Cybage®
Delivering Value. Scientifically.

ANSIBLE

Authored and Presented by: Amol Shende

Agenda

- Prerequisite- ssh, configuration management
- Need of Ansible
- What is Ansible?
- Architecture
- Key Concepts
- Features
- Cons
- Some Advance Concepts
- Case Studies

SSH (Secure Shell)

- SSH is cryptographic network protocol
- Used for secure data-communication, remote login, remote command execution and other secure network services
- Standard port 22
- Ansible uses SSH for performing its tasks
- Authentication is done using username-password or public-key cryptography algorithm
- Utilities- SCP, SFTP, x-11 forwording

Configuration Management

- What so you need to build good software? (**System, good environment)**
- It is process to setup such system and environment in well documented manner
- when applied to system, provides visibility, performance , maintainability and reduce cost and risk
- It makes sure that system is behaving as intended and documented way to support project life cycle
- Two ways to do 1) Manual 2) Automatic

Cybage® Delivering Value. Scientifically.

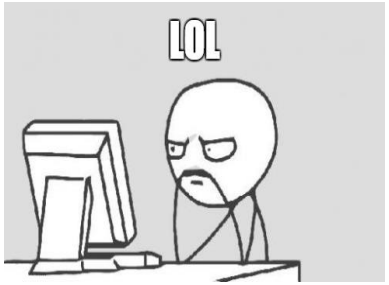
Need Of Ansible (Manual Vs Automatic)

Shell Scripting

1) Long and Tedious scripts
2) Difficult to maintain
3) Run scripts manually one machine-one script at a time
4) Need lots of time
5) Difficult to track
6) Can not handle Hybrid Infrastructure
7) Poor Documentation
8) Need Linux expert
9) No Idempotent

Ansible

1) Short and readable
2) Easy to maintain
3) Can run multiple scripts on 1000s of nodes at time
4) Very less time
5) Easy to track
6) Easy way to handle Hybrid Infrastructure
7) Proper and structured documentation
8) Basic Linux knowledge is sufficient
9) Idempotent



Cybage®
Delivering Value. Scientifically.

What is Ansible?

- Configuration Management Tool
- Developed by **Michael DeHaan**
- Initial Release February 2012

History

- 1st generation
  - CF Engine
- 2nd generation
  - Puppet
  - Chef
- 3rd generation
  - Ansible
  - Salt Stack

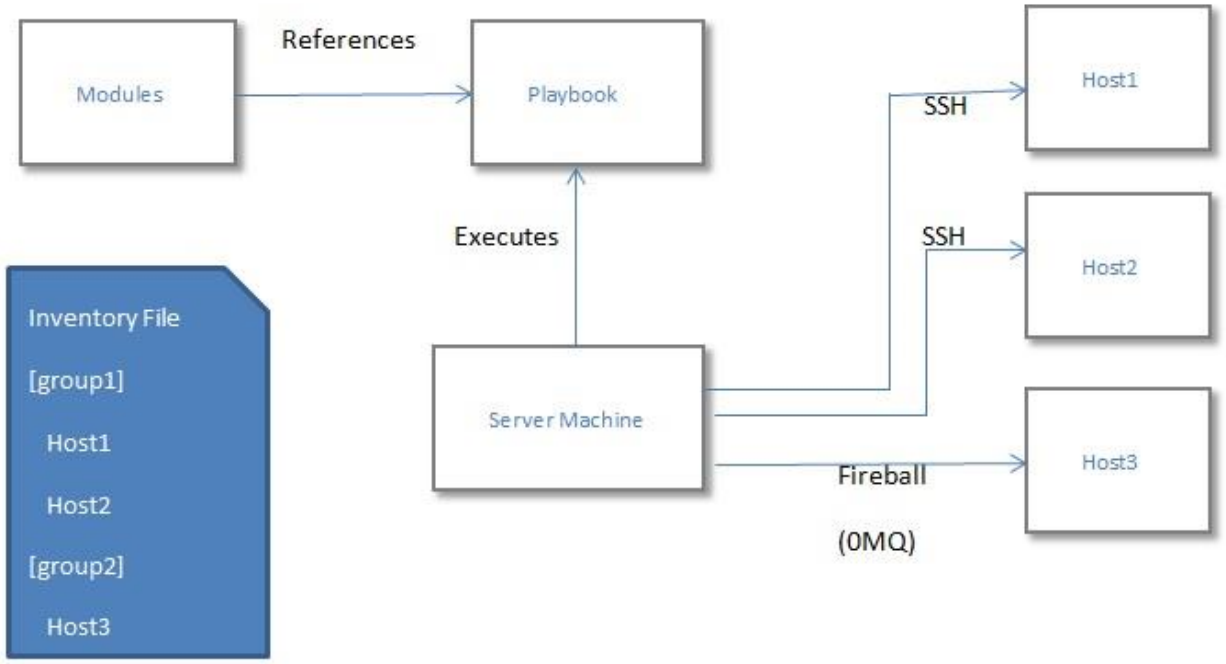
Who Uses?

Cybage®
Delivering Value. Scientifically.

Goals

- Configuration Management
- Application Deployment
- Cloud Provisioning
- AD-HOC Task execution
- Continuous Deployment
- zero downtime rolling updates

Architecture



Cybage®
Delivering Value. Scientifically.

Key Concepts

- Inventory
- Modules
- Ad Hoc Commands
- Playbooks
  - Roles
  - Tasks
  - Variables
  - Templates
  - Handlers

Cybage® Delivering Value. Scientifically.

Inventory

- Define a list of target hosts
- usually located in /etc/ansible/hosts

Inventory

```
[mongo_master]
168.197.1.14

[mongo_slaves]
mongo1.server
mongo2.server
mongo3.server

[www]
168.197.1.2
```

List of target hosts.
Usually located in
/etc/ansible/hosts

Cybage® Delivering Value. Scientifically.

Inventory

- If remote host ssh is not running on default port(22) then,

  hostname : ssh port

- Using alias for hostname

  Alias_name ansible_ssh_port=5326 ansible_ssh_host=172.27.26.9

- Connection type and user on a per host basis:

  localhost ansible_connection=local
  other1.example.com ansible_connection=ssh ansible_ssh_user=amol

- Group Variables : can assign variable to entire group of hosts

```
[atlanta]
 host1
 host2
[atlanta:vars]
 ntp_server=ntp.atlanta.example.com
 proxy=proxy.atlanta.example.com
```

Note: Best practice is not to keep group and host vars in inventory file. Instead keep in /etc/ansible/group_vars/file_name folder.

- Dynamic Inventory

Ex. local inventories

Ansible-playbook - i /inventory/file/path playbook.yml

Ex. cloud inventories (AWS EC2)

Ansible-playbook - i /ec2.py/file/path playbook.yml

Note: 1)To make API call AWS, you need to configure Boto(Python Interface for AWS)

2) Also need to export 2 environment varaibles

export AWS_ACCESS_KEY_ID='AK123'

export AWS_SECRET_ACCESS_KEY_ID='ab123'

- Hybrid Infrastructure Inventory

Ansible-playbook - i /path/of/inventory_file_directory playbook.yml

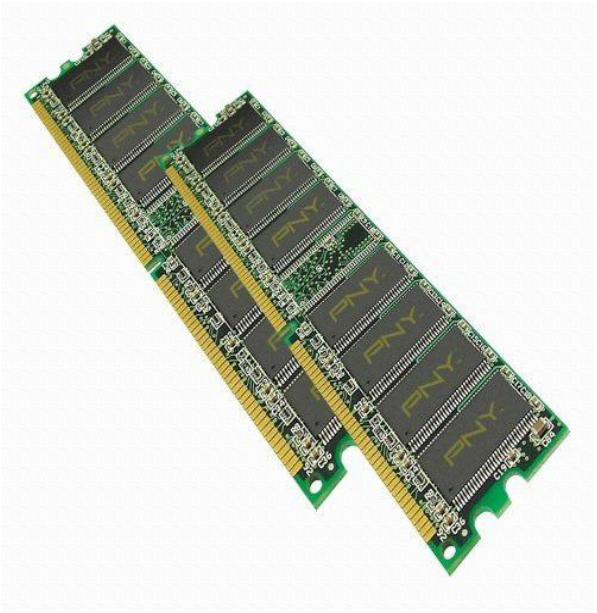
Host Defining Patterns

- All / *
- 192.168.1.*
- webservers:dbservers
- webservers:!phoenix
- webservers:&staging
- webservers[0]
- webservers[0:25]
- ansible-playbook site.yml --limit @retry_hosts.txt

Cybage® Delivering Value. Scientifically.

Modules

- They are like libraries
- Playbook internally calls these modules
- All core modules written in Python
- can be written in any language as long as they output JSON
- take parameters and conditions to define desired state
- handles processing of system resources, services, packages, files, etc. in idempotent fashion
- ansible comes preloaded with a lots of modules



```
- name: Install apache
  apt: pkg=apache2 state=present
```

Cybage® Delivering Value. Scientifically.

Ad-Hoc Commands

- These are ansible simple commands for simple tasks, such as making sure a service is running, or to trigger updates and reboots.
- run a single, one-off command
- run on a full or partial inventory
- no need to save for later
- Format : ansible <host pattern> <module> <arguments>

Ex. ansible all -m user -a "name=joe password=wat"

Cybage®
Delivering Value. Scientifically.

Playbook

- It is collection of plays (tasks)
- For more powerful configuration management
- Kept in source control, developed, validated
- Declare configurations of more complex multi-system environments
- Arrange and run tasks synchronously or asynchronously

Cybage® Delivering Value. Scientifically.

Anatomy Of Playbook

```
.
├── example_servers.yml
├── group_vars
│   ├── all
│   └── example_servers
├── host_vars
│   └── example-repository
├── hosts
├── repository_server.yml
├── roles
│   ├── nginx
│   │   |------ files
│   │   |------ templates
│   │   |------ tasks
│   │   |------ handlers
│   │   |------ vars
│   │   |------ default
│   │   |------ meta
│   ├── mongodb
│   └── common
└── site.yml
```

Cybage® Delivering Value. Scientifically.

Play Example

```
- - -
- hosts: mongo_masters
  user: root
  tasks:
    - name: Install apache2
      apt: name=apache2 state=present
      notify:
        - restart apache
    - name: ensure apache is running
      service: name=apache2 state=started
  handlers:
    - name: restart apache
      service: name=apache2 state=restarted
```

Running Playbook

ansible-playbook test.yml

ansible-playbook test.yml -- ask-pass -- ask-sudo-pass

ansible-playbook -i hosts -k -K test.yml

Tasks

- Each playbook contains a list of tasks.
- Tasks are executed in order, one at a time, against all machines matched by the host pattern, before moving on to the next task
- These tasks calls ansible modules to perform task.

Variables

- Vars can be defined at many levels (default, role, playbook, inventory)
- Then you can reference them-

1) on command line

```
$ ansible-playbook site.yml --extra-vars="usname=example domain=example.org”
```

2) in a task

```
- name: Create Vhost User
  user: name={{ usname }} state=present
```

Variables cont..

3) in a template

```
server {
listen 80;
server_name www.{{ domain }};
root /home/{{ usname }}/public_html;
index index.html index.php;
access_log /home/{{ usname }}/logs/access.log;
error_log /home/{{ usname }}/logs/error.log warn;
.....
```

4) Complex Variables

Variables cont..

5) Magical Variables

- reserved and used to access info on other hosts

Ex. 1) hostvars - lets you ask about the variables of another host, including facts

```
{{ hostvars['test.example.com']['ansible_distribution'] }}
```

2) groups - list of all the groups (and hosts) in the inventory

```
{% for host in groups['app_servers'] %}
    # something that applies to all app servers.
{% endfor %}
```

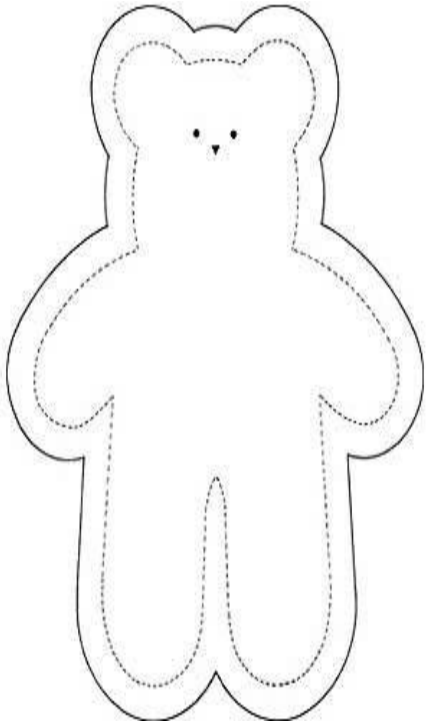
3) group_names - list (array) of all the groups the current host is in

6) Variables in other files

```
vars_files:
    - /vars/external_vars.yml
```

Templates

- Ansible uses jinja2 templating format
- Templates are interpreted by jinja2 template engine
- fill variables in differently depending on conditions
- It allows Ansible to use some programing language features
- Powerful conditionals
- Loops and iterators



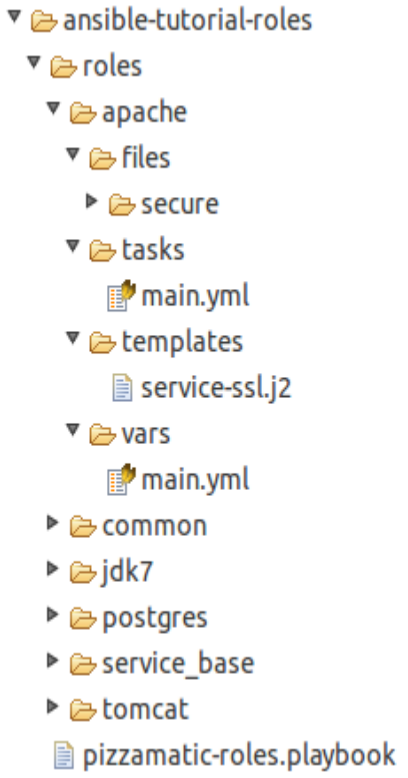
Handlers

- The things listed in the 'notify' section of a task are called handlers.
- Handlers are how you can perform various post-deployment tasks like restating some services when some of its configuration files changed. Handlers only get triggered if the state of the line actually changed
- These 'notify' actions are triggered at the end of each block of tasks in a playbook, and will only be triggered once even if notified by multiple different tasks.
- Notify handlers are always run in the order written.

Cybage®
Delivering Value. Scientifically.

Roles

- A feature of Ansible for encouraging reuse of code and best practices.
- Encapsulate a reusable service definition including Actions, Variables, Templates, Files
- Defined in standard directory structure (subproject-like)
- A host / group can have multiple roles
- A role can be applied to multiple groups
- You can upload or download roles from Ansible-Galaxy



Roles

- Parameterized roles

```
roles:
   - { role: foo_app_instance, dir: '/opt/a', port: 5000 }
```

- pre_tasks and post_tasks

Features

1) Agent less Architecture

2) Python based

3) YAML Syntax

4) Modules can be written in any language

5) Push based

6) Supports pull mode (git required)

7) Fireball mode/Parallel Mode/Accelerated mode

8) Encryption and security built in

9) Idempotent

Cons

- GUI is not well developed
- Came in 2012 , so not used by many
- Very less support for Windows

Ansible Window Support

- Starting in version 1.7, Ansible also contains support for managing Windows machines. This uses native powershell remoting, rather than SSH.
- In order for Ansible to manage your windows machines, you will have to enable Powershell remoting configured.
- A Linux control machine will be required to manage Windows hosts

Check Mode

- When ansible-playbook is executed with ”- -check” it will not make any changes on remote systems. Instead will report what changes they would have made rather than making them
- “ - - diff” flag to find out changes
- “ -- list-host” flag to list affecting hosts
- “ -- verbose” flag to see detailed output

Prompts

When running a playbook, you may wish to prompt the user for certain input, and can do so with the 'vars_prompt' section.
Ex.

```
---
-   hosts: all
    remote_user: root
    vars:
        from: "camelot"
    vars_prompt:
        name: "what is your name?"
        quest: "what is your quest?"
        favcolor: "what is your favorite color?"
```

Tags

If you have a large playbook it may become useful to be able to run a specific part of the configuration without running the whole playbook.

```
tasks:
-   name: install pkgs
    yum: name={{ item }} state=installed
    with_items:
      - httpd
      - memcached
    tags:
      - packages
```

```
aansible-playbook example.yml --tags "packages“
ansible-playbook example.yml --skip-tags "packages"
```

Fireball Mode

1) Deprecated
2) Used to sped up ansible working.
3) Alternatives are Pipelining and Accelerated mode.

Accelerared Mode

1) Use only if you are unable to use pipelining feature of ansible.
2) Accelerated mode can be anywhere from 2-6x faster than SSH.
3) Launches Demon Process Over SSH
4) Uses Socket connection
5) accelerate_port = 5099,---- default port
6) accelerate_timeout = 30, ----- timeout for receiving data from a client.(should be greater than 15s)
7) accelerate_connect_timeout = 1.0, ----- controls the timeout for the socket connect call. If 3 attempts fails then it will connect using default ssh or paramiko.

Pipelining

1) Enabling pipelining reduces the number of SSH operations required to execute a module on the remote server, by executing many ansible modules without actual file transfer.
2) This can result in a very significant performance improvement when enabled.
3) pipelining is better than accelerate mode for nearly all use cases.

Asynchronous Mode

1) By default, when you run playbook , the ssh connection remains open until the task is done on each node.
2) This may not always be desirable because you are blocking ssh connection or you may be running operations that take longer than the SSH timeout.
3) To resolve this problem Asynchronous Mode is used.
4) Here we kick off all modules at once and then poll until they are done.

Vault

"Vault" is a feature of ansible that allows encryption of ansible files and data

```
ansible-vault  create foo.yml
ansible-vault  edit foo.yml
ansible-vault  rekey foo.yml bar.yml baz.yml
ansible-vault  encrypt foo.yml bar.yml baz.yml
ansible-vault  decrypt foo.yml bar.yml baz.yml
ansible-vault  view foo.yml bar.yml baz.yml
ansible-playbook site.yml --ask-vault-pass
ansible-playbook site.yml --vault-password-file ~/.vault_pass.txt
```

Start and Step

- Start-at-task

Ex. ansible-playbook playbook.yml --start-at-task="install packages“

- Step

Ex. ansible-playbook playbook.yml –step

Perform task: install packages (y/n/c):

Task Delegation

1) Doing things on one host on behalf of another

Ex. – - hosts: webservers

tasks:

-Name: Add to load-balancer pool

command: add_to_loadbalancer_pool

delegate_to: 172.27.59.17

1) Suppose you running a playbook for group of 10 hosts. But you want particular task in playbook should run only once on particular host ,then you should use delegate_to option.

Ex. - task:

-Name: install nginx

apt: name=nginx state=present

run_once: true

delegate_to: 172.27.59.17

Cybage®
Delivering Value. Scientifically.

Any Questions?

Thank you!