ANSIBLE

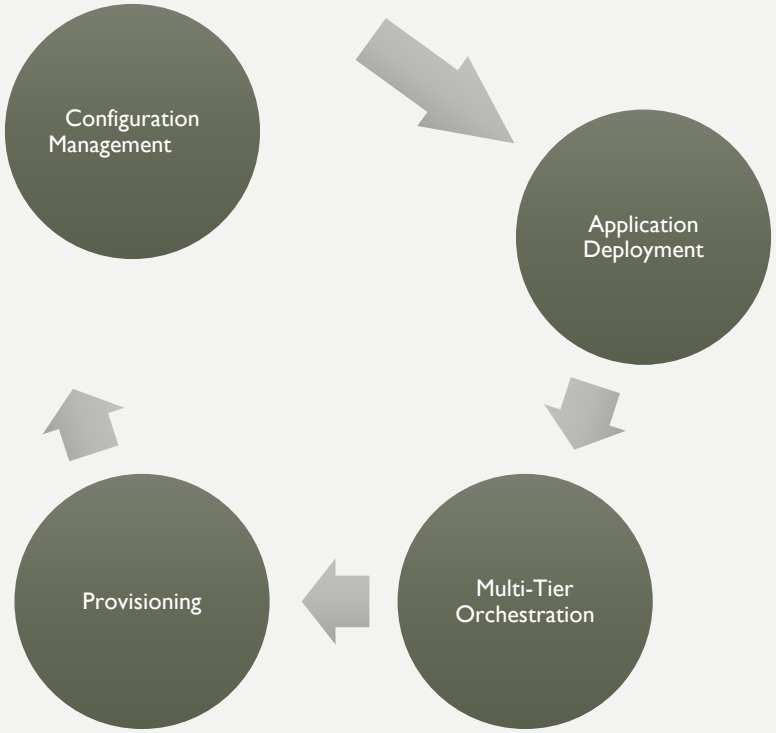
ANSIBLE

AGENDA

- Introduction
- Configuration Management & Orchestration
- Environment Setup
- Installation
- Foundation
  - Inventory
  - Host
  - Tasks
  - Plays
  - Ad Hoc Commands
- Playbooks
  - Introduction
  - Packages (apt, become, with_items)
  - Services
- Roles
  - Overview
  - Converting to Roles: tasks, handlers
- Advanced Concepts
  - Advanced Execution Introduction
  - Limiting Execution by Hosts: limit
  - Removing Unnecessary Steps: gather_facts
  - Extracting Repetitive Tasks: cache_valid_time
  - Limiting Execution by Tasks: tags
  - Idempotence: changed_when, failed_when
- Troubleshooting, Testing & Validation

INTRODUCTION

Complete Package

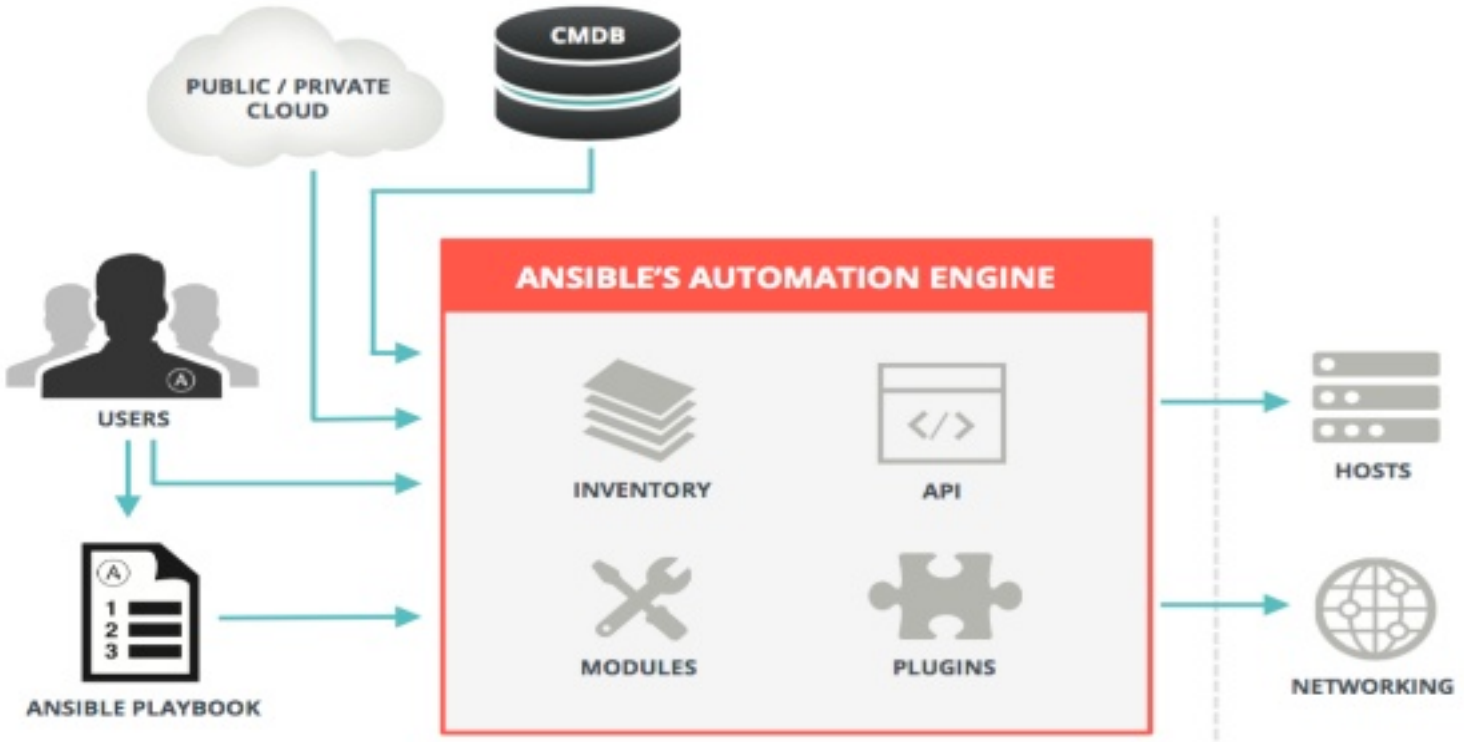


INSTALLATION

- pip install ansible
- sudo yum install ansible (centos, rhel)
- sudo apt-get install ansible (debian)

- How

HOW ANSIBLE WORKS



MODULES RUN COMMANDS

AD HOC COMMANDS DEMONSTRATION

- kaushiki-2:~ arun$ docker run -it --name ansible williamyeh/ansible:ubuntu16.04 /bin/bash
- root@23c06c085cbe:/# ansible all -i 'localhost,' -c local -m ping

```
localhost | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

- root@23c06c085cbe:/# ssh-keygen -t rsa

  Generating public/private rsa key pair.

  Enter file in which to save the key (/root/.ssh/id_rsa):

  Created directory '/root/.ssh'.

  Enter passphrase (empty for no passphrase):

  Enter same passphrase again:

  Your identification has been saved in /root/.ssh/id_rsa.

  Your public key has been saved in /root/.ssh/id_rsa.pub.

  The key fingerprint is:

  SHA256:G6vmWX17AzpdpGSrEoiTnYi3gekiUy4rKxNiJaqx1OA root@23c06c085cbe

- kaushiki-2:~ arun$ vi /tmp/ansible_id_rsa.pub
- kaushiki-2:~ arun$ cat /tmp/ansible_id_rsa.pub >> ~/.ssh/authorized_keys
- root@23c06c085cbe:/# ansible all -i '192.168.26.36,' -c local -m ping

```
192.168.26.36 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

- root@23c06c085cbe:/# ssh arun@192.168.26.36 -C "echo hello world"

  hello world

USE CASES

- Setup Full Web Stack Using ANSIBLE
  - Includes Tomcat
  - Includes JRE
  - Includes WAR deployment
- Setup Oracle 12 c database
- Setup Link or network between Web Stack and Database
- Setup ELK Stack Using Ansible
- Setup Jenkins Using Ansible

FILE AND DIRECTORY MANAGEMENT

- In addition to basic **create**, **remove**, **update**, and **delete** (**CRUD**) operations, we can also set permissions, change owners, set group owners, operate on recursive folder trees, and more.

```
# Create a directory using an Ansible Task

- name: Creates a directory
file: path=/opt/helloWorld state=directory

# Create a directory using an Ansible Task,
# which is owned by the baseballplayersgroup

- name: Creates a directory
file: path=/opt/helloWorld state=directory

# Creates a directory owned by the baseballplayers group
# with CHMO 0775 permissions
- name: Creates directory
file: path=/opt/helloWorld state=directory owner=baseballplayers group=baseballplayers mode=0775

# Changes the ownership of myconfiguration.conf to
# bob and changes permissions to 0644
- name:
file:
path: /opt/myconfiguration.conf
owner: bob
group: admin
mode: 0644
```

MANAGING USERS

```
# Create a User 'dortiz'
---
- hosts: all

tasks:
- name: Add David Ortiz User to the System
user:
name: dortiz
comment: "David Ortiz has entered the building"

# Create a User 'jdaemon' and add to group baseballplayers
---
- hosts: all

tasks:
- name: Add Johnny Daemon User to the System
user:
name: jdaemon
comment: "Johnny Daemon has entered the building"
groups: baseballplayers
```

MANAGING SERVICES

```
# Example action to start service httpd, if not running
- service:
name: httpd
state: started

# Example action to stop service httpd, if running
- service:
name: httpd
state: stopped

# Example action to restart service httpd, in all cases
- service:
name: httpd
state: restarted

# Example action to reload service httpd, in all cases
- service:
name: httpd
state: reloaded

# Example action to enable service httpd, and not touch the running state
- service:
name: httpd
enabled: yes

# Example action to start service foo, based on running process /usr/bin/foo
- service:
name: foo
pattern: /usr/bin/foo
state: started

# Example action to restart network service for interface eth0
- service:
name: network
state: restarted
args: eth0
```

TRANSFERRING FILES

```
# Example from Ansible Playbooks
- copy:
src: /srv/myfiles/foo.conf
dest: /etc/foo.conf
owner: foo
group: foo
mode: 0644

# The same example as above, but using a symbolic mode
# equivalent to 0644
- copy:
src: /srv/myfiles/foo.conf
dest: /etc/foo.conf
owner: foo
group: foo
mode: "u=rw,g=r,o=r"

# Another symbolic mode example, adding some
permissions
# and removing others
- copy:
src: /srv/myfiles/foo.conf
dest: /etc/foo.conf
owner: foo
group: foo
mode: "u+rw,g-wx,o-rwx"

# Copy a new "ntp.conf file into place, backing up the
# original if it differs from the copied version
- copy:
src: /mine/ntp.conf
dest: /etc/ntp.conf
owner: root
group: root
mode: 0644
backup: yes

# Copy a new "sudoers" file into place, after passing
# validation with visudo
- copy:
src: /mine/sudoers
dest: /etc/sudoers
validate: 'visudo -cf %s'
```

BEYOND FUNDAMENTALS

- Playbook And Conditional Logic
- Loops and Iterators
- Includes
- Roles
- Registers
- Error Trapping
- Ansible Handlers

PLAYBOOK AND CONDITIONAL LOGIC

```
# Reboot Debian Flavored Linux Systems using the WHEN operator tasks:
- name: "Reboot all Debian flavored Linux systems"
  command: /sbin/reboot -t now
  when: ansible_os_family == "Debian"
```

```
# Display Hello World to DevOps readers
vars:
  is_enabled: true
tasks:
 - name: "Tell only DevOps People Hello"
   shell: echo “Hello DevOps Readers“
   when: is_enabled
```

```
tasks:

- shell: echo "The operator has not been set"
  when: myvar is undefined
```

```
# Iterator with conditional logic to stop the iteration at a specified number

tasks:
- command: echo {{ item }}
with_items: [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ]
when: item > 7
```

```
# Conditional Logic directly in the Ansible Roles

- hosts: all
roles:
- { role: centos_config, when: ansible_os_family == 'CentOS' }
```

ITERATORS AND LOOPS

- Basic Looping Using **with_items**

 # playbook.yml without list based iterators

```
---
- hosts: all
tasks:
- name: Install Apache2
apt: name=apache2 state=installed

- name: Install VIM
apt: name=vim state=installed

- name: Install TMUX
apt: name=tmux state=installed

- name: Install MOSH
apt: name=mosh state=installed
```

 # playbook.yml using an Iterator to install packages

```
---
- hosts: all
tasks:
- name: Install list of packages
apt: name={{item}} state=installed
with_items:
- apache2
- vim
- tmux
- mosh
```

NESTED LOOPS USING WITH_NESTED

```
# Demo of Nested Loops Using Ansible. To execute use the following command:
# > Ansible-playbook -i 'localhost,' -c local nested_loops.yml

---
- name: Demo of nested loops using with_nested
hosts: all
remote_user: root
vars:
listA: [1, 2]
listB: [a, b]
tasks:
- name: Say Hello using Nested Loops
debug: msg="The values in the array are {{item[0]}} and {{item[1]}}"
with_nested:
- listA
- listB
```

LOOPING OVER HASHES USING WITH_DICT

```
- name: Say Hello to our Favorite Looney Tune Characters
hosts: all
vars:
looney_tunes_characters:
bugs:
full_name: Bugs A Bunny
daffy:
full_name: Daffy E Duck
wiley:
full_name:Wiley E Coyote
tasks:
- name: Show Our Favorite Looney Tunes
debug:
msg: "Hello there: {{ item.key }} your real name is {{ item.value.full_name }}"
with_dict: "{{ looney_tunes_charachters }}"
```

ITERATING OVER FILES USING WITH_FILE

hello.txt

```
Hello There:
```

favourite_toons.txt

```
Bugs Bunny
Daffy Duck
Mickey Mouse
Donald Duck
Wiley E. Coyote
```

```
# Example Playbook which Iterates Over the Contents of Two Files (iterator_file_contents.yml)
---
- name: Say hello to our favorite Looney Toons
hosts: all
tasks:
- name: Say Hello to Our Favorite Looney Toons
debug:
msg: "{{ item }}"
with_file:
- hello.txt
- favorite_toons.txt
```

ITERATING OVER SEQUENTIAL NUMBERS

```
# Ansible Example provided by Ansible.com
# create some test users
- user:
name: "{{ item }}"
state: present
groups: "evens"
with_sequence: start=0 end=32 format=testuser%02x
```

THE DO UNTIL OPERATOR

```
- action:
        /usr/bin/tail -n 1 /var/log/auth.log
  register: result
  until: result.stdout.find("Cannot create session") != -1
  retries: 100
  delay: 1
```

INCLUDE/IMPORT : PLAY LEVEL

- This provides an easy way to embed **Play** from another **YAML** files
- Include has been deprecated and now one should use **import_playbook**

```
---
  - import_playbook: play.yml
  - name: Including play.yml
    hosts: all
    tasks:
     - debug: msg=hello
```

```
---
  - name: Playing play.yml
    hosts: all
    tasks:
     - debug: msg=hello I am included
```

INCLUDE/IMPORT : TASK LEVEL

- This provides an easy way to embed **Play** from another **YAML** files
- Include has been deprecated and now one should use **import_playbook**

```
---
  - name: Including tasks play1.yml
    hosts: all
    tasks:
     - import_tasks: play1.yml player=Arun
```

```
---
  - name: Playing play1.yml
    hosts: all
    tasks:
     - debug: msg="Player {{player}} is playing"
```

DYNAMIC INCLUDES : TASK LEVEL

- This provides an easy way to embed **Play** from another **YAML** files
- Include has been deprecated and now one should use **import_playbook**

```
---
  - name: Including tasks play1.yml
    hosts: all
    tasks:
     - import_tasks: play1.yml player=Arun
```

```
---
  - name: Playing play1.yml
    hosts: all
    tasks:
     - debug: msg="Player {{player}} is playing"
```

ROLES

- Provide us way to divide out our automation into uniquely defined responsibilities
- Provide Configuration Management modularization

A role is a set of Ansible tasks for configuration management automation grouped by a common purpose or responsibility

ANSIBLE REGISTER VARIABLES

- It provide us the way of capturing the results of a given task and executing a set of additional automations based on the captured results
- It is sort of global variable
- It enables us to store the captured data for later and then conditionalizing future tasks based on the results of previous ones

SIMPLE ANSIBLE REGISTERS

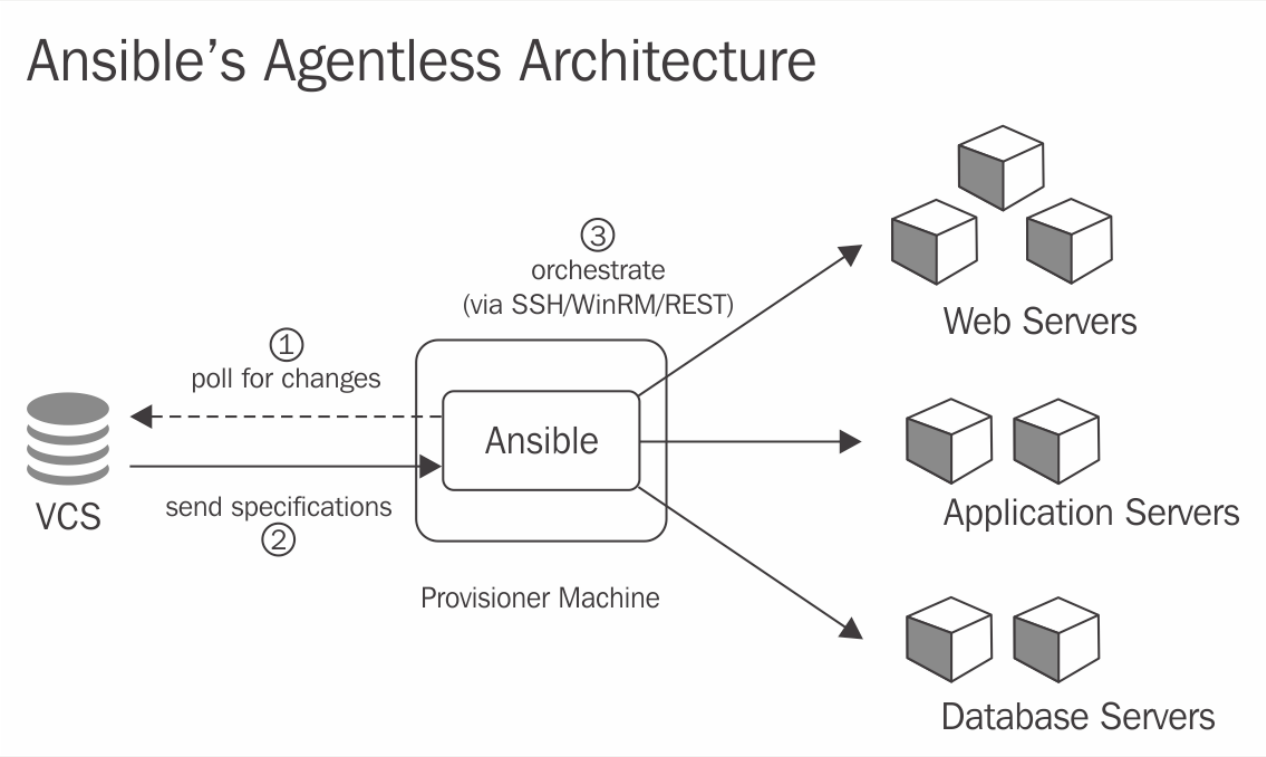
- Most basic Ansible register implementations require us to only register the output of a given operation

```
---
 - name: A simple ansible register example
```

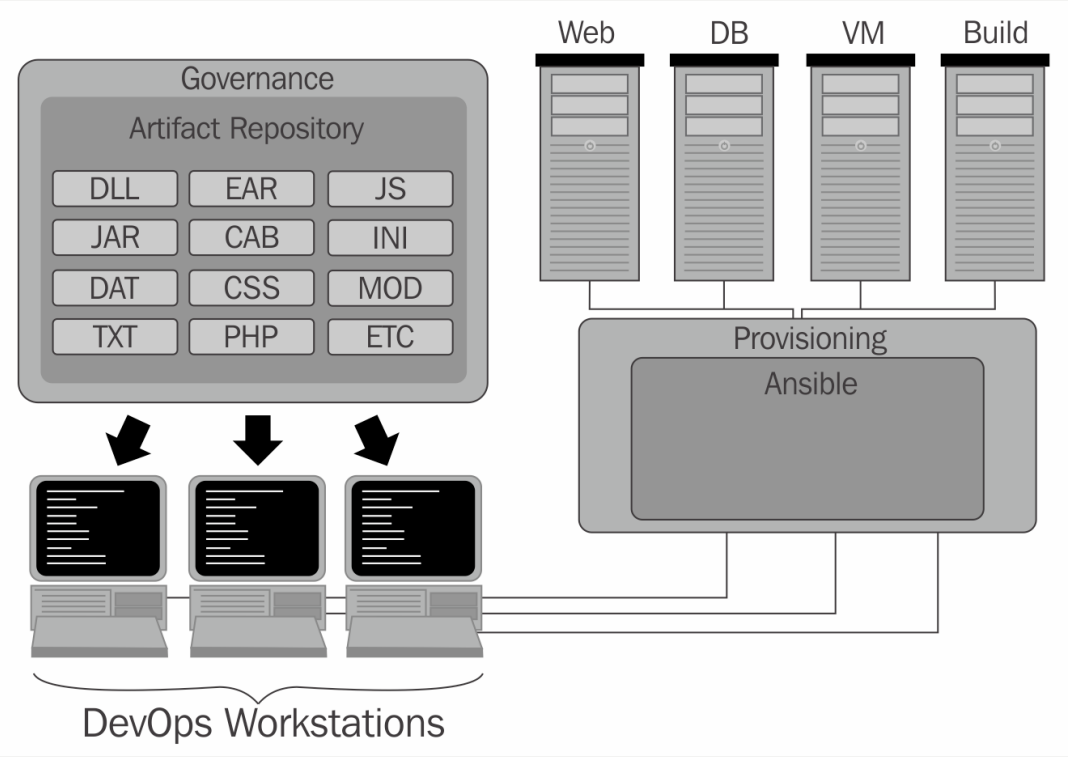
ACCESSING REGISTERS

ITERATING OVER REGISTER CONTENTS

ANSIBLE HANDLERS


Ansible's Agentless Architecture
③
orchestrate
(via SSH/WinRM/REST)
①
poll for changes
Ansible
VCS
send specifications
②
Provisioner Machine
Web Servers
Application Servers
Database Servers

BINARRY ARTIFACT MANAGEMENT AND ANSIBLE



VARIABLES AND FACTS

- Defining Variables in Playbooks
  - Simplest way to define variables is to put a *vars* section in playbook with the names and values of variables

```
vars:
    key_file: /etc/nginx/ssl/nginx.key
    cert_file: /etc/nginx/ssl/nginx.crt
    conf_file:  /etc/nginx/sites-available/default
    server_name: localhost
```

- Ansible allows us to put variables into one or more files, using a section called **vars_files.** The above syntax can be replaced by putting the content in the nginx.yml and replacing the **vars** section with **vars_files** section

```
vars_files:
   - nginx.yml
```

nginx.yml

```
key_file: /etc/nginx/ssl/nginx.key
cert_file: /etc/nginx/ssl/nginx.crt
conf_file:  /etc/nginx/sites-available/default
server_name: localhost
```

- Viewing the values of variables
  - debug: var=myvarname
- Registering Variables
  - Often we require to capture the result of a task into a variable
  - This is done by creating a ***registered variable*** using the register clause when invoking a module

```
PLAY [Show return value of command module]
*********************************************************************************
TASK [Gathering Facts] ********************************************************************************
ok: [vagrant1]

TASK [Capture ouput of id command]
*********************************************************************************
changed: [vagrant1]

TASK [debug] ******************************************************************************************
ok: [vagrant1] => {
    "login": {
        "changed": true,
        "cmd": [
            "id",
            "-un"
        ],
        "delta": "0:00:00.004917",
        "end": "2018-01-15 20:53:57.529687",
        "failed": false,
        "rc": 0,
        "start": "2018-01-15 20:53:57.524770",
        "stderr": "",
        "stderr_lines": [],
        "stdout": "vagrant",
        "stdout_lines": [
            "vagrant"
        ]
    }
}
PLAY RECAP
*********************************************************************************vagrant1                   :
ok=3    changed=1    unreachable=0    failed=0
```