Ansible:
Debug, optimization, vault

Marica Antonacci - INFN BARI
marica.antonacci@ba.infn.it

Corso Ansible/Foreman/Puppet
5-8 June 2018, INFN BARI

Debugging

The verbose flag

When running Ansible with verbose (-vvv or --verbose), it prints out all the values that were returned by each module after it runs.

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

The debug module

It's Ansible's version of a `print` statement.

You can use it to print out either the value of a variable or an arbitrary string.

```
- debug: var=myvariable

- debug: msg="The value of myvariable is {{ var }}"
```

The assert module

The assert module will fail with an error if a specified condition is not met.

For example, to fail the playbook if there's no eth1 interface:"

```
- name: assert that eth1 interface exists

  assert:

    that: ansible_eth1 is defined
```

The pause module

Another technique is to use the `pause` module to pause the playbook while you examine the configured machine as it runs. This way, you can see changes that the modules have made at the current position in the play, and then watch while it continues with the rest of the play.

Syntax check

The --syntax-check flag will check that your playbook's syntax is valid, but it will not execute it.

List tasks

The --list-tasks flag will output the tasks that the playbook will run against. It will not execute the playbook.

```
playbook: site.yml

  play #1 (db): db      TAGS: []
    tasks:
      mysql : Install MySQL packages    TAGS: []
      mysql : Create datadir if it does not exist       TAGS: []
      mysql : Ensure MySQL is started and enabled on boot.      TAGS: []
      mysql : Set root Password TAGS: []
      mysql : Copy my.cnf       TAGS: []
      mysql : Remove all anonymous user accounts        TAGS: []
      mysql : Remove the MySQL test database    TAGS: []
      mysql : Create MySQL database     TAGS: []
      mysql : Create user       TAGS: []

  play #2 (web): web    TAGS: []
    tasks:
      apache : Install httpd Package    TAGS: []
      apache : Start and Enable httpd service   TAGS: []
      phpmyadmin : Add repo file        TAGS: []
      phpmyadmin : Add repo key TAGS: []
      phpmyadmin : Install phpMyAdmin   TAGS: []
      phpmyadmin : Install php  TAGS: []
      phpmyadmin : Add default username and password for MySQL connection.      TAGS: []
      phpmyadmin : Grant access to /phpmyadmin  TAGS: []
```

Check mode

The -C and --check flags will run Ansible in check mode (sometimes known as dry-run), which tells you whether each task in the playbook would modify the host, but does not make any actual changes to the server.

Diff (show file changes)

The -D and --diff flags will output differences for any files that will be changed on the remote machine. It's a helpful option to use in conjunction with --check to show how Ansible would change the file if it were run normally.

```
TASK [phpmyadmin : Grant access to /phpmyadmin] ******************************
--- before: /etc/httpd/conf.d/phpMyAdmin.conf (content)
+++ after: /etc/httpd/conf.d/phpMyAdmin.conf (content)
@@ -14,6 +14,7 @@
    <IfModule mod_authz_core.c>
      # Apache 2.4
      <RequireAny>
+       Require all granted
        Require ip 127.0.0.1
        Require ip ::1
      </RequireAny>

changed: [172.21.0.50]
```

Playbook debugger - new in Ansible 2.5

The debugger plugin allows you to debug a task.

You can check or set the value of variables, update module arguments, and re-run the task with the new variables and arguments to help resolve the cause of the failure.

`debugger` keyword

```
- name: Execute a command
  command: false
  debugger: on_failed
```

```
- name: Play
  hosts: all
  debugger: on_skipped
  tasks:
    - name: Execute a command
      command: true
      when: False
```

Commands:

`p(print) task/task_vars/host/result`

`task.args[key] = value`

`r(edo)`

`task_vars[key] = value`

`c(ontinue)`

`q(uit)`

Ansible Vault

Managing secrets

Playbooks often require access to sensitive information, e.g. database and administrator passwords, private keys, etc.

Ansible Vault provides the ability to secure any sensitive data that is necessary to successfully run Ansible plays but should not be publicly visible.

Ansible automatically decrypts vault-encrypted content at runtime when the key is provided.

ansible-vault utility

Vault allows encrypted content to be incorporated transparently into Ansible workflows.

`Ansible-vault` is the utility for encrypting sensitive data on disk.

To integrate these secrets with regular Ansible data, both the `ansible` and `ansible-playbook` commands, for executing ad hoc tasks and structured playbook respectively, have support for decrypting vault-encrypted content at runtime.

ansible-vault encrypt

A typical use of Ansible Vault is to encrypt variable files.

Encrypt an existing file:

```
ansible-vault encrypt defaults/main.yml
```

Create an encrypted file:

```
ansible-vault create defaults/extra.yml
```

These commands will prompt you for a password twice (a second time to confirm the first).

ansible-vault encrypt (cont.)

Encrypt specific variables:

```
ansible-vault encrypt_string -n mysql_user_password 's3cret'
```

```
mysql_user_password: !vault |
          $ANSIBLE_VAULT;1.1;AES256
31343861363466316435343466333562666562393036646365663034346363303633633262356338
33373531386237613831326535326164613132323666236 0a32373639623630326530313530653 2
65396465633534643732323230366538363133666166343966663430663362613765366230393934
31316232376630383 10a6463623563613666386430376439633462343739326537303166373030 64
6437
```

Running ansible with encrypted variables

Ask for Vault password:

```
ansible-playbook --ask-vault-pass -i inventory_file
some_playabook.yml
```

Use a Vault file:

```
echo "secret_password" > vault_password
```

```
ansible-playbook --vault-password-file=vault_password -i
inventory_file some_playabook.yml
```

ansible-vault - other commands

Edit an encrypted file:

```
ansible-vault edit defaults/main.yml
```

Decrypt a file:

```
ansible-vault decrypt defaults/main.yml
```

View an encrypted file (read-only):

```
ansible-vault view defaults/main.yml
```

ansible-vault - other commands

Change the password of encrypted files:

```
ansible-vault rekey defaults/main.yml
```

You will first be prompted with the file's current password and then you will be asked to enter and confirm a new vault password.

Finally you will receive a message indicating success of the re-encryption process.

Notes

If you lose your password, you lose your data → use a shared password manager

If you're managing your Ansible roles with git, you'll notice that even opening an encrypted file (and not changing it) will change the file requiring a new git commit → this can be mitigated by encrypting only specific variables within a file as shown above

Optimizations

SSH multiplexing and ControlPersist

When Ansible runs a playbook, it will make many SSH connections, in order to do things such as copy over files and run commands.

Each time Ansible makes a new SSH connection to a host, it has to pay the negotiation penalty.

OpenSSH supports an optimization called **SSH multiplexing**, which is also referred to as ControlPersist: a master connection is opened for each host and a control socket is used to communicate with the remote host instead of making a new TCP connection.

SSH Multiplexing options in Ansible

`ControlMaster`	default=auto
`ControlPath`	default=$HOME/.ansible/cp/ansible-ssh-%h-%p-%r
`ControlPersist`	60s

Pipelining

When Ansible executes a task

- It generates a Python script based on the module being invoked
- Then it copies the Python script to the host
- Finally, it executes the Python script

Ansible supports an optimization called **pipelining**, where it will execute the Python script by piping it to the SSH session instead of copying it. This saves time because it tells Ansible to use one SSH session instead of two.

To enable pipelining, set `pipelining=true` in the [`default`] section of *ansible.cfg*
Moreover requiretty must be disabled in your /etc/sudoers

Facts caching

If you enable facts caching, Ansible will gather facts for a host only once, even if you rerun the playbook or run a different playbook that connects to the same host.

There are several cache plugins:

```
ansible-doc -t cache -l
```

jsonfile JSON formatted files

memcached Use memcached DB for cache

memory RAM backed, non persistent

mongodb Use MongoDB for caching

pickle Pickle formatted files.

redis Use Redis DB for cache

yaml YAML formatted files.

Enable facts caching

Edit *ansible.cfg*:

```
[defaults]
gathering = smart
fact_caching_timeout = 86400
fact_caching = ...
```

Ansible will only gather facts if they are not present in the cache or if the cache has expired.

Parallelism

For each task, Ansible will connect to the hosts in parallel to execute the tasks. But Ansible doesn't necessarily connect to all of the hosts in parallel. Instead, the level of parallelism is controlled by a parameter, which defaults to 5.

You can modify the Ansible configuration file (ansible.cfg) by setting a `forks` option in the defaults section.