DevOps
Ansible

Trainer: Abhijith V G – AWS, eCommerce, Mobile & DevOps Architect

Magento Certified DEVELOPER PLUS

amazon web services | Certified

Developer - Associate

Ansible: Introduction to Ansible

Open Source Project started in 2012 by Michael De Haan to eliminate complexities of other Configuration management tools.

Principles of Ansible

SIMPLE

Human readable automation

No special coding skills needed

Tasks executed in order

Get productive quickly

POWERFUL

App deployment

Configuration management

Workflow orchestration

Orchestrate the app lifecycle

AGENTLESS

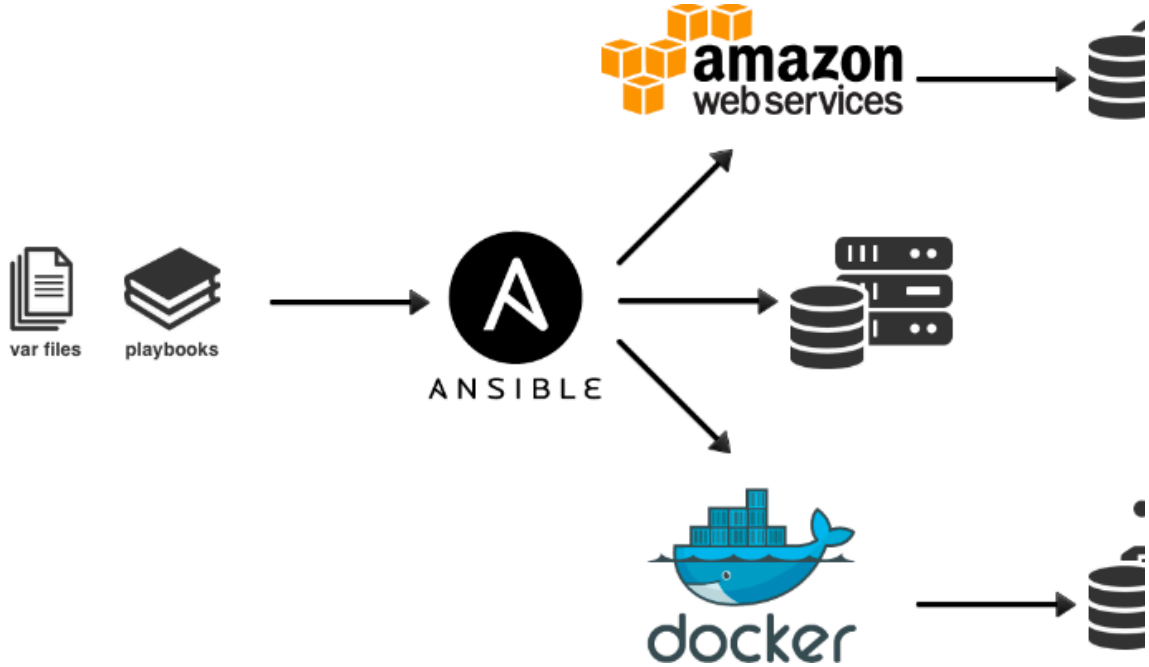
Agentless architecture

Uses OpenSSH & WinRM

No agents to exploit or update

More efficient & more secure

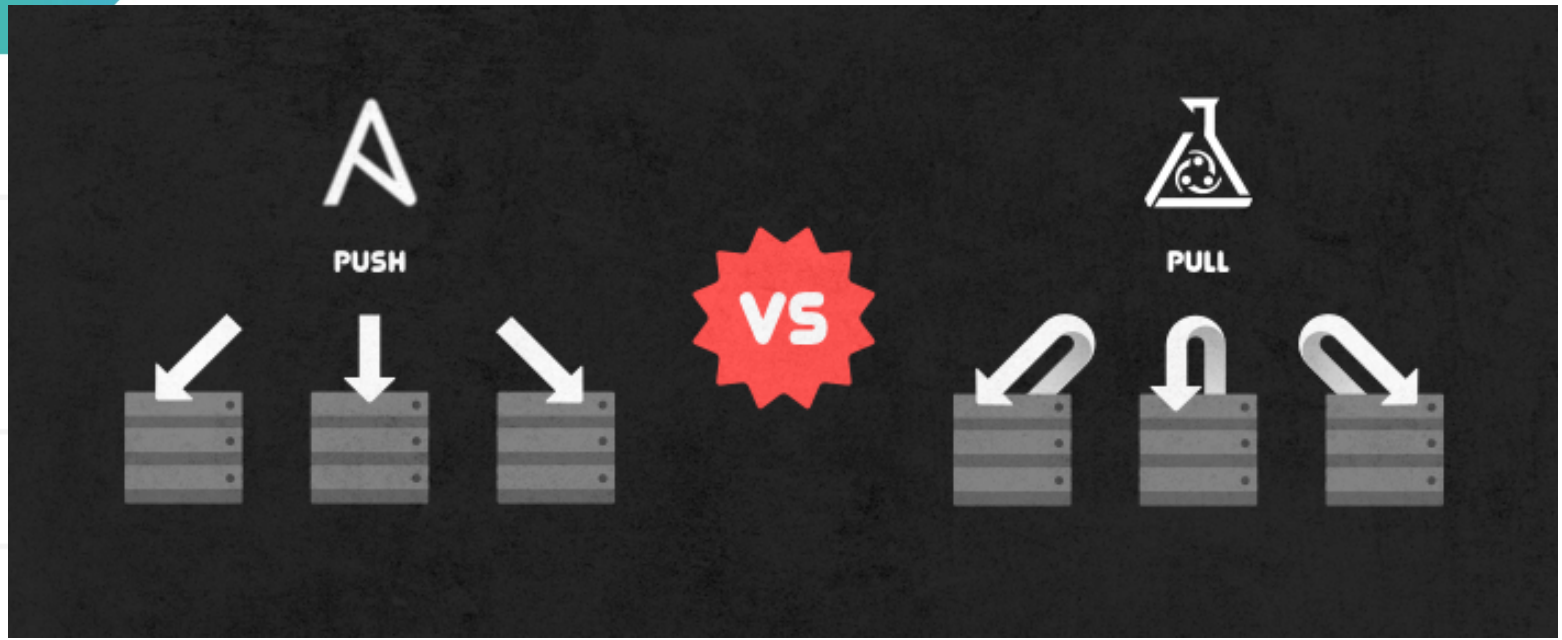
Ansible: Introduction to Ansible



Ansible: Introduction to Ansible: Uses Push not Pull (Chef/Puppet)

Ansible

Chef/Puppet



Ansible: Introduction to Ansible: Uses Push not Pull (Chef/Puppet)

A function is idempotent if repeated applications has the same affect as a single application

IDEMPOTENCE

Ansible: Introduction to Ansible: Uses Push not Pull (Chef/Puppet)

It's clean!

- No agents
- No database
- No residual software
- No complex upgrades

Ansible: Introduction to Ansible: Uses Push not Pull (Chef/Puppet)

YAML

Ansible Execution

No programming required

NOT a markup language

Structured

Easy to read and write

Whitespace and comments

- # comments start with a hash-mark.
- Yml indents with 2 spaces.
- White space helps readability.
- Add a blank line before vars, roles, tasks, handlers.

Ansible: Format of a YAML Playbook

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

- name: 'install.yml'              # quote names for syntax highlighting
  hosts: localhost                 # scope the play appropriately
  connection: local                #
  gather_facts: False              # booleans: /^(y|yes|n|no|true|false|on|off)$/i

  tags:                            # use tags for plays, and actions
    - preparation

  vars:                            # use group_vars for environment specifics
    - url: "https://galaxy.ansible.com"  # quote when value has ':'

  tasks:                           # list tasks, but consider using a role
    - name: 'check network'        # format parameters for small terminal size
      uri:                         # the best way is to use 'Native YML' format
        url: "{{ url }}"
        method: HEAD
        return_content: no
        status_code: 200
        timeout: 60
        follow_redirects: all

    - name: 're-import roles from Galaxy'
      command: ansible-galaxy install --force -r roles/requirements.yml
```

Ansible: Introduction to Ansible: Control Server

Python 2.6+

Must be *NIX (Linux/Unix/Mac)

Windows not supported

Ansible: Introduction to Ansible: Node/Remote Servers

*NIX:

Python 2.4 (simplejson)

Python 2.5+

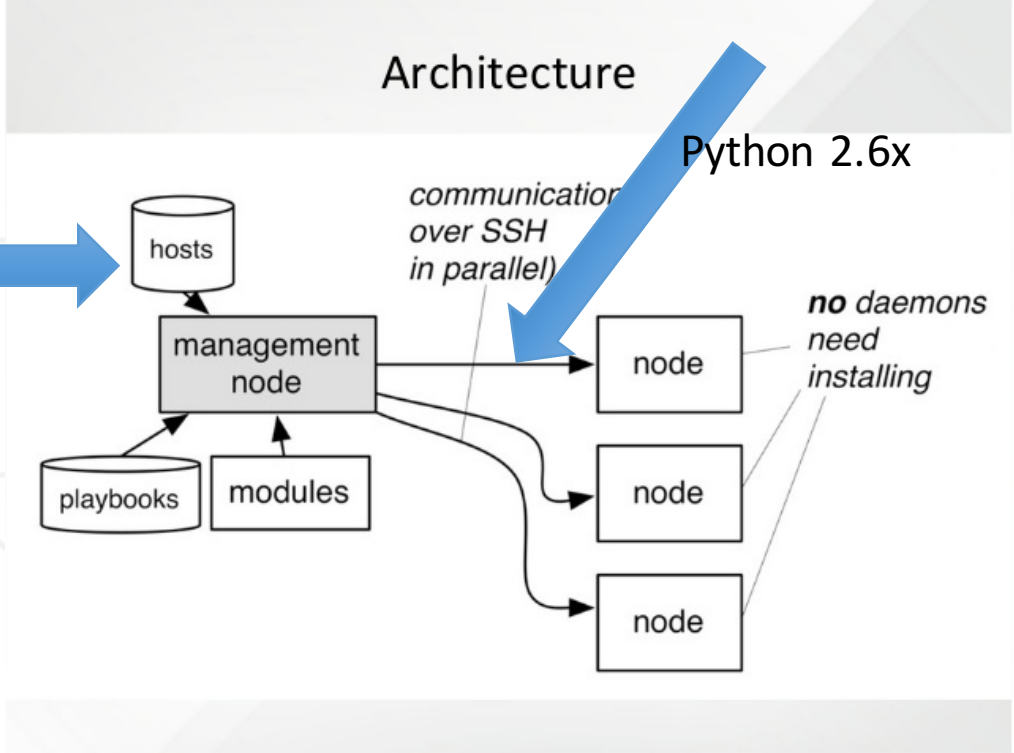
SSH

Windows:

Remote Powershell Enabled

Ansible: Architecture

Inventory



Ansible: Architecture

Module

A programmed unit of work to be done.

Play

A single task from a module, executed on a host or set of hosts

Playbook

A set of plays built in specific order sequence to produce an expected outcome or outcomes.

Ansible: Ansible Terminology

Playbooks: One or more plays. Each play in turn is responsible for mapping tasks to groups of hosts. Tasks are typically calls to Ansible modules.

Plays: A playbook is a list/collection of plays. A play is minimally a mapping between a set of and the tasks which run on those hosts to define the role that those systems will perform.

Task: Playbooks exist to run tasks. Tasks combine an action (a module and its arguments) with a name.

```
SreeMacMin16GB:ws sree$ cat ansible/playbook.yml
- hosts:
    - web
  tasks:
    - action: ping
```

Ansible: Ansible Terminology

Modules:

Modules are the units of work that Ansible ships out to remote machines.

Roles:

Roles are units of organization in Ansible.

Facts:

Facts are simply things that are discovered about remote nodes. They can be used in playbooks and templates just like variables. Facts are things that are inferred, rather than set. Facts are automatically discovered by Ansible when running plays by executing the internal setup module on the remote nodes. You never have to call the setup module explicitly, it just runs, but it can be disabled to save time if it is not needed or you can tell ansible to collect only a subset of the full facts via the **gather_subset: option**. For the convenience of users who are switching from other configuration management systems, the *fact module will also pull in facts from the ohai and facter tools if they are installed*. (These may also be disabled via gather_subset:)

Ansible: Ansible Terminology

Rolling Updates:

The act of addressing a number of nodes in a group N at a time to avoid updating them all at once and bringing the system offline.

For instance, in a web topology of 500 nodes handling very large volume, it may be reasonable to update 10 or 20 machines at a time, moving on to the next 10 or 20 when done. The *serial: keyword* in an Ansible playbooks control the size of the rolling update pool. The *default is to address the batch size all at once*, so this is something that you must opt-in to.

Ansible Galaxy:

Ansible Galaxy refers to the Galaxy website where users can share roles, and to a command line tool for installing, creating and managing roles.

Jinja2:

Jinja2 is the preferred templating language of Ansible's template module. It is a very simple Python template language that is generally readable and easy to write.

Ansible: Installation & Configuration

http://docs.ansible.com/ansible/latest/intro_installation.html

DEBIAN:

```
sudo apt-get update
sudo apt-get install software-properties-common
sudo apt-add-repository ppa:ansible/ansible
sudo apt-get update
```

CENTOS:

```
sudo yum install epel-release
sudo yum install ansible
```

Ansible: Lab Setup Node Setup (3 Servers)

```
docker pull schogini/docker-ansible-ssh-node-ubuntu
docker run -d –p 80 --name node1 schogini/docker-ansible-ssh-node-ubuntu
docker run -d –p 80 --name node2 schogini/docker-ansible-ssh-node-ubuntu
docker run -d –p 80 --name node3 schogini/docker-ansible-ssh-node-ubuntu
```

Find the IP Addresses of each of these servers(nodes)

```
Sree>docker inspect --format '{{.NetworkSettings.IPAddress }}' node1 node2 node3
172.17.0.5
172.17.0.6
172.17.0.7
Sree>
```

Ansible: Lab Setup Ansible Workstation (Ansible Server/Management)

Ansible Workstation Image

docker pull schogini/docker-ansible-ws-ubuntu

Copy the default /etc/ansible folder to the local system so that we can use it as a base to create use & throw ansible workstation containers.

```
Sree>mkdir -p ws
Sree>cd ws
```

```
Sree>docker run --rm -tid --rm --name ansible-ws --entrypoint bash schogini/docker-ansible-ws-ubuntu
9550b942ee19ae452ab657cd689ee019267ccf5711fbed7451cfd38a9e622e38
Sree>docker cp ansible-ws:/etc/ansible .
Sree>ls -l ansible/
total 56
-rw-r--r--  1 sree  staff  19840 Sep 21 14:47 ansible.cfg
-rw-r--r--  1 sree  staff     56 Sep 21 14:47 hosts
-rw-r--r--  1 sree  staff     47 Sep 21 14:47 playbook.yml
drwxr-xr-x  2 sree  staff     68 Sep 20 04:14 roles
Sree>docker rm -f ansible-ws
ansible-ws
Sree>
```

Ansible: How to add nodes to server (Ansible Workstation)

- The file in which the Ansible nodes are defined is called ***Inventory***
- Text files that follows the below format can be passed to the ansible commands as an inventory file.
- By default, ansible will search for a file by the name *hosts*.
- Inventory files generally do not have any extension.
- Host files can be used to define environments, group hosts, variables and more.

```
[web]
172.17.0.6
172.17.0.7

[db]
172.17.0.6
172.17.0.8
```

Ansible: Lab Setup Ansible Workstation

```
Sree>nano ansible/hosts
```

```
[web]
172.17.0.5
172.17.0.6

[db]
172.17.0.5
172.17.0.7
```

Ansible: Lab Setup : Test Ansible

```
Sree>docker run --rm -v $PWD/ansible:/etc/ansible schogini/docker-ansible-ws-ubuntu ansible all -m ping
172.17.0.7 | SUCCESS => {
    "changed": false,
    "failed": false,
    "ping": "pong"
}
172.17.0.6 | SUCCESS => {
    "changed": false,
    "failed": false,
    "ping": "pong"
}
172.17.0.5 | SUCCESS => {
    "changed": false,
    "failed": false,
    "ping": "pong"
}
Sree>
```

Ansible: Lab Setup Ansible Workstation

```
Sree>cat ansible/playbook.yml
- hosts:
    - web
  tasks:
    - action: ping
Sree>
```

Ansible: Lab Setup : Test Playbook

```
Sree>docker run --rm -v $PWD/ansible:/etc/ansible schogini/docker-ansible-ws-ubuntu ansible-playbook /etc/ansible/playbook.yml

PLAY [web] *********************************************************************

TASK [Gathering Facts] *********************************************************
ok: [172.17.0.6]
ok: [172.17.0.5]

TASK [ping] ********************************************************************
ok: [172.17.0.5]
ok: [172.17.0.6]

PLAY RECAP *********************************************************************
172.17.0.5                 : ok=2    changed=0    unreachable=0    failed=0
172.17.0.6                 : ok=2    changed=0    unreachable=0    failed=0

Sree>
```

Ansible: Global Configuration: /etc/ansible/ansible.cfg

Sample config file: https://raw.githubusercontent.com/ansible/ansible/devel/examples/ansible.cfg

```
Sree>cat ansible/ansible.cfg
# config file for ansible -- https://ansible.com/
# ===============================================

# nearly all parameters can be overridden in ansible-playbook
# or with command line flags. ansible will read ANSIBLE_CONFIG,
# ansible.cfg in the current working directory, .ansible.cfg in
# the home directory or /etc/ansible/ansible.cfg, whichever it
# finds first

[defaults]

# some basic default values...

#inventory      = /etc/ansible/hosts
#library        = /usr/share/my_modules/
#module_utils   = /usr/share/my_module_utils/
#remote_tmp     = ~/.ansible/tmp
#local_tmp      = ~/.ansible/tmp
#forks          = 5
#poll_interval  = 15
```

Ansible: Components of Ansible

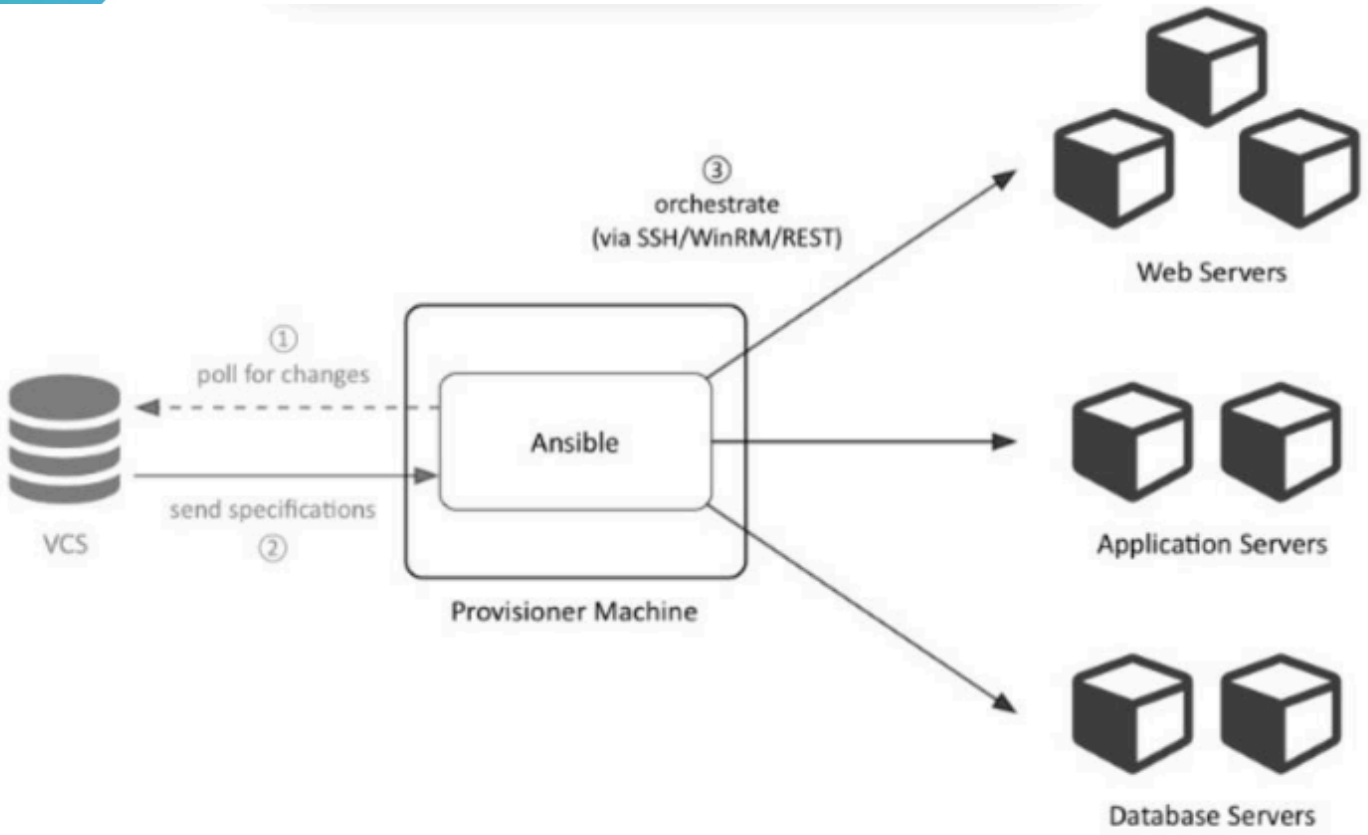
The three major components of Ansible:

1. **An automation language that describes application infrastructure in Ansible Playbooks.** Playbooks are expressed in YAML format and have a minimum of syntax, which intentionally tries to not be a programming language or script, but rather a model of a configuration or a process.
2. **An automation engine that runs Playbooks.**
3. **An enterprise framework, called *Ansible Tower*, that controls, secures and manages automation.**

Ansible: Inventory

- Use separate inventories for stages like 'test' and 'production'.
- Target an environment with the -i flag.
- Test things first in a stage environment before running them in 'production'.

Ansible: Inventory: Host Groups



Ansible: Shell Commands

```
Sree>docker run --rm -v $PWD/ansible:/etc/ansible schogini/docker-ansible-ws-ubuntu ansible all -m shell -a "echo 'Hello'"
172.17.0.6 | SUCCESS | rc=0 >>
Hello
172.17.0.5 | SUCCESS | rc=0 >>
Hello
172.17.0.7 | SUCCESS | rc=0 >>
Hello
Sree>
```

Ansible: Modules

1. Modules are the units of work that Ansible ships out to remote machines.
2. Modules can be implemented in any language, including Perl, Bash, or Ruby – but can leverage some useful communal library code if written in Python.
3. Modules just have to return JSON.
4. Once modules are executed on remote machines, they are removed, so no long running daemons are used. Ansible refers to the collection of available modules as a library.
5. Ansible ships with a number of modules (called the 'module library') that can be executed directly on remote hosts (Ad-hoc commands) or through Playbooks.
   - Ad-Hoc Commands: Execution directly via CLI using ***ansible*** command.
   - Playbooks: Executing playbooks using ***ansible-playbook*** command.
6. Users can also write their own modules. These modules can control system resources, like services, packages, or files (anything really), or handle executing system commands.
7. Ansible has more than a few modules built in and you can find out about them here: http://docs.ansible.com/ansible/latest/modules_by_category.html

Ansible: Playbooks (One Play)

A "play" is one group of hosts and tasks, and a playbook can contain multiple instances of these.

```
- hosts:
    - hostXXX.ws.nsrc.org
    - hostYYY.ws.nsrc.org
  tasks:
    - name: ensure package cache is up to date
      apt: update_cache=yes cache_valid_time=3600
      tags: install
    - name: install web server
      apt: pkg=apache2 state=present
      tags: install
    - name: install index page
      copy: src=front.html dest=/var/www/html/index.html backup=yes
      tags: configure
      notify: restart apache2
  handlers:
    - name: restart apache2
      service: name=apache2 state=restarted
```

Ansible: Playbooks (Two Plays)

```
- hosts:
    - hostXXX.ws.nsrc.org
    - hostYYY.ws.nsrc.org
  tasks:
    - name: ensure package cache is up to date
      apt: update_cache=yes cache_valid_time=3600
      tags: install
    - name: install web server
      apt: pkg=apache2 state=present
      tags: install
    - name: install index page
      copy: src=front.html dest=/var/www/html/index.html backup=yes
      tags: configure
      notify: restart apache2
  handlers:
    - name: restart apache2
      service: name=apache2 state=restarted

- hosts:
    - hostZZZ.ws.nsrc.org
  tasks:
    - name: install mysql server
      apt: pkg=mysql-server state=present
```

Ansible: Ansible Commands

Executing a playbook: ansible-playbook playbook.yml
Ping hosts: ansible <HOST_GROUP> -m ping
Display gathered facts: ansible <HOST_GROUP> -m setup | less
Filter gathered facts: ansible <HOST_GROUP> -m setup -a "filter=ansible_distribution*"
Copy SSH key manually: ansible <HOST_GROUP> -m authorized_key -a "user=root key='ssh-rsa AAAA...XXX == root@hostname'"
Specifying a user: ansible-playbook playbooks/atmo_playbook.yml --user atmouser
Using a specific SSH private key: ansible -m ping hosts --private-key=~/.ssh/keys/id_rsa -u centos
Passing arguments: ansible-playbook playbooks/atmo_playbook.yml -e "ATMOUSERNAME=atmouser"
Limit to one host: ansible-playbook playbooks/PLAYBOOK_NAME.yml --limit "host1"
Limit to multiple hosts: ansible-playbook playbooks/PLAYBOOK_NAME.yml --limit "host1,host2"

Ansible: Ansible Commands

Negated limit: ansible-playbook playbooks/PLAYBOOK_NAME.yml --limit 'all:!host1' [NOTE: Single quotes MUST be used to prevent bash interpolation.]
Limit to host group: ansible-playbook playbooks/PLAYBOOK_NAME.yml --limit 'group1'
Limit to all tags matching install: ansible-playbook playbooks/PLAYBOOK_NAME.yml --tags 'install'
Skip any tag matching sudoers: ansible-playbook playbooks/PLAYBOOK_NAME.yml --skip-tags 'sudoers'
Clear Cache: ansible-playbook playbooks/PLAYBOOK_NAME.yml --flush-cache
Check for bad syntax: ansible-playbook playbooks/PLAYBOOK_NAME.yml --syntax-check
Run in dry-run mode: ansible-playbook playbooks/PLAYBOOK_NAME.yml --check
Using raw module to run command similar to running directly via SSH: ansible -m raw -s -a "yum install libselinux-python -y" new-atmo-images

https://ansible-tips-and-tricks.readthedocs.io/en/latest/ansible/commands/

Ansible: Roles

Roles are units of organization in Ansible.

Assigning a role to a group of hosts (or a set of groups, or host patterns, etc.) implies that they should implement a specific behavior. A role may include applying certain variable values, certain tasks, and certain handlers – or just one or more of these things. Because of the file structure associated with a role, roles become redistributable units that allow you to share behavior among playbooks – or even with other users.

Roles allow you to call a set of variables, tasks, and handlers by simply specifying a defined role. Roles require the use of a defined file structure in order to work. Per the Ansible documentation, that structure looks like this…

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

Ansible: Roles

image

```
roles/
    common/               # this hierarchy represents a "role"
        tasks/            #
            main.yml      #  <-- tasks file can include smaller files if warranted
        handlers/         #
            main.yml      #  <-- handlers file
        templates/        #  <-- files for use with the template resource
            ntp.conf.j2   #  <------- templates end in .j2
        files/            #
            bar.txt       #  <-- files for use with the copy resource
            foo.sh        #  <-- script files for use with the script resource
        vars/             #
            main.yml      #  <-- variables associated with this role
        defaults/         #
            main.yml      #  <-- default lower priority variables for this role
        meta/             #
            main.yml      #  <-- role dependencies

    webtier/              # same kind of structure as "common" was above, done for the webtier role
    monitoring/           # ""
    fooapp/               # ""
```

Ansible: Roles

A sample playbook without using roles

```
---
- hosts: linuxservers
  tasks:
    - name: Install Apache Web Server
      yum: name=httpd state=latest
      notify:
        - openport
        - startwebserver
  handlers:
    - name: openport
      service: name=httpd state=started
    - name: startwebserver
      firewalld: port=80/tcp permanent=true state=enabled immediate=yes

- hosts: ansibleclient2
  tasks:
    - name: upload index page
      get_url: url=http://www.purple.com/faq.html dest=/var/www/html/index.html
```

Ansible: Creating a Role

Step 1: Create the folders. Remember, the role name is the folder name.

```
1 mkdir -p /etc/ansible/roles/webserver/tasks
2 mkdir -p /etc/ansible/roles/webserver/handlers
3 mkdir -p /etc/ansible/roles/webserver/templates
4 mkdir -p /etc/ansible/roles/webserver/files
5 mkdir -p /etc/ansible/roles/webserver/vars
6 mkdir -p /etc/ansible/roles/webserver/defaults
7 mkdir -p /etc/ansible/roles/webserver/meta
```

Ansible: Roles

Step 2: Create a task inside the role's tasks/ folder

The first thing we want to do is create a 'main.yml' file in the tasks folder that looks like this...

```
1 ---
2 - name: Install Apache Web Server
3   yum: name=httpd state=latest
4   notify:
5     - openport
6     - startwebserver
```

Ansible: Roles

Step 3: Create a handler inside the role's handlers/ folder

Notice that this is just the task that was defined in our initial playbook. Next create another 'main.yml' file under the handler directory that looks like this…

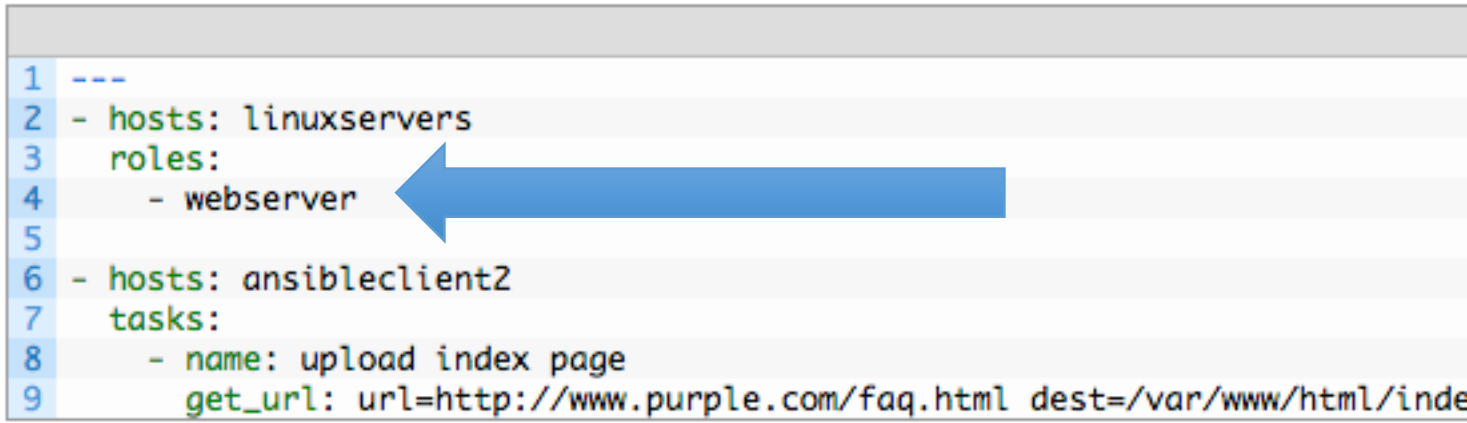
```
1 ---
2 - name: openport80
3   service: name=httpd state=started
4 - name: startwebserver
5   firewalld: port=80/tcp permanent=true state=enabled immediate=yes
```

Similarly, you can create files in the other folders and they will in auto included wherever you apply the role

Ansible: Using Roles

```
---
- hosts: linuxservers
  tasks:
    - name: Install Apache Web Server
      yum: name=httpd state=latest
      notify:
        - openport
        - startwebserver
  handlers:
    - name: openport
      service: na
    - name: start
      firewalld:

- hosts: ansiblec
  tasks:
    - name: uploa
      get_url: ur
```



```
---
- hosts: linuxservers
  roles:
    - webserver

- hosts: ansibleclient2
  tasks:
    - name: upload index page
      get_url: url=http://www.purple.com/faq.html dest=/var/www/html/inde
```

Tag all the things

- Tags help organizing playbooks.
- Tags can help in testing.
- You can run or skip parts of playbooks:

  --tags=only,run,these,tags

  --skip-tags=tags,to,skip

Ansible: Tags

```
- hosts:
    - hostXXX.ws.nsrc.org
    - hostYYY.ws.nsrc.org
  tasks:
    - name: ensure package cache is up to date
      apt: update_cache=yes cache_valid_time=3600
      tags: install
    - name: install web server
      apt: pkg=apache2 state=present
      tags: install
    - name: install index page
      copy: src=front.html dest=/var/www/html/index.html backup=yes
      tags: configure
      notify: restart apache2
  handlers:
    - name: restart apache2
      service: name=apache2 state=restarted
```

```
ansible-playbook web.yml -t configure
```

Ansible: Tags Usage

```
ansible-playbook -i production site.yml --limit webservers


ansible-playbook -i production site.yml --tags ntp

ansible-playbook -i acceptance dbservers.yml --tags restore
```

Ansible: Ansible Galaxy

https://galaxy.ansible.com/

Ansible Galaxy refers to the Galaxy website where users can share roles, and a command line tool ***ansible-galaxy*** for installing, creating and managing roles.

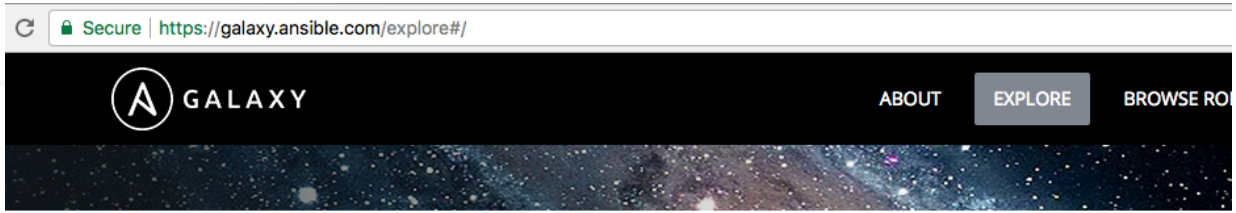
Galaxy, is a free site for finding, downloading, and sharing community developed roles. Downloading roles from Galaxy is a great way to jumpstart your automation projects.

You can also use the site to *share roles that you create*. By authenticating with the site using your *GitHub account*, you're able to import roles, making them available to the Ansible community. Imported roles become available in the Galaxy search index and visible on the site, allowing users to discover and download them.

The ***ansible-galaxy*** command comes bundled with Ansible, and you can use it to install roles from Galaxy or directly from a git based SCM. You can also use it to create a new role, remove roles, or perform tasks on the Galaxy website.

Ansible: How to download roles from Galaxy

Ansible Galaxy provides pre-packaged units of work known to Ansible as roles.
Roles can be dropped into Ansible Playbooks and immediately put to work.
Remember, roles must be downloaded before they can be used in playbooks.
ansible-galaxy install username.rolename



Most Starred

Role	Stars
carlosbuenosvinos.ansistrano-deploy	1272
dev-sec.os-hardening	488
jdauphant.nginx	465
ANXS.postgresql	411
geerlingguy.mysql	397
elastic.elasticsearch	394

Most Watched

Role	Watchers
plone.plone_server	176
elastic.elasticsearch	141
couchbaselabs.couchbase-server	108
cumulus.CumulusLinux	91
ceph.ceph-defaults	86
ceph.ceph-config	73

Most Dow

Role
DavidWitt
Stouts.gra
nickhamm
geerlinggu
geerlinggu
geerlinggu

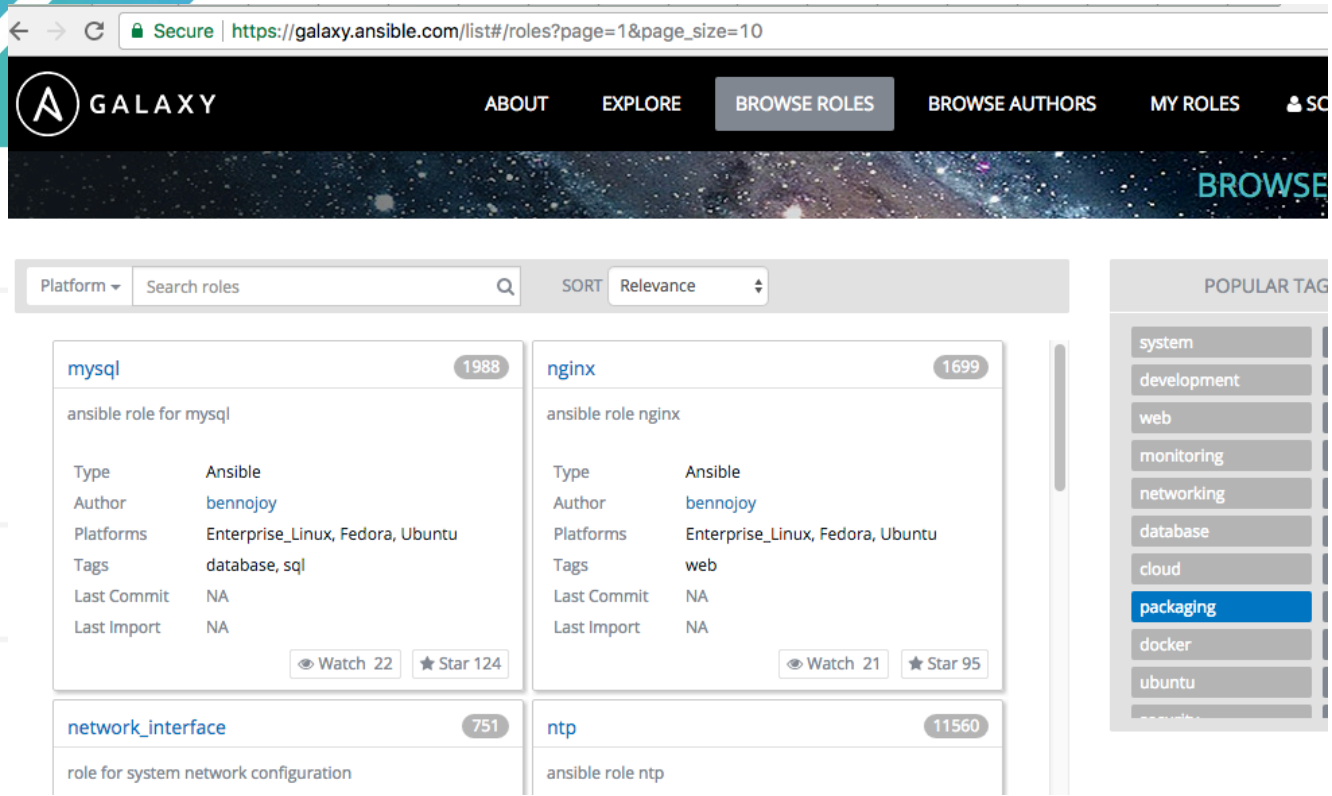
Ansible: How to download roles from Galaxy

```
Sree>tree ansible/
ansible/
├── ansible.cfg
├── hosts
├── playbook.yml
└── roles
```

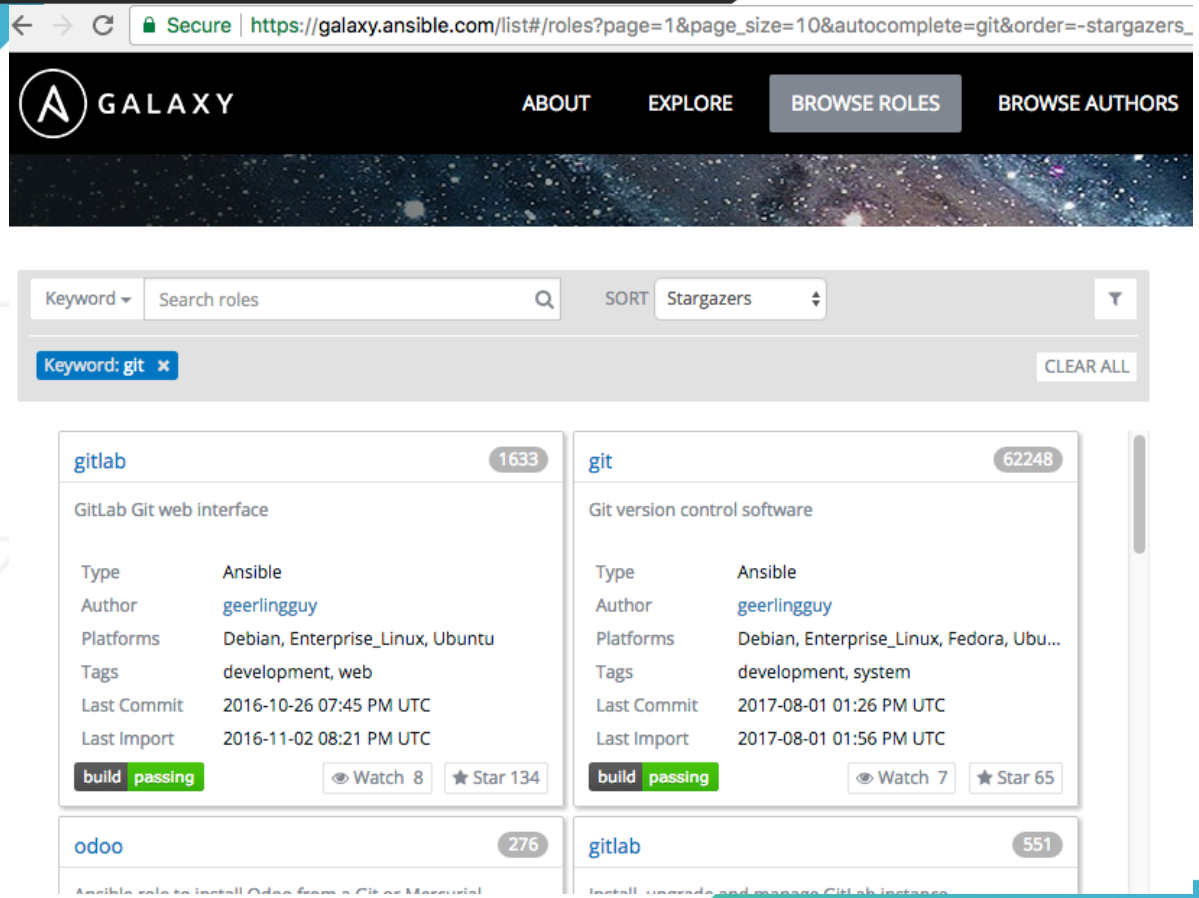
File: ansible/ansible.cfg

```
# additional paths to search for roles in, colon separated
roles_path    = /etc/ansible/roles
```

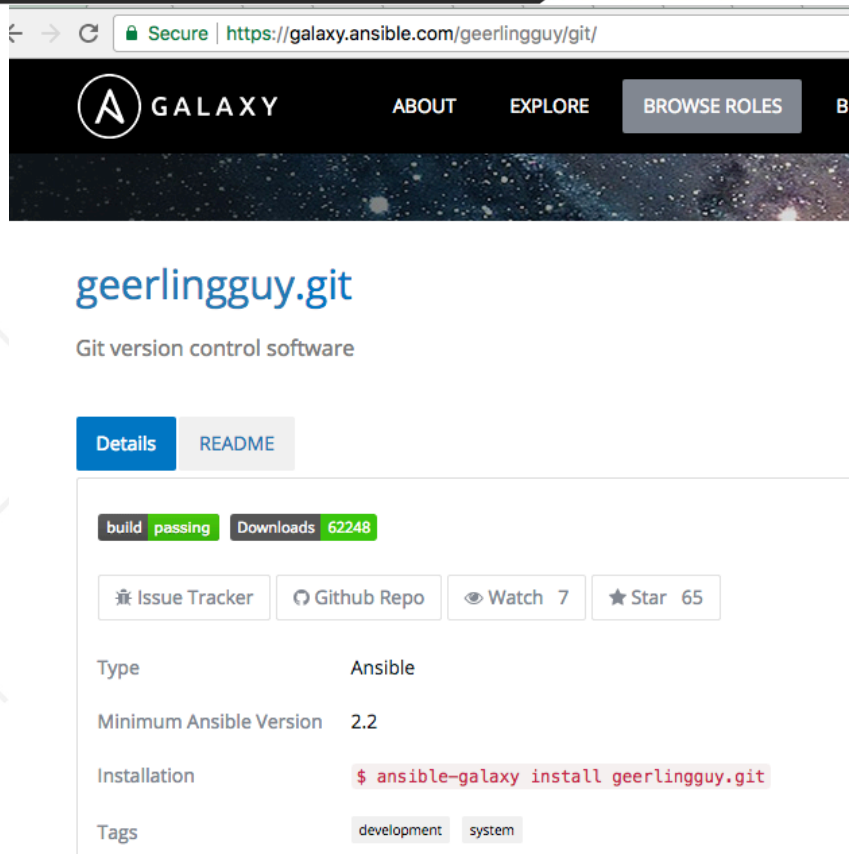
Ansible: How to download roles from Galaxy



Ansible: How to download roles from Galaxy



Ansible: How to download roles from Galaxy



Ansible: How to download roles from Galaxy

```
Sree>docker run --rm -v $PWD/ansible:/etc/ansible schogini/docker-ansible-ws-ubuntu ansible-galaxy install geerlingguy.git
- downloading role 'git', owned by geerlingguy
- downloading role from https://github.com/geerlingguy/ansible-role-git/archive/1.4.0.tar.gz
- extracting geerlingguy.git to /etc/ansible/roles/geerlingguy.git
- geerlingguy.git (1.4.0) was installed successfully
Sree>
```

```
Sree>tree ansible/
ansible/
├── ansible.cfg
├── hosts
├── playbook-nginx.retry
├── playbook-nginx.yml
├── playbook.yml
└── roles
    ├── ansible
    └── geerlingguy.git
        ├── LICENSE
        ├── README.md
        ├── defaults
        │   └── main.yml
        ├── meta
        │   └── main.yml
        ├── tasks
        │   ├── install-from-source.yml
        │   └── main.yml
        ├── tests
        │   ├── README.md
        │   ├── test-source.yml
        │   └── test.yml
        └── vars
            ├── Debian.yml
            ├── Fedora.yml
            ├── RedHat.yml
            └── main.yml

8 directories, 18 files
Sree>
```

Ansible: How to use the role downloaded from Galaxy

```
Sree>nano ansible/playbook-git.yml
```

```
- hosts:
    - web
  roles:
    - geerlingguy.git
```

```
Sree>cat ansible/playbook-git.yml
- hosts:
    - web
  roles:
    - geerlingguy.git
#  roles:
#    - { role: geerlingguy.git  }
#  vars:
#    git_install_from_source: yes
```

```
Sree>docker run --rm -v $PWD/ansible:/etc/ansible schogini/docker-ansible-ws-ubuntu ansible-playbook /etc/ansible/playbook-git.yml
```

Ansible: How to download roles from Galaxy

```
Sree>docker exec -ti node1 bash
root@1b31839769f3:/# git
bash: git: command not found
root@1b31839769f3:/#
```

```
Sree>docker run --rm -v $PWD/ansible:/etc/ansible schogini/docker-ansible-ws-ubuntu ansible-playbook /etc/ansible/playbook-git.yml

PLAY RECAP *********************************************************************
172.17.0.5                 : ok=3    changed=1    unreachable=0    failed=0
172.17.0.6                 : ok=3    changed=1    unreachable=0    failed=0
```

```
root@1b31839769f3:/# git --version
git version 1.9.1
root@1b31839769f3:/#
```

Ansible: A Simple Web Playbook

```
Sree>mkdir -p ansible/tmp
Sree>nano ansible/tmp/simpleweb.html
```

```
<!DOCTYPE html>
<html>
<head>
        <title>{{title}}</title>
</head>
<body>
        <h1>{{title}} from Host {{ansible_host}}</h1>
        <h3>The {{desc}}</h3>
        <hr>
        <h3>I was created by {{who}}</h3>
</body>
</html>
```

Ansible: A Simple Web Playbook

```
Sree>nano ansible/simpleweb_playbook.yml
```

```
- hosts: web
  become: true
  vars:
    who: Sreeprakash
    title: Sree
    desc: Test Description via ansible playbook
  remote_user: root
  tasks:
    - name: install apache2
      apt: name=apache2 update_cache=yes state=latest

    - name: enabled mod_rewrite
      apache2_module: name=rewrite state=present
      notify:
        - restart apache2

    - name: install PHP module for Apache
      apt: name=libapache2-mod-php5 state=present
      notify:
        - restart apache2

    - name: copy simpleweb template to /var/www/html/ folder
      template: src=/etc/ansible/tmp/simpleweb.html dest=/var/www/html/index.html

  handlers:
    - name: restart apache2
      service: name=apache2 state=restarted
```

Ansible: A Web Playbook

```
Sree>docker run --rm -v $PWD/ansible:/etc/ansible schogini/docker-ansible-ws-ubuntu ansible-playbook /etc/ansible/simpleweb_playbook.yml

PLAY [web] *********************************************************************

TASK [Gathering Facts] *********************************************************
ok: [172.17.0.5]
ok: [172.17.0.6]

TASK [install apache2] *********************************************************
ok: [172.17.0.5]
ok: [172.17.0.6]

TASK [enabled mod_rewrite] *****************************************************
changed: [172.17.0.6]
changed: [172.17.0.5]

TASK [install PHP module for Apache] *******************************************
changed: [172.17.0.6]
changed: [172.17.0.5]

TASK [copy simpleweb template to /var/www/html/ folder] ************************
changed: [172.17.0.6]
changed: [172.17.0.5]

RUNNING HANDLER [restart apache2] **********************************************
changed: [172.17.0.6]
changed: [172.17.0.5]
```

Ansible: A Simple Web Playbook

```
Sree>docker inspect --format="{{.NetworkSettings.IPAddress}} {{.NetworkSettings.Ports}}" node1 node2 node3
172.17.0.5 map[22/tcp:[{0.0.0.0 32798}] 80/tcp:[{0.0.0.0 32797}]]
172.17.0.6 map[22/tcp:[{0.0.0.0 32800}] 80/tcp:[{0.0.0.0 32799}]]
172.17.0.7 map[22/tcp:[{0.0.0.0 32802}] 80/tcp:[{0.0.0.0 32801}]]
```

```
Sree>curl localhost:32797
<!DOCTYPE html>
<html>
<head>
        <title>Sree</title>
</head>
<body>
        <h1>Sree from Host 172.17.0.5</h1>
        <h3>The Test Description via ansible playbook</h3>
        <hr>
        <h3>I was created by Sreeprakash</h3>
</body>
</html>
```

Ansible: Extra

1. See what happens behind the scene when run a play (/tmp)
2. Seeing facts

```
Sree>docker run --rm -v $PWD/ansible:/etc/ansible schogini/docker-ansible-ws-ubuntu ansible web -m setup
```

```
Sree>docker run --rm -v $PWD/ansible:/etc/ansible schogini/docker-ansible-ws-ubuntu ansible 172.17.0.5 -m setup
172.17.0.5 | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "172.17.0.5"
        ],
```

Ansible: HW

1. Start 3 Docker Ubuntu Containers with SSHD installed
2. Create a hosts ping and try the ping command
3. Create a playbook and try the ping command on all hosts
4. Experiment with more modules and playbooks
5. Download a Role from Ansible Galaxy and create a playbook and play it

6. **ANSIBLE SOLUTIONS**
    1. SimpleWeb Solution is given in this presentation
    2. https://github.com/schogini/ansible-customeweb-playbook.git


Cognixia
THANK YOU
average 45%