Automation with Ansible

Khizer Naeem
Senior Systems Engineer
Professional Cloud Architect
Devops Enthusiast

khizernaeem@gmail.com

What is Ansible?

- Automation
- Change Management
- Provisioning
- Orchestration

Automation

- Core of Ansible
- Run tasks
  - Update a software package
  - Create a user
  - Open/Close ports
- Conditions
- Scale

Change Management

- System State
  - Declarative Model
  - Define → Enforce
  - Example
    - Apache web server version 2.4.x installed
    - PHP 5.4.x installed
    - Apache web server started
    - webadmin user exist with authorized key
  - Deviation from the state would warrant a change
  - Ansible operations are Idempotent

Provisioning

- Built on top of Automation and Change Management
- Preparing a system
- Installing, updating, configuring software
- For Example:
  - Start with a basic installation of OS
  - Update the operating system
  - Install the web server
  - Deploy the application
  - Configure the application
  - Start the web server

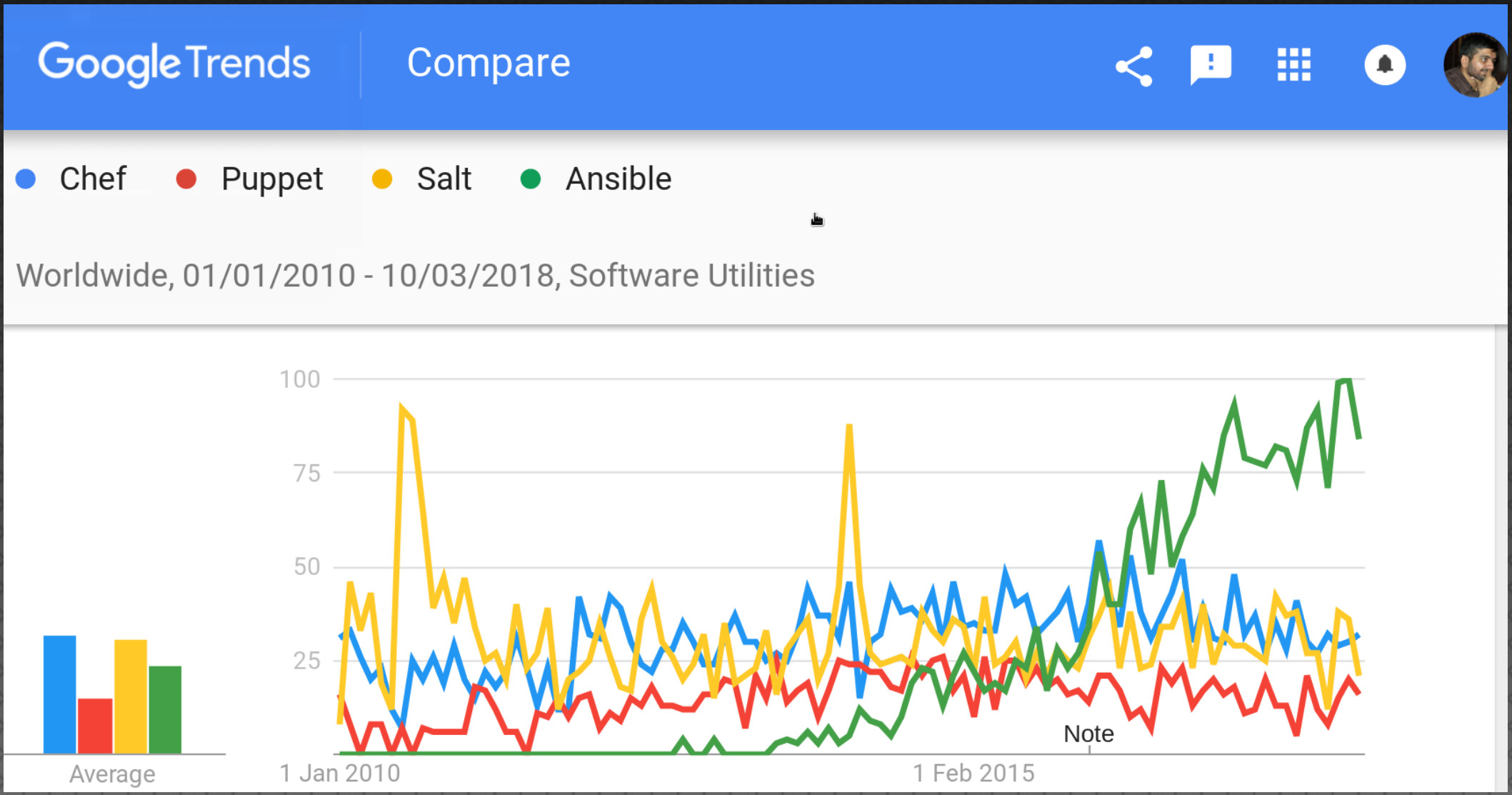
Orchestration

- Orchestration is not Automation
- Coordination between systems
- Order sensitive tasks
- For example:
  - Remove web1 from LB
  - Run tasks on web1
  - Add web1 to LB
  - ....

Why Ansible?

- Simple and Lightweight
  - No Agents
  - No database
- Multi-platform
  - Windows, Linux, Unix, Mac ..
- YAML
- Built-in Security
- Extendable

Why Ansible?


Google Trends
Compare
Chef
Puppet
Salt
Ansible
Worldwide, 01/01/2010 - 10/03/2018, Software Utilities
100
75
50
25
Note
Average
1 Jan 2010
1 Feb 2015

Why Ansible?

stackoverflow Questions Developer Jobs Tags Users 91 1 2 +10

Stack Overflow Insights > Trends

Stack Overflow Trends

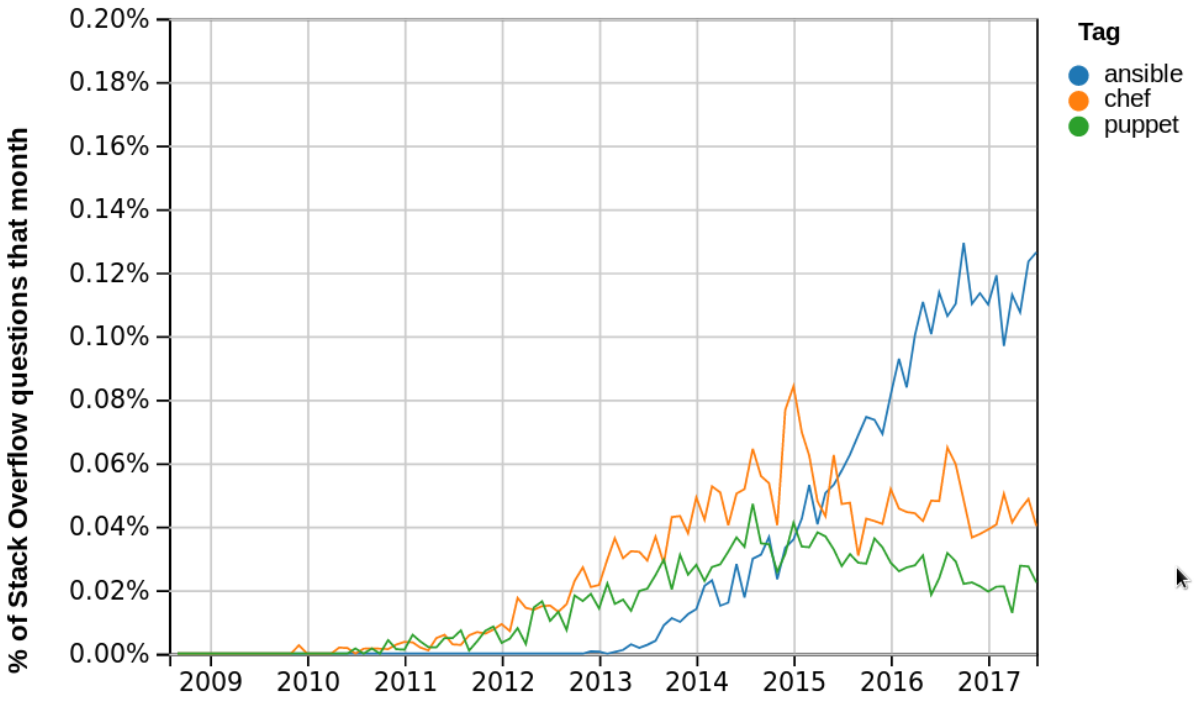
See how technologies have trended over time based on use of their tags since 2008, when Stack Overflow was founded. Enter up to 15 tags to compare growth and decline.

Tags:

ansible × chef × puppet ×

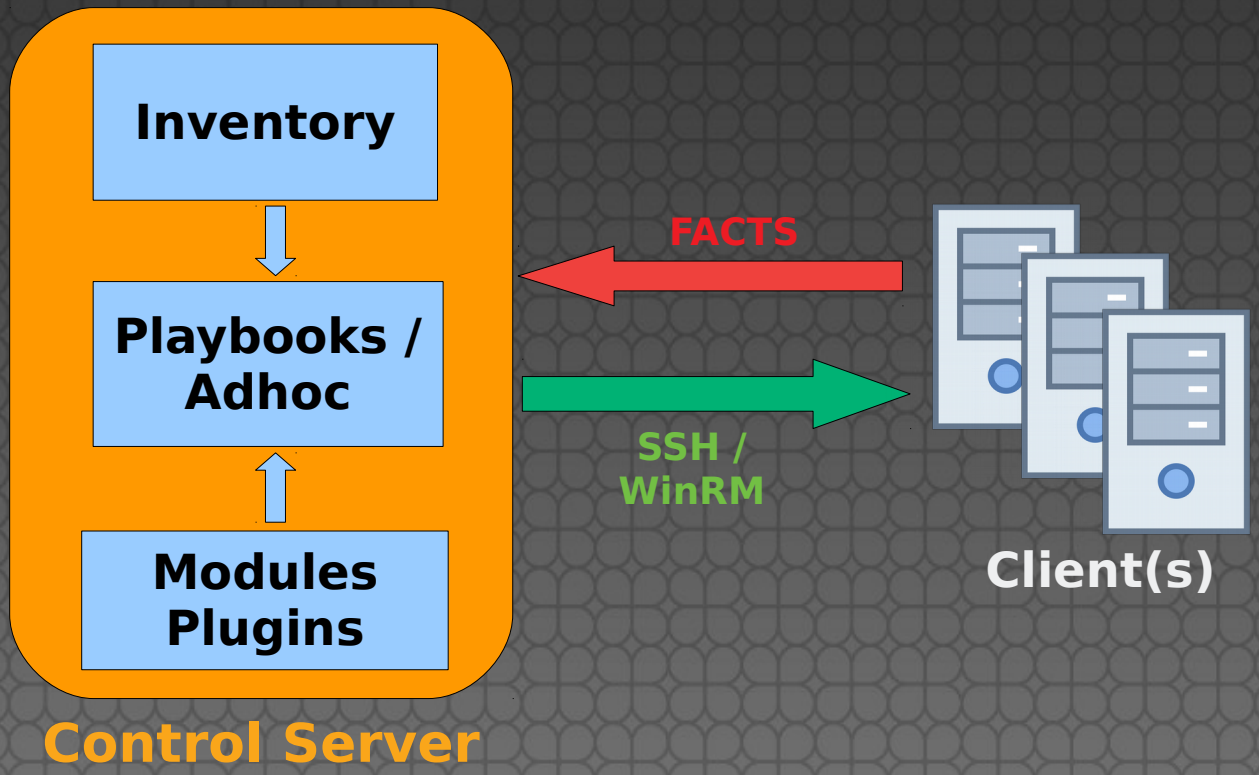
Don't know what tags to look at? Try one of our presets:

- Most Popular Languages (TIOBE Index for May 2017)
- Operating Systems
- Mobile Operating Systems
- Javascript Frameworks
- Smaller Javascript Frameworks
- Closed-source Browser Plugins
- Data Science and Big Data
- Apache Open-source Projects



Ansible Architecture

- Architecture
  - Control Server → Client
  - Gather facts from clients
- Control Server
  - Inventory
  - Playbooks
  - Modules
  - Plugins
- Client
  - SSH
  - WinRM



Inventory

- Default: /etc/ansible/hosts
- Custom inventory using -i switch
  - Use custom inventories to isolate environments e.g. Prod, dev, US
- Hosts, Groups, Variables
- Default Groups: all, ungrouped
- For example:

```
mail.example.com

[webservers]
one.example.com
alpha.example.com ansible_host=192.0.2.50

[dbservers]
one.example.com
two.example.com
three.example.com ansible_host=192.0.2.99
```

Modules

- Ansible ships with ~500 Modules
  - You can write your own!
- Each modules is automating a task for you.
- Module Support (Read the docs)
  - Core
    - Maintained by the Ansible Engineering Team
  - Network
    - Maintained by the Ansible Network Team
  - Certified
    - Part of a future planned program currently in development
  - Community
    - not maintained by Ansible

Playbooks

- Written in YAML
  - Watch the whitespaces!
- Playbooks: Collection of Plays
  - Plays: Collection of tasks
    - Tasks: Collection of modules
- Sequential order of execution
- Stops further execution on failure
  - ignore_errors: yes
  - retry file for failed hosts
- You can include other playbooks

Playbooks – Example

```
---
- hosts: webservers

  tasks:
  - name: Install Apache Webserver
    yum: name=httpd state=present

  - name: Start Apache Webserver
    service: name=httpd state=started enabled: yes

- hosts: dbservers

  tasks:
  - name: Install MariaDB Server
    yum: name=mariadb-server state=present

  - name: Start MariaDB Server
    service: name=mariadb-server state=started enabled: yes
```

Playbooks – Example

```
---
- hosts: webservers

  tasks:
  - name: Install Apache Webserver
    yum: name=httpd state=present

  - name: Start Apache Webserver
    service: name=httpd state=started enabled: yes

- hosts: dbservers

  tasks:
  - name: Install MariaDB Server
    yum: name=mariadb-server state=present

  - name: Start MariaDB Server
    service: name=mariadb-server state=started enabled: yes
```

Playbooks – Example

```
---
- hosts: webservers

  tasks:
  - name: Install Apache Webserver
    yum: name=httpd state=present

  - name: Start Apache Webserver
    service: name=httpd state=started enabled: yes

- hosts: dbservers

  tasks:
  - name: Install MariaDB Server
    yum: name=mariadb-server state=present

  - name: Start MariaDB Server
    service: name=mariadb-server state=started enabled: yes
```

Ansible Advance Topics

- Variables
- Conditions
- Handlers
- Loops
- Templates
- Includes
- Roles
- Ansible Galaxy
- Playbook Scripts

Variables

- Facts

```
ansible webservers -m setup
```

- Magic Variables
  - hostvars, group_names, groups
- Variables Defined in:
  - Inventory
  - Playbook
  - Include files
  - Roles

Variables

- Variables in inventory

```
webserver.labXX ansible_port=2992 ansible_host=1.2.3.4
webserver.labXX http_port=80 maxRequestsPerChild=100
```

```
[app]
webserver.lab
database.lab

[webservers]
webserver.lab

[dbservers]
database.lab

...
```

```
...
[app:vars]
ntp_server=1.2.3.4

[webservers:vars]
http_port=80
htdocs=/var/www/html

[dbservers:vars]
mariadb_port=3306
db_user = dbadmin
```

- Inventory variables in files:
  - /etc/ansible/host_vars/webserver.lab.yml
  - /etc/ansible/group_vars/app.yml

Variables

- Variables in playbook

```
- hosts: webservers

  vars:
    http_port: 80
    htdocs: /var/www/html

  tasks:
    - name: Blah blah
      module:
        ...
```

Variables

- Register variables

```
- hosts: webservers

  tasks:
    - name: Run shell script
      shell: /root/script.sh
      register: script_output
    ...
```

Conditions

- When Statement

```
- hosts: webservers

  tasks:
    - name: Run shell script
      yum: name=httpd state=present
      when: ansible_os_family == "RedHat"

    - name: Run shell script
      apt: name=apache2 state=present
      when: ansible_os_family == "Debian"

    ...
```

Conditions

- "When" on Register variables

```
- hosts: all

  tasks:
      - name: Check apache vhost conf file
        stat:
            path: /etc/httpd/conf.d/app.conf
        register: appconf

      - name: Copy appconf file
           copy:
             src: /opt/application/apache/app.conf
             dest: /etc/httpd/conf.d/app.conf
        when: not appconf.stat.exists

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

Handlers

- Running Operations On Change

```
- hosts: all

  tasks:
      - name: Check apache vhost conf file
        stat:
            path: /etc/httpd/conf.d/app.conf
        register: appconf

      - name: Copy appconf file
          copy:
            src: /opt/application/apache/app.conf
            dest: /etc/httpd/conf.d/app.conf
        when: not appconf.stat.exists
        notify: Restart Apache

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

Loops

- Standard Loops using “with_items:”

```
- hosts: all

  tasks:
    - name: Add user user1
      user:
        name: "user1"
        state: present
        groups: "wheel"

    - name: Add user user2
      user:
        name: "user2"
        state: present
        groups: "wheel"
```

```
- hosts: all

  tasks:
    - name: add users user1 and 2
      user:
        name: "{{ item }}"
        state: present
        groups: "wheel"
      with_items:
        - user1
        - user2
```

Loops

- File iteration using “with_file”

```
- hosts: all

  tasks:
    - name: Copy app.php
      copy:
        src: /opt/app/app.php
        dest: /var/www/html/
        owner: apache
        mode: 600

    - name: Copy config.php
      copy:
        src: /opt/app/config.php
        dest: /var/www/html/
        owner: apache
        mode: 600
```

```
- hosts: all

  tasks:
    - name: Copy app files
      copy:
        src: “{{ item }}”
        dest: /var/www/html/
        owner: apache
        mode: 600
      with_file:
        - “/opt/app/app.php”
        - “/opt/app/config.php”
```

Loops

- File iteration using “with_fileglob”

```
- hosts: all

  tasks:
    - name: Copy app.php
      copy:
        src: /opt/app/app.php
        dest: /var/www/html/
        owner: apache
        mode: 600

    - name: Copy config.php
      copy:
        src: /opt/app/config.php
        dest: /var/www/html/
        owner: apache
        mode: 600
```

```
- hosts: all

  tasks:
    - name: Copy app files
      copy:
        src: “{{ item }}”
        dest: /var/www/html/
        owner: apache
        mode: 600
      with_fileglob:
        - “/opt/app/*.php”
```

Templates

- Ansible uses jinja2 templating engine
- Template modules
  - Similar to copy module
  - Replaces the variables
  - Can contain loops and conditions
- Check the official Jinja2 docs:

  http://jinja.pocoo.org/docs/2.9/

Templates

- Jinja2 Basics
  - `{% ... %}` for Statements
  - `{{ ... }}` for Expressions
  - `{# ... #}` for Comments
- Variables
  - `{{ foo.bar }}`
- Filters
  - `{{ htmldata | striptags | title }}`
  - `{{ list | join(', ') }}`

Templates

- Example: ntp.conf.j2

```
driftfile /var/lib/ntp/drift

restrict 127.0.0.1
restrict -6 ::1

server {{ ntpserver }}

includefile /etc/ntp/crypto/pw

keys /etc/ntp/keys
```

Templates

- Example: my.cnf.j2

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to
prevent assorted security risks
symbolic-links=0
port={{ mysql_port }}

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

Templates

- Using the templates

```
...
- name: Configure ntp file
  template:
    src: ntp.conf.j2
    dest: /etc/ntp.conf
  notify: Restart ntp

...

- name: Configure MariaDB
  template:
    src: my.cnf.j2
    dest: /etc/my.cnf
  notify: restart mariadb
...
```

Includes

- Break up bits of configuration policy into smaller files
- Simplify, organize and reuse plays
- Task includes
  - Inclusions under the “tasks” directive
- Play includes
  - Inclusions along the same level of tasks
- You can pass variables when calling the include statement
  - One “template” playbook with variables can be used multiple times with different variables
- Example:

Includes – Example

```
# tasks/common.yml

- name: Disable SELinux
  selinux:
    state: disabled

- name: Install Vim
  yum:
    name: vim
    state: present
```

```
# tasks/httpd.yml
- name: Install httpd
  yum:
    name: httpd
    state: present

- name: Start httpd
  service:
    name: httpd
    state: started
    enabled: yes
```

```
# tasks/mariadb.yml
- name: Install mariadb
  yum:
    name: mariadb-server
    state: present

- name: Start mariadb
  service:
    name: mariadb
    state: started
    enabled: yes
```

```
- hosts: all
  tasks:
    - include tasks/common.yml

- hosts: webservers
  tasks:
    - include: tasks/httpd.yml

- hosts: dbservers
  tasks:
    - include: tasks/httpd.yml
```

Ansible Roles

- Best way to organize your playbooks
- Includes on steroids
  - no additional magic except directory structure and search path handling
- Special directory structure
  - You don't need to have all the directories
  - Only what you need
  - A simple role will only have tasks/main.yml
- main.yml
  - can have include files beside it

```
examplerole/
├── defaults/
│   └── main.yml
├── files/
│   └── app.zip
├── handlers/
│   └── main.yml
├── meta/
│   └── main.yml
├── tasks/
│   └── main.yml
├── templates/
│   └── conf.ini.j2
└── vars/
    └── main.yml
```

Ansible Roles – Rules

- If tasks/main.yml exists, tasks listed therein will be added to the play.
- If handlers/main.yml exists, handlers listed therein will be added to the play.
- If vars/main.yml exists, variables listed therein will be added to the play.
- If defaults/main.yml exists, variables listed therein will be added to the play.
- If meta/main.yml exists, any role dependencies listed therein will be added.
- Any copy, script, template or include tasks in the role can reference files in: files, templates, tasks without having to path them relatively or absolutely

```
examplerole/
├── defaults/
│   └── main.yml
├── files/
│   └── app.zip
├── handlers/
│   └── main.yml
├── meta/
│   └── main.yml
├── tasks/
│   └── main.yml
├── templates/
│   └── conf.ini.j2
└── vars/
    └── main.yml
```

Ansible Roles – Example

```
# roles/memcached/tasks/main.yml
- name: Install memcacahed
  yum:
        name: memcached
        state: present

- name: Enable memecached service
  service:
        name: memcached
        state: started
```

```
# roles/httpd/tasks/main.yml
- name: Install httpd
  yum:
        name: httpd
        state: present

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

```
# playbook.yml

- hosts: webservers

  roles:
    - memcached
    - httpd
```

```
roles/
├── httpd/
│   └── tasks/
│       └── main.yml
└── memcached/
    └── tasks
        └── main.yml
```

Ansible Galaxy

- Free repository of community developed roles
- You can also use the site to share roles that you create
- Uses github authentication
- You can deploy your own internal Galaxy server
- Installing Roles

```
ansible-galaxy install username.role_name
```

- By default installs to /etc/ansible/roles
- A good reference point for writing your own

Ansible Playbooks as Scripts

- Ansible tries not to be a programming language
- Capable of replacing scripts:
  - vars_prompt, pause can allow interactivity
  - -e “var=value” for non-interactive
  - -e @file.json can import json data files
- Shabang the playbook:

```
#!/usr/bin/ansible-playbook

- hosts: webservers

  roles:
    - role1
    - role2
```

Ansible Scripts – Example

```
#!/usr/bin/ansible-playbook

- hosts: all
  tasks:
    - block:
        - name: Restart Apache
          service: name=httpd24-httpd state=restarted
        - name: Wait for Apache to restart
          wait_for: port=80 timeout=10
      rescue:
        - shell: journalctl --no-pager -u httpd24-httpd -n100
          register: service_status
        - name: Restarting failed emailing ops
          mail:
            from: k.naeem@hbmsu.ac.ae
            to: k.naeem@hbmsu.ac.ae
            subject: "Apache restart failed on {{ inventory_hostname }}"
            body: "{{service_status}}"
            host: "mail.hbmsu.ac.ae"
          delegate_to: localhost
```

Thank You