Study Guide

Expertise in Ansible Automation

Contents

Prerequisites

Linux

- You can use 2.6.x or newer kernels

Installation

- On CentOS 7
  - Install the EPEL repository.
  - yum -y install epel-release
  - yum update
  - yum -y install ansible openssl

What is Ansible?

- Ansible is an automated provisioning system for your environments. It doesn't require agents or additional security infrastructure, so it is easy to deploy. You require an SSH connection to the server and the ability to use sudo.
- It uses a simple language (YAML) in playbooks that allow you to describe your automated jobs in a method similar to plain English.

Basic Ansible Commands

- ansible - Ansible ad-hoc commands
- ansible-playbook - Run an Ansible playbook
- ansible-vault - Manage encrypted Ansible vars files
- ansible-galaxy - Manage roles using galaxy.ansible.com
- ansible-doc - Show documentation on Ansible commands
- ansible-pull - Pull playbooks from server

Ansible Core Components

Plays and Playbooks

- An Ansible playbook is made of of individual plays
- A play is a task that is performed
- Playbooks are in YAML format
- Playbook must start with --- at the top, though a comment may also be on the same line

Inventories

- Inventory files contain the list of servers you want to manage and are in an ini format
- Static inventories
  - Inventories can be static or dynamic
  - Static inventories by default are in /etc/ansible/hosts
  - Static inventories can be located anywhere if you use the -i option to include the file. A path should be included. An example is :
    - ansible all -i /home/ansible/myhosts -m ping
  - Can include variable data for use with hosts or groups of hosts
- Dynamic inventories
  - Contain dynamic lists of Ansible hosts
  - The file must be executable or Ansible will expect an ini format
  - Output of executable is expected to be in a JSON format
  - File must provide a list of servers if called with --list
  - File must provide host information if called with --host (HOSTNAME)

Modules

- Modules are used to perform the tasks you require
- Ansible ships with many of the modules you require and they can be used with the Ansible ad-hoc command or through Ansible plays and playbooks
- You can write your own and documentation can be found at:
  - http://docs.ansible.com/ansible/dev_guide/developing_modules.html

Variables

- Allow you to customize behavior for systems, since not all systems are the same
- Variables are how we deal with the differences between systems
- Variable names should be letters, numbers and underscores
- Variables should always start with a letter
- Variables can be defined in the inventory
- Variables can be defined in a playbook or role and can be referenced by templates

Ansible Facts

- Ansible obtains facts through the setup module and is a way of getting data from systems
- They can be used via ad-hoc commands or in playbooks
- Gathering facts are not always required
- Using the gathering_facts: no option can save time

Ansible config file

- Default Ansible config file is locate at /etc/ansible/ansible.cfg. You can use a different Ansible config file if you require
- There is an order of precedence for Ansible config files
  - ANSIBLE_CONFIG - an environment variable with the location
  - ansible.cfg - in your current directory location
  - .ansible.cfg - located in your users home directory
  - /etc/ansible/ansible.cfg

Templates

- Templates are used with variable substitution
- They are processed by the Jinja2 templating system
  - http://jinja.pocoo.org/docs/
- Useful for creating premade config files and then substituting the variables when the playbook runs

Handlers

- Tasks can trigger handlers
- They are used to handle error conditions
- They are called at the end of each play
- Tasks can trigger multiple actions

Roles

- A playbook is a file that Ansible runs against your server, and Roles can be thought of as a playbook that's split up into multiple parts
- The format of a Role is as follows

```
Roles/
└── apache
    ├── defaults
    │   └── main.yml
    ├── files
    ├── handlers
    │   └── main.yml
    ├── meta
    │   └── main.yml
    ├── README.md
    ├── tasks
    │   └── main.yml
    ├── templates ├──
    tests
    │   ├── inventory
    │   └── test.yml
    └── vars
        └── main.yml
```


- In the example above, the role itself is called apache and it sits in a folder called Roles.
- The folders located below apache are where files, handlers, meta data, templates and variables should be located. Ansible expects the required portions of the playbook to be inside the main.yml files.
- A playbook using that role would look like the following:

```
---
- hosts: local
  become: yes roles:
    - Roles/apache
```

ansible-vault

- ansible-vault is an encrypted store
- It's used for storing variables or passwords or files in an encrypted format
- It uses an AES-256 cipher
- The command line tool, ansible-vault, is used to work on the files
- When using an encrypted file in a playbook you need to use the following options when running the playbook:
  - --ask-vault-pass
  - --vault-password-file

Repositories

Static Repositories

- Static repositories hold the information on which hosts are being managed by Ansible and under which groups they belong.
- This information is held, by default, in the /etc/ansible/hosts file
- You can use your own file as a repository, and select it with the -i option when a command such as the ad-hoc command is run

Example Hosts File Used for Repositories

```
[local]
localhost
[labservers]
server1.mylabserver.com
server2.mylabserver.com maxRequestsPerChild=100
server3.mylabserver.com
[webserver-group]
www[01:11].devopsacademy.com
```

- The portions with the [ ] are defining the group name of those servers. So the servers under the [labservers] are in the labserver group and would be used with an ad-hoc command viasomething like the following:
  - ansible labservers -m ping
- The servers defined by www[01:11] are a way of selecting multiple servers without typing in their names. This will select put all servers with the hostnames of www01.linuxacademy.com to www11. qualitythough.com into the webserver-group

- Variables can be defined in repositories and an example is seen above with maxRequestsPerChild=100
- Variables can also be defined for groups of servers as well as individual servers

Dynamic Repositories

- Dynamic repositories allow you to pull inventories via a more dynamic process than allowed for with a ini based file
- Many cloud platforms are supported in Ansible and those providers have instructions on whats required to use their service with Ansible
- The output provided from the executable file that links you with the providers system must be in
  JSON format.
- More information on dynamic inventories can be found on the Ansible page.
- http://docs.ansible.com/ansible/intro_dynamic_inventory.html

Modules

About Modules

- Modules are what makes Ansible powerful. Modules control systems and perform the actions or tasks you need
- Modules are what perform the actual work in Ansible and are what gets run with playbooks or ad-hoc tasks
- Most modules require arguments
- Arguments used in modules are generally in a key=value format that is space delimited
- Some modules take no arguments e.g the shell module takes a string of the command you want to run
- To find out the information about what a module needs to run, view this URL: http://docs.ansible.com/ansible/modules_by_category.html
- You can write your own modules and the documentation for that can be found at this URL: http://docs.ansible.com/ansible/dev_guide/developing_modules.html

Ansible Ad-hoc Commands

About Ad-hoc Commands

- Ansible ad-hoc commands are useful for quick tasks you need to get done.
- They are a great place to get started with Ansible if you're not familiar with it
- Format of an ad-hoc command is ansible <host-group> [options]
- An example of an ad-hoc command to install php on all servers in the group called centos would be the following:
  - ansible centos -b -m yum -a 'name=php state=latest'
- Normally a command will run on the target server as the user who runs it. You may not have the authority if the command you run needs sudo or root access
- You would use the -b option, as shown above, to 'become' root. The user needs to be able to sudo to root for this to be effective
- Ansible ad-hoc uses Ansible modules to perform the tasks

Sample Ad-hoc Commands

- These commands do a variety of tasks:
  - ansible all -m ping
    - Check connectivity to the servers
  - ansible local -m setup -a 'filter=ansible_default_ipv4'
    - Uses the setup module to pull information about the server, then filter it to only the sectionansible_default_ipv4
  - ansible centos -b -m yum -a 'name=httpd state=latest'
    - Installs the latest Apache webserver on all servers in the centos group
  - ansible webhosts -i myhosts -b -m yum -a "name=elinks state=latest"
    - Installs elinks onto hosts in the webhosts group that are in the myhosts inventory file

Plays and Playbooks

About Plays and Playbooks

- Plays are the individual tasks that are performed inside a playbook and a playbook is made up of one or more plays
- Playbooks describe a set of steps in a process
- They can describe a policy you want to enforce
- They force a specific end state to your systems
- They are designed to be human readable
- They can be broken out to roles and templates
- Playbooks are more efficient for multiple tasks
- Easier to put under source control than ad-hoc tasks
- Playbooks are written in YAML format and use a minimum of syntax
- Uses standard YAML but without the metadata at the start. Because of this, you define the start of the YAMl file with 3 dashes on the first line like this:
  - ---
- Playbooks should be idempotent. So you should be able to rerun them multiple times without problems. For instance, if a file is going to be overwritten and cause problems you should check first and not change it

An Example Playbook That Installs Apache

```
---
- hosts: local
  become: yes
  tasks:
  - name: install apache
    yum: name=httpd state=latest
```

- The playbook above first restricts the actions to the servers in the local group
- Then it uses `become: yes` to perform the actions on the target server as the root user
- Then it sets up the tasks that are required to be performed with the tasks
- Then it use `name:` to call the play "install apache"
- Then it uses the yum module and passes the required parameters of `name=httpd` (which is the

Apache package on Red Hat) and state=latest. There are several different choices of state.

Another Example of a Playbook That Performs Several Actions

```
---
-   hosts: databases tasks:

        -   shell: cat /etc/motd
            register: motd_contents
        -   debug: msg="stdout={{motd_contents}}"
        -   debug: msg="MOTD is EMPTY"
            when: motd_contents.stdout == ""
```

- This play book runs against the hosts in the databases section of the inventory
- It has several tasks.
  - Its first task is the shell module that puts the contents of the /etc/motd file into the register called motd_contents. A register is a way that Ansible can store the responses to an action that has been performed
  - Then it uses the debug module that displays what the shell command sent to the stdout and shows that on the info sent back by the running playbook
  - The next debug command will echo to the responses from the running playbook, MOTD is EMPTY, only when the modt_contents is empty
- Here is what the response from running that playbook looks like:

```
[ansible@server roles]$ ansible-playbook check-motd.yml
PLAY [local] ********************************************************
*********
TASK [setup] ********************************************************
*********
ok: [localhost]
TASK [command] ******************************************************
*********
changed: [localhost]
TASK [debug] ********************************************************
*********
ok: [localhost] => {
        "msg": "stdout={u'changed': True, u'end': u'2017-03-29
11:06:24.451976', u'stdout': u'', u'cmd': u'cat /etc/motd', u'rc': 0,
u'start': u'2017-03-29 11:06:24.448336', u'stderr': u'', u'delta':
u'0:00:00.003640', 'stdout_lines': [], u'warnings': []}"
}
TASK [debug] ********************************************************
*********
ok: [localhost] => {
        "msg": "MOTD is EMPTY"
}
```

```
PLAY RECAP ********************************************************** *********

localhost : ok=4 changed=1 unreachable=0 failed=0
```

Templates

What are Ansible Templates?

- Templates use the template module. The module can take variables that you have defined and replace those in files. The use is to replace the information and then send that information to the target server.
- Templates are processed by the Jinja2 templating language. Documentation about this language can be found here: http://jinja.pocoo.org/docs

An Example of a Template and Playbook

- Here is what is in the template file called template.j2.

```
<p>
Hello there <p>
ServerName = {{description}}
```

- Here is a sample playbook that uses that template:

```
---
- hosts: databases
  become: yes
  vars:
     description: "{{ ansible_hostname }}" tasks:

  - name: write the index file
    template: src=template.j2 dest=/var/www/html/index.html notify:

    - restart httpd
  - name: ensure apache is running
    service: name=httpd state=running
  handlers:
    - name: restart httpd
      service: name=httpd state=restarted
```

- Here is the contents of the /var/www/html/index.html file once the playbook has run:

```
<p>
Hello there <p>
ServerName = server
```

- For this particular server, the hostname is 'server'

Roles

What are Roles in Ansible

- Roles in Ansible use the idea of using include files and combines them to form reusable sections
- It allows you to reuse portions of your code easier. You break up the playbook into sections and when the playbook is run it pulls all the sections together and runs against your target hosts
- Ansible roles must be in a particular format to work as expected. You need a folder and subfolders to be in a specified format.
- As an example, if we create a folder called Roles and we want to store our roles in there then we would create the fodler for the project and its subfolders as needed. The `ansible-galaxy` command can be used to create the correct format as shown below
- Starting in the Roles directory. The command `ansible-galaxy init apache` will create the following tree and files:

```
apache/
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
├── templates ├──
tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

- We would edit the files as required for the portions that our project needs. For instance, we would edit the `apache/tasks/main.yml` file to put in the tasks that are required. We would edit the `apache/vars/main.yml` to put in any variables that are needed and so on.
- If you don't need a section then it's not used. So, for instance, if we put no data into `handlers/main.yml`, then it would be ignored when the role is run

An Example of a Role in Ansible

- Here is an example of a role in Ansible. The only file we require is tasks since this is a simple

example. Here is the file system tree with only the files that are needed:

```
.
└── apache
    └── tasks
        └── main.yml
```

- Here is the contents of main.yml

```
- yum: name=httpd state=present
```

- Here is the contents of the playbook that will use the role:

```
---
- hosts: local
  become: yes roles:
    - Roles/apache
```

- Here is the command that's run:

```
ansible-playbook playbook.yml
```

- When the playbook is run it includes the tasks in apache/tasks/main.yml and runs them

Ansible Galaxy

What is Ansible Galaxy

- Ansible Galaxy is a website where users can share roles.
- It also refers to the command line tool that is installed with Ansible.
- The command line tool is called with the following format
  - ansible-galaxy [delete|import|info|init|install|list|login|remove|search|setup] [-- help] [options] ...

- By default, roles are downloaded to the /etc/ansible/roles folder. If you want to store them there you may need to preface the command with sudo
- You can change where the role is installed by useing the -p option when you use the command
- Roles can have dependancies, but those will automatically be installed
- You don't need an Ansible Galaxy profile to download. If you wish to contribute roles, then you will need a profile on the site.

- The URL for the Ansible Galaxy site is https://galaxy.ansible.com/

An Example of Using Ansible Galaxy

- To use Ansible Galaxy you need to find a role you wish to download
- You can use ansible-galaxy search to search for roles or you can search from the Ansible Galaxy website
- To install a role from Ansible Galaxy you specify the download option
- Here is an example:
  - ansible-galaxy install bennojoy.nginx -p Roles
    - The output of that command is shown below:

```
- downloading role 'nginx', owned by bennojoy
- downloading role from https://github.com/bennojoy/nginx/archive/ master.tar.gz

- extracting bennojoy.nginx to Roles/bennojoy.nginx
- bennojoy.nginx was installed successfully
```

- The command installed the role into the Roles folder. Here is the tree to show you the format:

```
└── Roles
    ├── bennojoy.nginx
    │   ├── defaults
    │   │   └── main.yml
    │   ├── files
    │   │   └── epel.repo
    │   ├── handlers
    │   │   └── main.yml
    │   ├── meta
    │   │   └── main.yml
    │   ├── README.md
    │   ├── tasks
    │   │   └── main.yml
    │   ├── templates
    │   │   ├── default.conf.j2
    │   │   ├── default.j2
    │   │   ├── nginx.conf.j2
    │   │   └── site.j2
    │   └── vars
    │       └── main.yml
```

- You would use the role in a playbook the same as a normal role.

Parallelism

What is Parallelism in Ansible?

- It's how many processes that Ansible uses to talk to the server to perform its tasks. By default, it's 5, but that can be changed
- You can change it in the config file, on the command line or in a playbook.

How to Change It

- Ansible calls its forks; here is an example of the setting being changed to 20:
- In a config file:
  - forks = 20
- On the command line with an ad-hoc command:
  - ansible centos -m ping -f 20
- In a playbook:

```
---
- hosts: ec2
  serial: 20
```

Ansible Vault

What is Ansible Vault?

- Ansible Vault is an encrypted store
- It's used for storing variables or passwords or files in an encrypted format
- It uses an AES-256 cipher
- The command line tool, ansible-vault, is used to work on the files
- When using an encrypted file in a playbook, you need to use the following options when running the playbook:
  - --ask-vault-pass
  - --vault-password-file

Ansible Vault Commands

- Here is an example of using vault to encrypt a file

```
ansible-vault encrypt Roles/apache-install/vars/main.yml
Vault password:
Encryption successful
```

How You Use It

- When you call a play that has an ecrypted file in it, you need to let Ansible know to ask for the decryption key. If you don't, then it will fail, as shown below:

```
ansible-playbook testplay1.yml
ERROR! Decryption failed on /home/ansible/roles/Roles/apache-install/ vars/main.yml
```

- To run the playbook and decrypt the encrypted file you would use the following:
  - ansible-playbook testplay1.yml --ask-vault-pass

Ansible Tower

What is Ansible Tower?

- Ansible Tower is a web-based solution that is designed to help you manage your Ansible installation.
- Ansible Tower provides access control over your playbooks, inventory, SSH credentials. It can manage who has access to those credentials. It has logging that helps you monitor your systems.
- Find out more about Tower features and how to download it on the Ansible Tower webpage. Tower is free for usage for up to 10 nodes.
- https://ansible.com/tower

Links to Other Resources

- Developing your own modules:
  http://docs.ansible.com/ansible/dev_guide/developing_modules.html
- Jinja2 templating system: http://jinja.pocoo.org/docs/
- Ansible website about dynamic inventories:
  http://docs.ansible.com/ansible/intro_dynamic_inventory.html
- Information about default modules:
  http://docs.ansible.com/ansible/modules_by_category.html
- Write your own modules:
  http://docs.ansible.com/ansible/dev_guide/developing_modules.html