES
PLUGINS
NETWORK DEVICES
AUTOMATE EVERYTHING
RHEL, Ubuntu, Debian,
Cisco routers, Arista switches, Juniper routers,
Windows hosts, Checkpoint firewalls and more

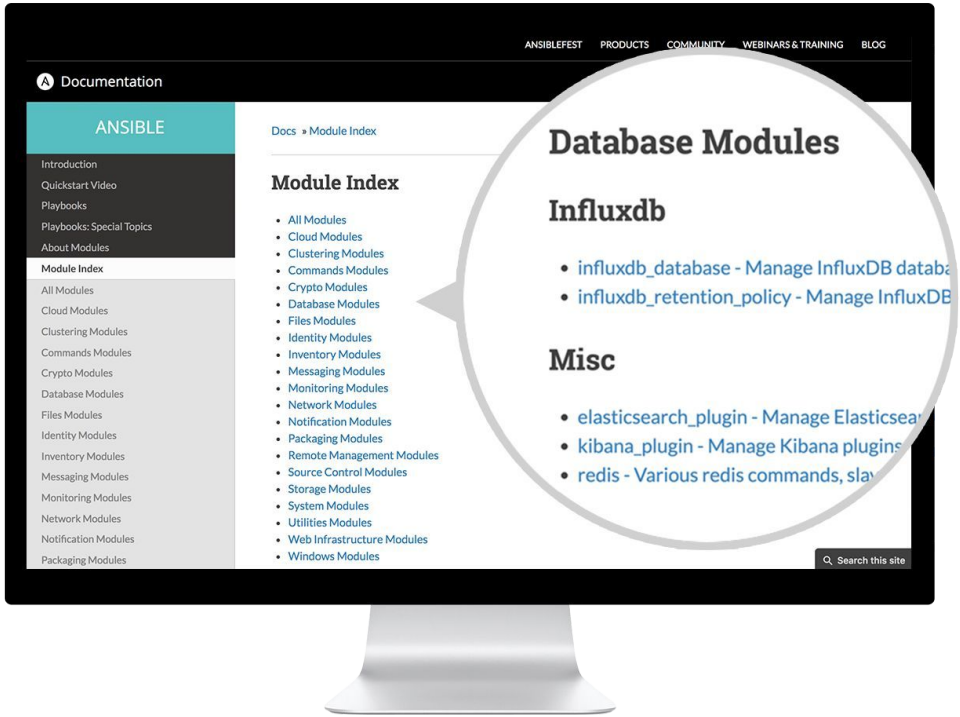
Modules

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

docs.ansible.com

ANSIBLE



redhat.

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

Static Inventory Example

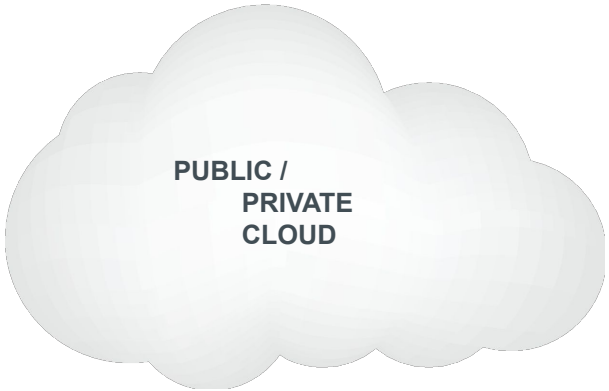
```
10.42.0.2
10.42.0.6
10.42.0.7
10.42.0.8
10.42.0.100
host.example.com
```

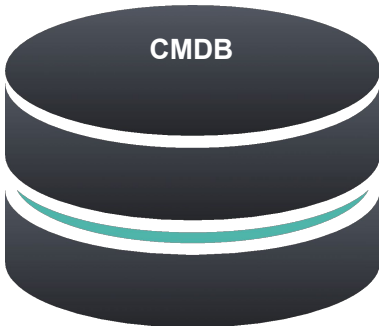
Static Inventory Example

```
[control]
tower 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
```

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





Ad-Hoc Commands

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

Ad-Hoc Commands

```
# check all my inventory hosts are ready to be
# managed by Ansible
$ ansible all -m ping

# collect and display the discovered facts
# for the localhost
$ ansible localhost -m setup

# run the uptime command on all hosts in the
# web group
$ ansible web -m command -a "uptime"
```

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_default_ipv4": {
          "address": "192.168.1.37",
          "alias": "wlan0",
          "gateway": "192.168.1.1",
          "interface": "wlan0",
          "macaddress": "c4:85:08:3b:a9:16",
          "mtu": 1500,
          "netmask": "255.255.255.0",
          "network": "192.168.1.0",
          "type": "ether"
      },
```

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.

- extra vars
- task vars (only for the task)
- block vars (only for tasks in block)
- role and include vars
- play vars_files
- play vars_prompt
- play vars
- set_facts
- registered vars
- host facts
- playbook host_vars
- playbook group_vars
- **Inventory host_vars**
- **inventory group_vars**
- inventory vars
- 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: httpd package is present
  yum:
    name: httpd
    state: latest

- name: 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.

If a package gets installed or updated, notify a service restart task that it needs to run.

Example Handler Task in a Play

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

- name: 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.

Playbook Example

```
---
- name: install and start apache
  hosts: web
  become: yes
  vars:
    http_port: 80

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

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

  - name: start httpd
    service:
      name: httpd
      state: started
```

Human-Meaningful Naming

```
---
- name: install and start apache
  hosts: web
  become: yes
  vars:
    http_port: 80

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

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

  - name: httpd is started
    service:
      name: httpd
      state: started
```

Host Selector

```
---
- name: install and start apache
  hosts: web
  become: yes
  vars:
    http_port: 80

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

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

  - name: httpd is started
    service:
      name: httpd
      state: started
```

Privilege Escalation

```
---
- name: install and start apache
  hosts: web
  become: yes
  vars:
    http_port: 80

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

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

  - name: httpd is started
    service:
      name: httpd
      state: started
```

Play Variables

```
---
- name: install and start apache
  hosts: web
  become: yes
  vars:
    http_port: 80

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

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

  - name: httpd is started
    service:
      name: httpd
      state: started
```

Tasks

```
---
- name: install and start apache
  hosts: web
  become: yes
  vars:
    http_port: 80

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

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

  - name: httpd is started
    service:
      name: httpd
      state: started
```

Roles

Roles are a 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

Project with Embedded Roles Example

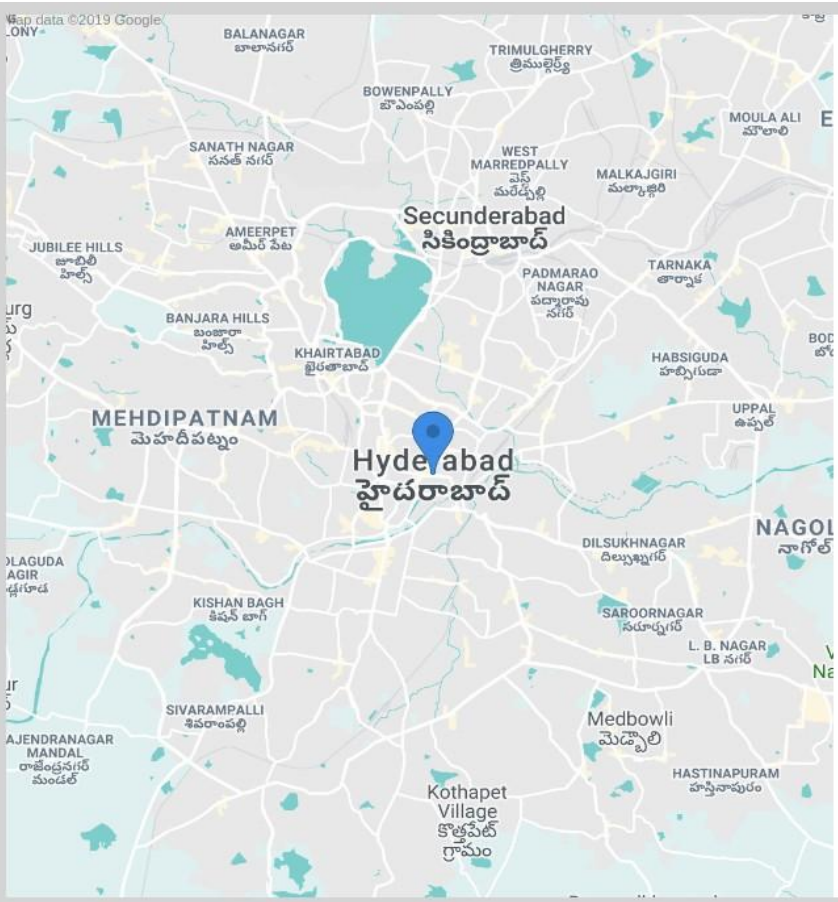
```
site.yml
roles/
    common/
      files/
      templates/
      tasks/
      handlers/
      vars/
      defaults/
      meta/
    apache/
      files/
      templates/
      tasks/
      handlers/
      vars/
      defaults/
      meta/
```

Project with Embedded Roles Example

```
# site.yml
---
- hosts: web
  roles:
     - common
     - apache
```

FAQ

1. Can i create templates for my config files?
2. Can we write variables?
3. Can i notify any other tasks when current task is completed?
4. Can we debug our playbooks?
5. My inventory is not Static, How to deal with it?
6. Can we do parallelism?
7. Can manage my servers with Zero downtime?
8. Can i delegate a task to only specific host?
9. How can we integrate with jenkins pipeline?
10. How to use Ansible to test my application?
11. Can we use ansible as load balancer or autoscaling ?
12. How many modules will be supportive of version patches or releases?
13. What daemon will this ansible run on which i need to do monitoring if anyone have access to server?
14. I want to see few of the best practices of storing playbooks to increase re-usability?
15. Suppose I want to configure some “xyz”, how should be my approach to find the particular modules for that particular “xyz” . What should be my right approach?



Contact Us

Ansible-Hyderabad, Hyderabad, Telangana, India

- ansible.com/community
- @YoursJonathan
- www.meetup.com/ansible-hyderabad
- www.linkedin.com/in/rameshgodishela

Q&A

Thank
You!