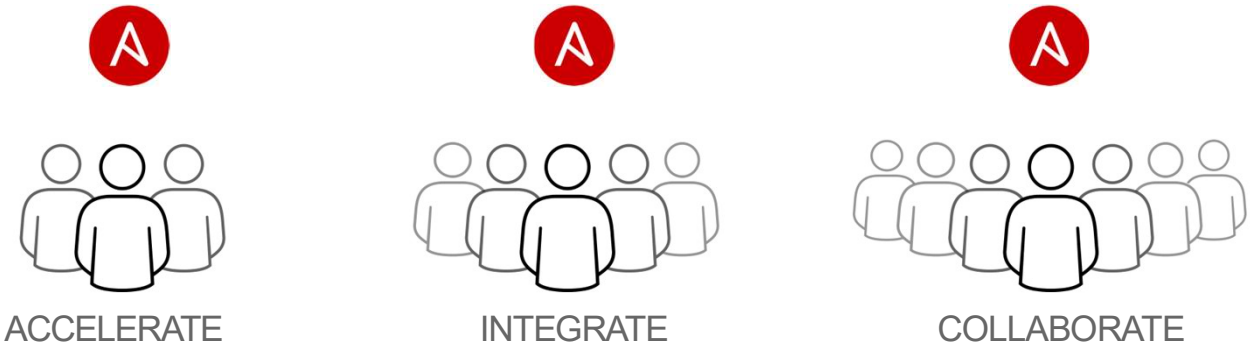
Ansible

An Introduction to Configuration Management

Ansible

- Open source tool, created
- Configuration Management + Application Deployment + Provisioning + Orchestration.
- Written in Python.
- Competes with Puppet, Chef, Salt Stack

Ansible Automation

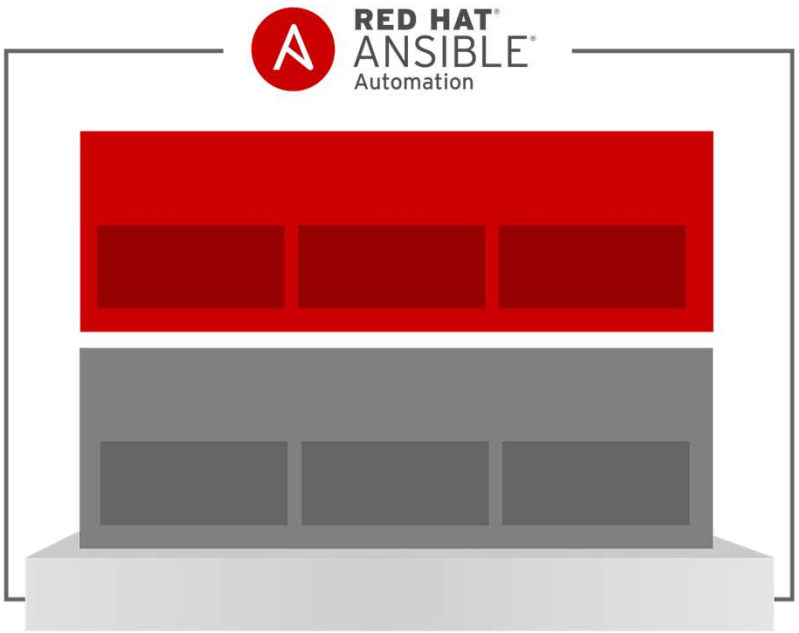


Ansible Automation

- Ansible Automation is the enterprise **framework** for automating across IT operations.

- Ansible Engine runs Ansible Playbooks, the automation **language** that can perfectly describes an IT application infrastructure.

- Ansible Tower allows you **scale** IT automation, manage complex deployments and speed productivity.



Ansible Automation

SIMPLE

Human readable automation

No special coding skills

needed Tasks executed in

order Usable by every

team

Get productive quickly

App deployment

Configuration

management Workflow

orchestration Network

automation

Orchestrate the app

lifecycle

Agentless architecture

Uses OpenSSH & WinRM

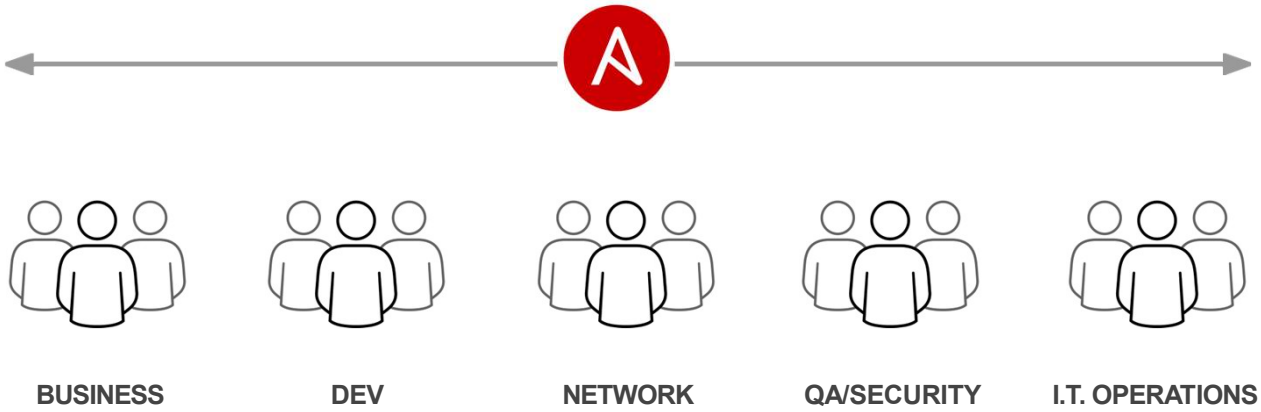
No agents to exploit or

update Get started

immediately

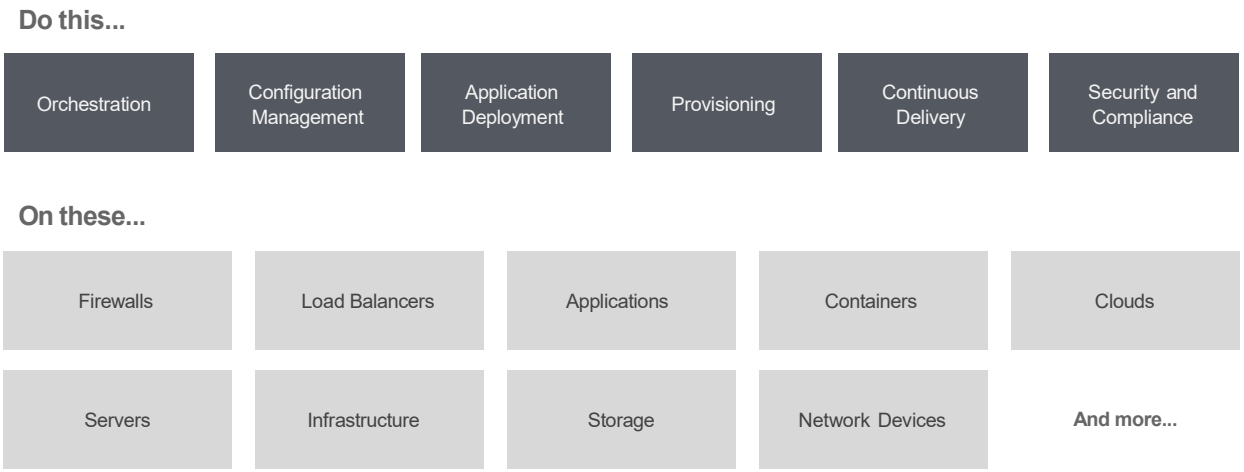
More efficient & more secure

Ansible Automation



What can be done?

Automate the deployment and management of your entire IT footprint.



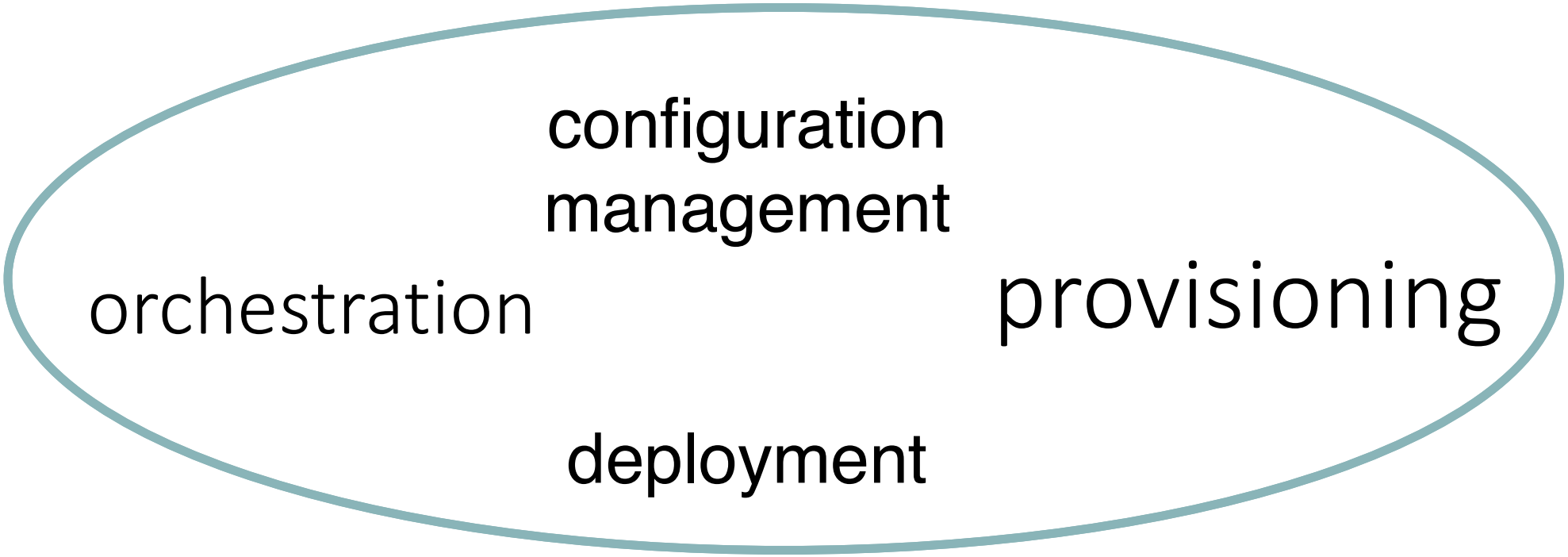
ANSIBLE – FAST

- Minimal Setup
- Manage 5 or 5000 nodes
- Short learning curve - It is easy to learn

ANSIBLE – CLEAR

- Developers
- System Administrators
- IT Management

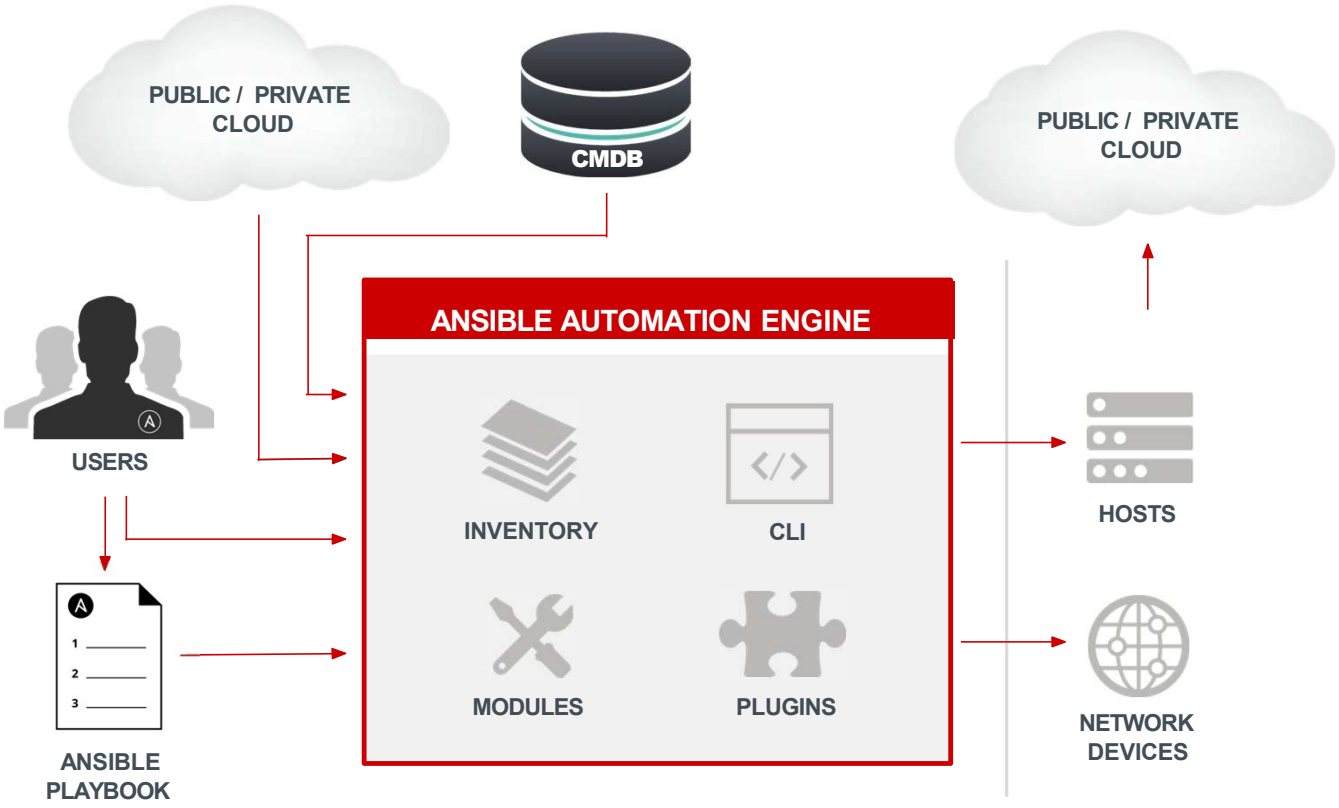
ANSIBLE – COMPLETE

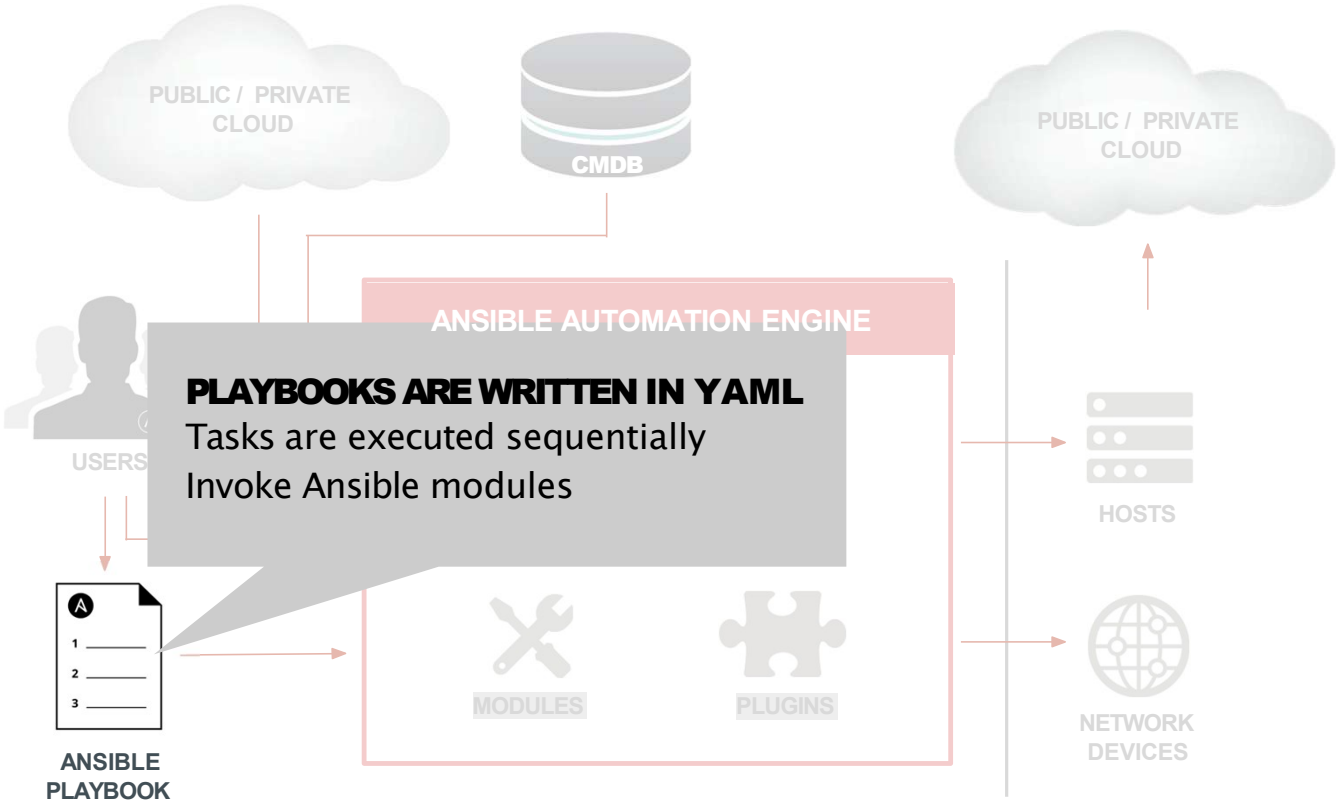


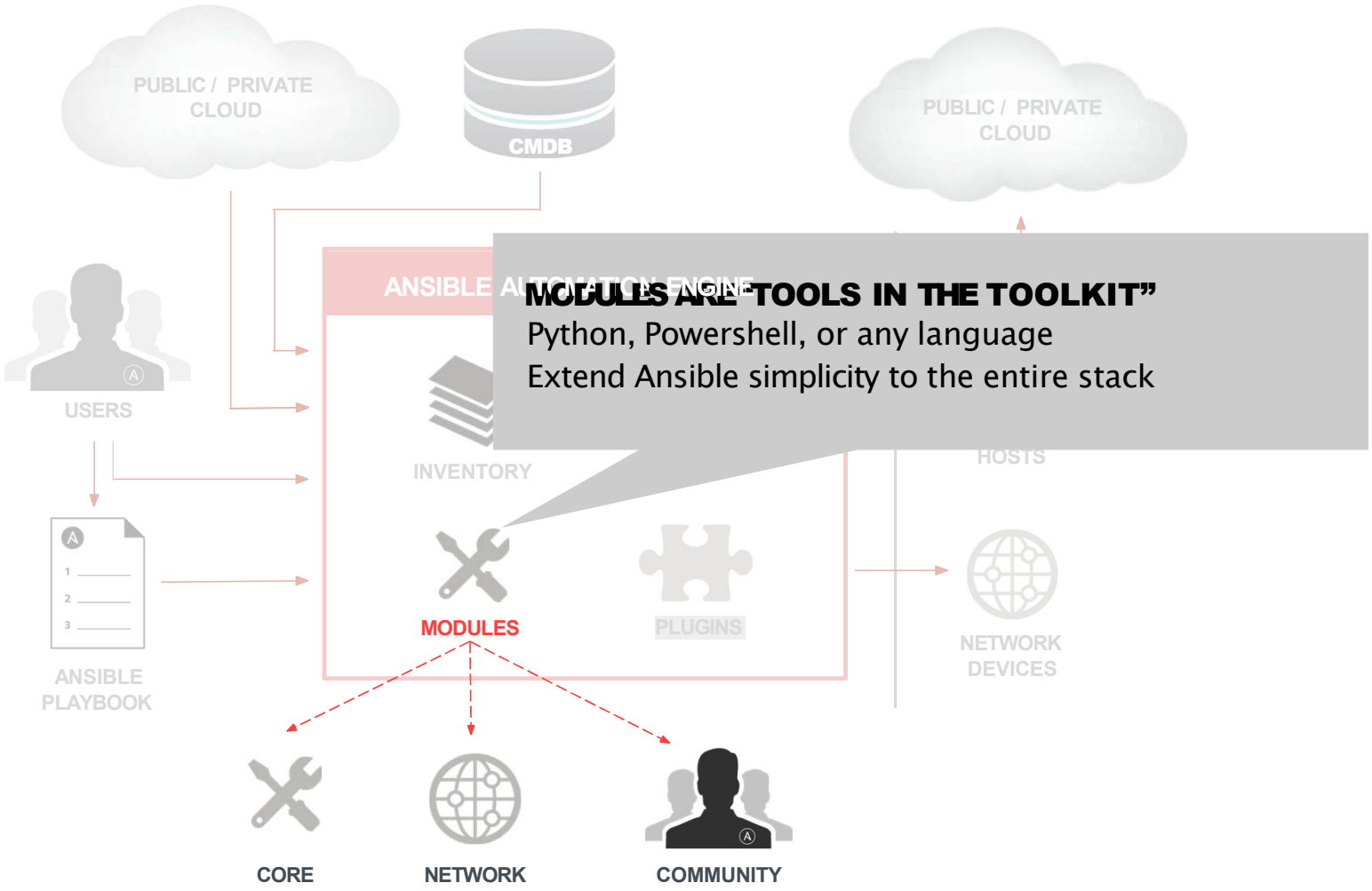
Ansible - Secure

- Go Agentless!
- SSH transport
- No additional firewall rules
- No additional open ports
- Use your own user
- You can sudo

Dynamic Provisioning


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

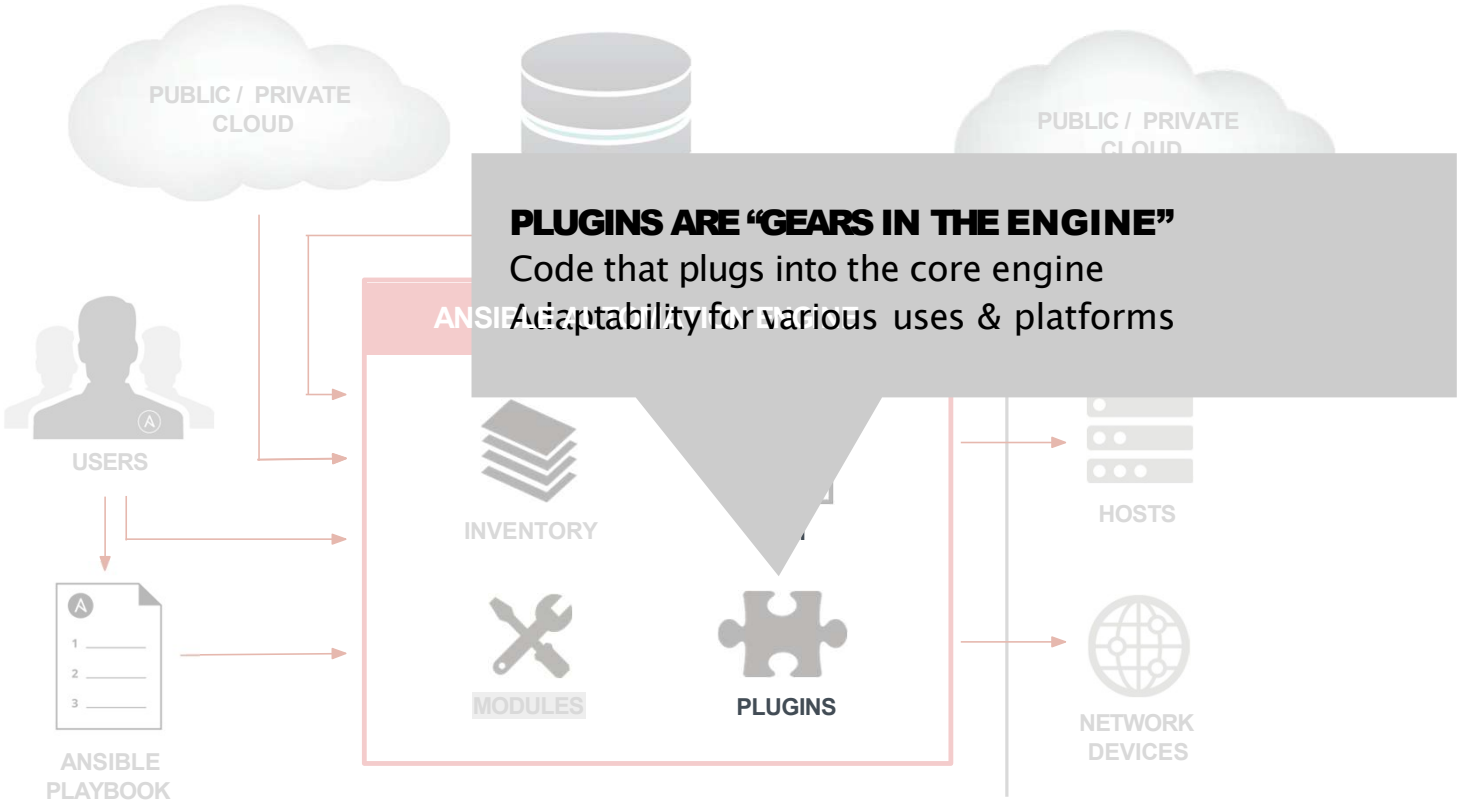

PUBLIC / PRIVATE CLOUD
CMDB
PUBLIC / PRIVATE CLOUD
ANSIBLE AUTOMATION ENGINE
USERS
PLAYBOOKS ARE WRITTEN IN YAML
Tasks are executed sequentially
Invoke Ansible modules
HOSTS
MODULES
PLUGINS
NETWORK DEVICES
ANSIBLE PLAYBOOK



MODULES ARE "TOOLS IN THE TOOLKIT"

Python, Powershell, or any language

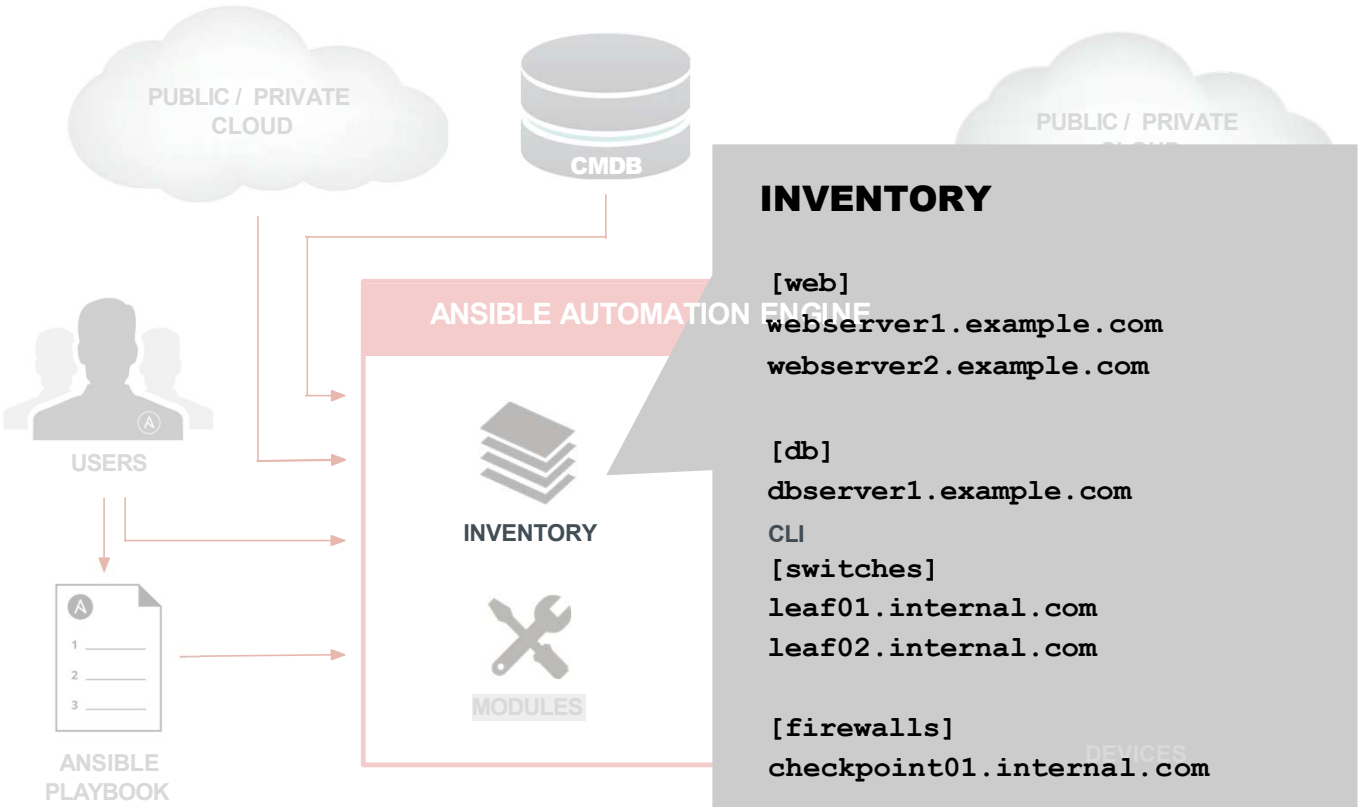
Extend Ansible simplicity to the entire stack

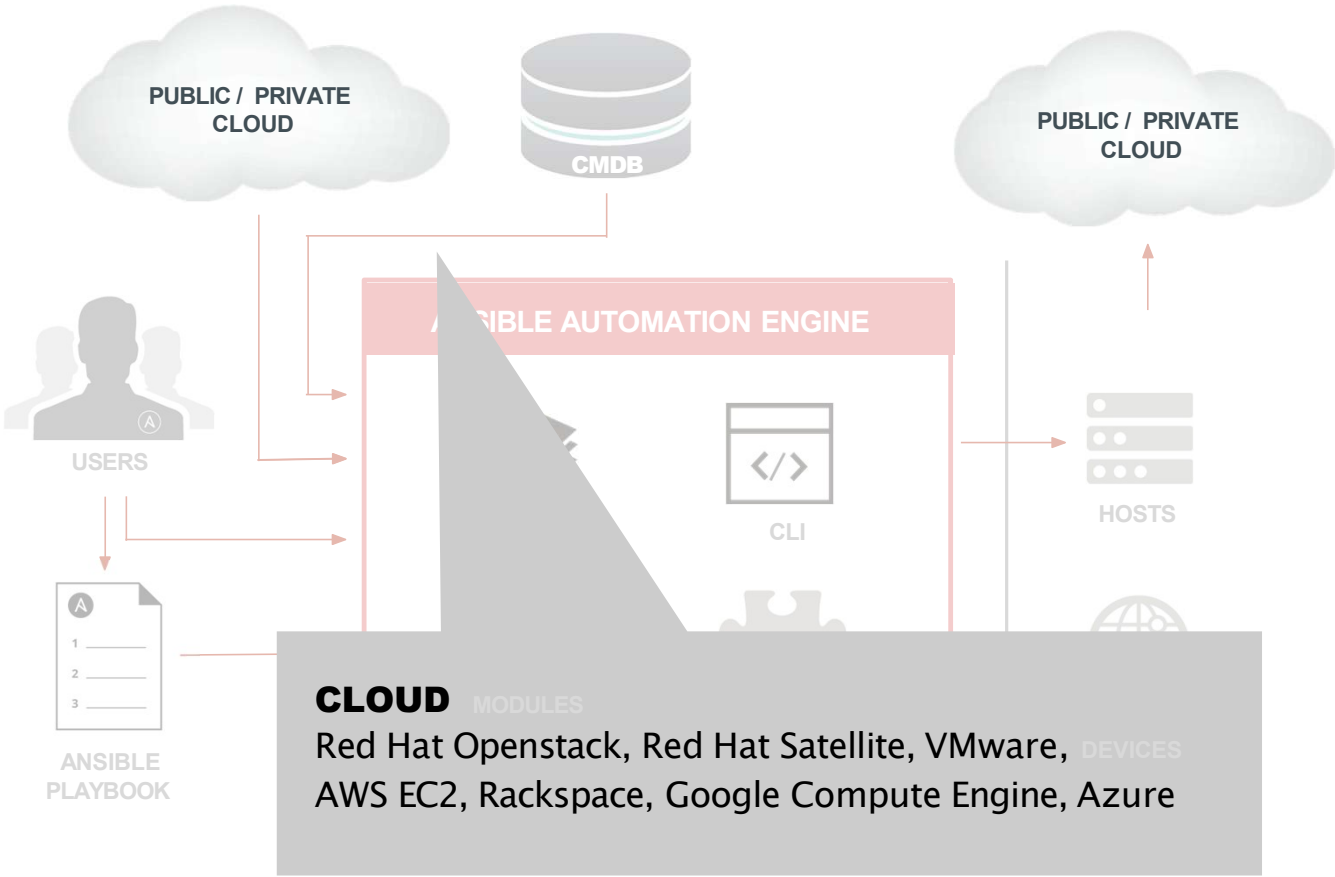


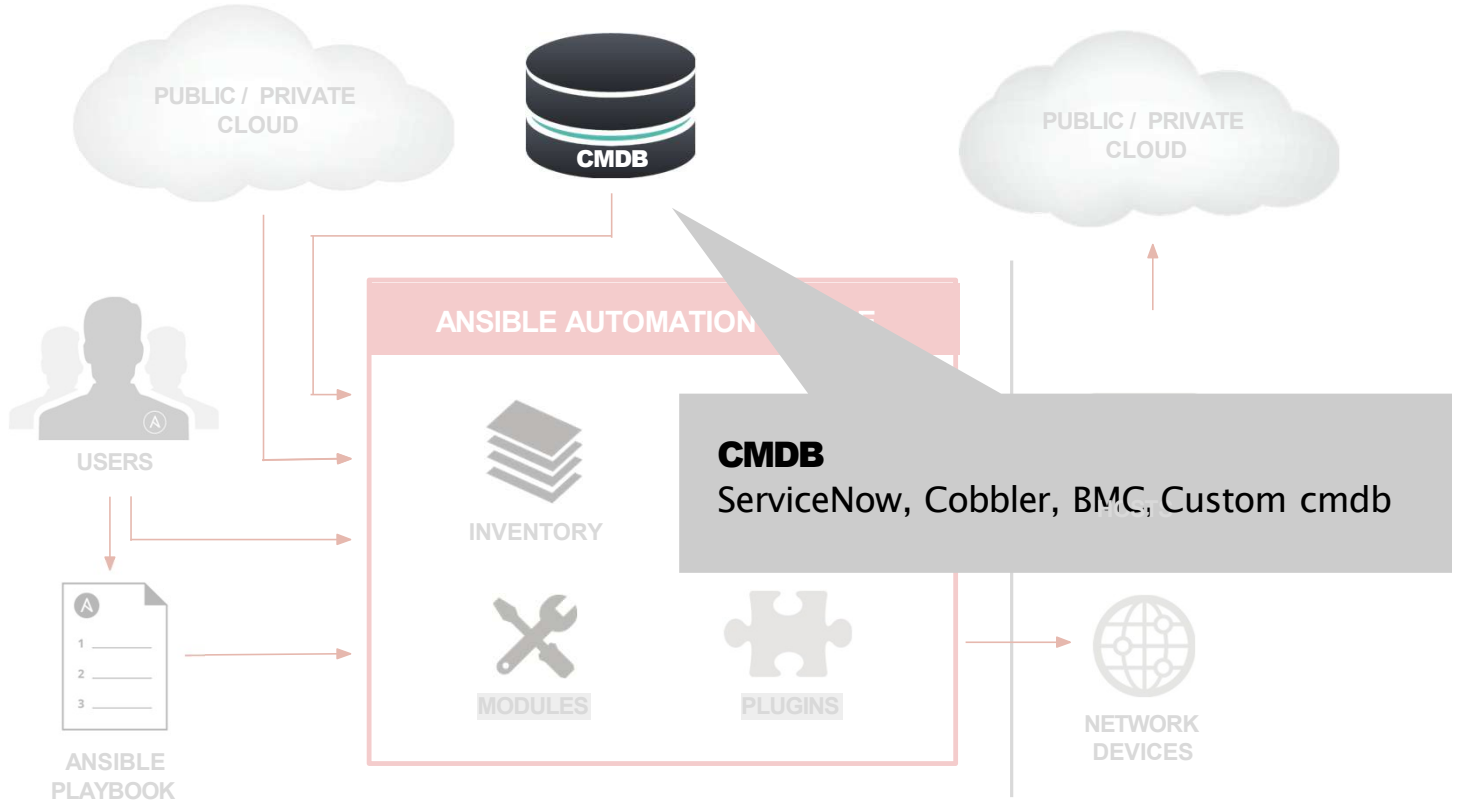
PLUGINS ARE "GEARS IN THE ENGINE"

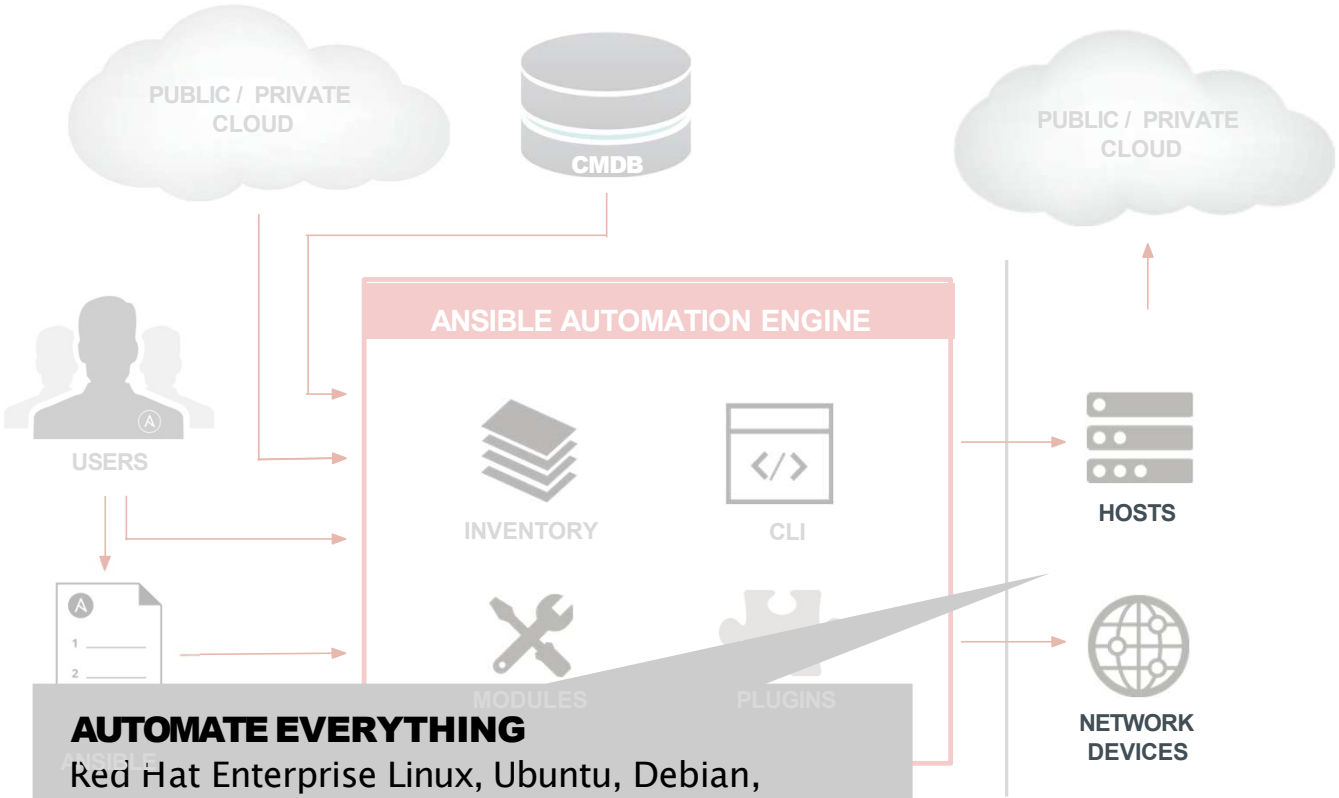
Code that plugs into the core engine

Adaptability for various uses & platforms


PUBLIC / PRIVATE CLOUD
CMDB
PUBLIC / PRIVATE
ANSIBLE AUTOMATION ENGINE
USERS
INVENTORY
MODULES
CLI
ANSIBLE PLAYBOOK
DEVICES
INVENTORY
[web]
webserver1.example.com
webserver2.example.com
[db]
dbserver1.example.com
[switches]
leaf01.internal.com
leaf02.internal.com
[firewalls]
checkpoint01.internal.com
[lb]
f5-01.internal.com


PUBLIC / PRIVATE CLOUD
CMDB
PUBLIC / PRIVATE CLOUD
USERS
CLI
HOSTS
ANSIBLE PLAYBOOK
MODULES
DEVICES
CLOUD
Red Hat Openstack, Red Hat Satellite, VMware,
AWS EC2, Rackspace, Google Compute Engine, Azure


PUBLIC / PRIVATE CLOUD
CMDB
PUBLIC / PRIVATE CLOUD
ANSIBLE AUTOMATION
USERS
INVENTORY
CMDB
ServiceNow, Cobbler, BMC, Custom cmdb
ANSIBLE PLAYBOOK
MODULES
PLUGINS
NETWORK DEVICES


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

ANSIBLE – KEY COMPONENTS

- Inventory
- Modules/Tasks
- Ad-Hocs
- Plays
- Playbooks

INVENTORY

- Hosts and Groups
- Ports and Addresses
- Remote / Sudo usernames

- `web1.example.com ansible_ssh_port=5555 ansible_ssh_host=192.168.1.50`

Inventory Name

SSH port

Connection address

Modules

- Bits of code copied to the target system.
- Modules avoid changes to the system unless a change needs to be made.
- You can write your own modules.

Modules

- apt/yum
- copy
- ec2
- file
- service
- git
- User
- 200 +

Task

- A Task is a declaration about the state of the system

```
- name: install memcached
  yum: name=memcached state=present

- name: Create database user with all database privileges
  mysql_user: name=bob password=12345 priv=*.*:ALL
state=present
```

Inventory

- Defined in /etc/ansible/hosts
- Alternatively use –i <filename>
- You can use patterns (all matches all ☺) to run the commands against
- You can also use Inventory plugins
- All Hosts belong to one or more group
  - Either ungrouped or grouped
  - Always belongs to all
- Hosts can belong to multiple groups
- You can nest groups using the Children prefix

Inventory

```
mail.sample.com

[webservers]
ansible-node1
ansible-node2

[others]
ansible-node3
```

Inventory

```
[webservers]
www[01:50].example.com
```

Ansible Playbooks

```
---
- 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
```

Ansible Playbooks

- Can be chained e.g.
  - *include: playbook-one.yml*
  - *include: playbook-two.yml*
- Can include conditional runs
- See
  - Dummy groups
  - Gather_facts
  - Register
  - add_host
  - Assert
  - Dynamic groups

ANSIBLE SECRETS

- 3 Solutions
  - GPG
  - Ansible Vault
  - HashiCorp Vault
- GPG is the easiest
- Install GPG
- Encrypt a file using the key
- In the step, use Command to decrypt the file
- Include the decrypted file and use as variables. (use /etc/ansible/group_vars for global variables)

ANSIBLE SECRETS

- ANSIBLE VAULT
- Only Protects data at rest
- Use no_log true as best practice
- Usage:
  - ansible-vault encrypt|decrypt|view|edit <file>
  - Ansible-playbook --ask-vault-password –i<inv files> <nameofplaybook> # will ask for vault password, use ––vault-password-file as a workaround

ANSIBLE SECRETS

- HASHICORP VAULT
- Install Vault from Hashicorp
- Create a config.hcl file
- Start the server (vault server –-config=config.hcl)
- Export VAULT_ADDDR (Default port is 8200)
- Initialize vault – vault operator init
- Unseal the vault – will need 3 of 5 keys (login first)
- Enable Secrets – vault secrets enable –-path=ansible kv
- Put a secret – vault kv put <namespace> <var=value>
- Get a secret – vault kv get

ANSIBLE SECRETS

- Use the secret with a lookup
- Token and URL can be passed as Environment variables

```
msg: "{{ lookup('hashi_vault', 'ansible/demo
token=<token> url=http://127.0.0.1:8200') }}"
```

Ansible+Docker

- docker module
- docker_images module
- docker_facts module
- Docker inventory plugin
- Uses docker-py Docker client python library
- Run/Verify/build/deploy docker images

Other Topics

- Ansible Tower
- Ansible Galaxy – Roles and Collections that can be used in your playbook

Managing Multiple Environments

- Strategy #1: Group Variables
  - Group by
    - Function (DB/Web Servers/Node)
    - Stage (Prod/Dev)
    - Region
    - Business Functionality…
  - An asset may belong to 1 group per category.
  - Issue: A host may belong to more than 1 group and each group can have its own value. No Way to set precedence.
  - Loaded alphabetically and last loaded one wins.

Managing Multiple Environments

- Strategy #2: Group Children and Hierarchy
  - Syntax [groupname: children]
  - Children override the parents
  - E.g. Environment is a group that has some base vars
  - Dev/qa/prod will be children of environment and each will override the base, if set. If not, base values will be used.
  - Issue: Host may still belong to multiple categories e.g. dev in NY. In such cases, we establish a hierarchy (highest to lowest) e.g.
    - DEV
      - REGION
        - FUNCTION

Managing Multiple Environments

- E.g
  - function
    - [function: chidren]
    - Web
    - Database
    - Loadbalancer
    - Region # region is a child of function
    - [region:children]
    - Myc
    - Webnode
    - Environments
    - [environments:children] #environment is a child of region
    - Dev
    - Stage
    - prod

Managing Multiple Environments

- Strategy #3: Allow Explicit Loading Order
- Ansible allows
  - Var_files #for plays
  - Include_files #can be used in tasks
- Common variables are set in group_vars file
- Example of var files will be
- File: vars/dev.yml
  - Server_memory: 512mb
- File: vars/prod.yml
  - Server_memory: 1024mb

Managing Multiple Environments

- A play book can then use it like this:

- vars_files:
- – “vars/{{ env }}.yml”
- – “vars/{{ function }}.yml” # the ordering of this will change the value..

- NB: The group_vars file for dev (group_vars/dev) will contain
  - env: dev

Managing Multiple Environments

```
.
├── ansible.cfg
├── environments/          # Parent directory for our environment-specific directories
│   │
│   ├── dev/               # Contains all files specific to the dev environment
│   │   ├── group_vars/    # dev specific group_vars files
│   │   │   ├── all
│   │   │   ├── db
│   │   │   └── web
│   │   └── hosts          # Contains only the hosts in the dev environment
│   │
│   ├── prod/              # Contains all files specific to the prod environment
│   │   ├── group_vars/    # prod specific group_vars files
│   │   │   ├── all
│   │   │   ├── db
│   │   │   └── web
│   │   └── hosts          # Contains only the hosts in the prod environment
│   │
```

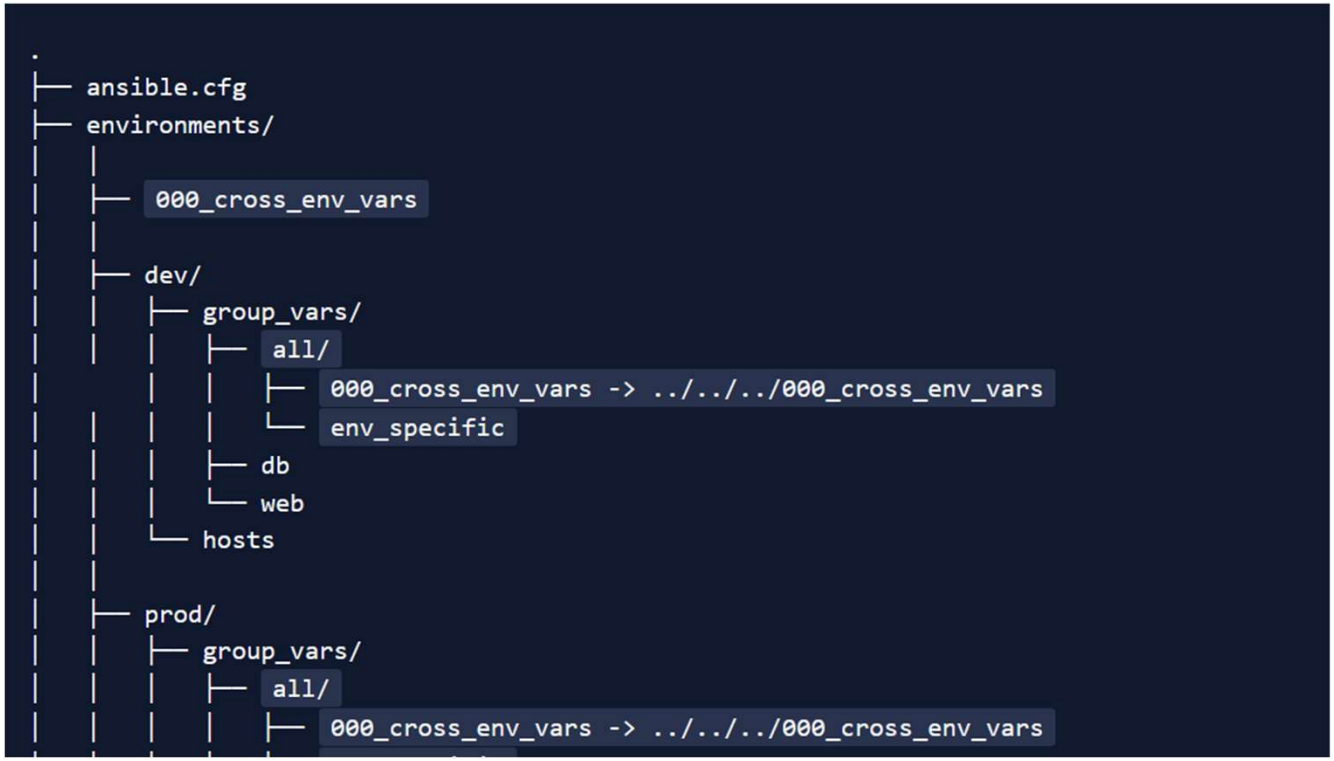
ram

Managing Multiple Environments

- There are some duplication in above e.g. web and db in each stage.
- Actually this makes sense so that changes can be done in one env. Tested and then moved to next stage.
- Cannot share variables across environments.
- A workaround:
  - Replace the all file with an all directory
  - Create a var file inside the directory and create a symbolic link
- Code sample

```
cd dev/group_vars
mv all env_specific
mkdir all
mv env_specific all
cd all
ln –s ../../../00_cross_env_vars
```

Managing Multiple Environments



Managing Multiple Environments

- In ansible.cfg,

  [defaults]

  inventory = ./environments/dev

- Now a command like: ansible –m ping will select all hosts for dev. (note: no more –i to be typed in ☺

Idempotent Playbooks

- Most Ansible plugins provide idempotency not all
- Consider the below

```
- name: Create fstab entry
  lineinfile:
    path: /etc/fstab
    line: '{{ nfs_dir_server_ip }}:{{ nfs_dir_mnt_path }}  {{ nfs_dir_mnt_path }} nfs
defaults,soft,bg,noauto,rsize=32768,wsize=32768,noatime  0 0'
    state: present
```

- The above will add a line to an existing file

Idempotent Playbooks

- We do not want the line to be added every time.
- To avoid this, we add a Regexp to add only one instance)

```
- name: Create fstab entry
  lineinfile:
    path: /etc/fstab
    line: '{{ nfs_dir_server_ip }}:{{ nfs_dir_mnt_path }}  {{ nfs_dir_mnt_path }} nfs
defaults,soft,bg,noauto,rsize=32768,wsize=32768,noatime  0 0'
   regex: 'nfs defaults,soft,bg,noauto'
    state: present
```

- The above will add a line to an existing file (Only if its's not present)

Demo Time

Ansible Ad-hoc Commands, Modules, Deployments

Q &A

We're done!

Thank you for your time and participation.