ansible-flow Documentation

Release 0.2.0

John Vrbanac

July 12, 2016

Contents

Ansible-flow is a simple utility to help make ansible easier to use with a specific set of production use-cases.

If you need to do the following, then ansible-flow might be for you:

- Use the same playbooks against multiple environments
- Use bastions in your deployment
- Run a collection of playbooks in sequence.

CHAPTER 1

Contents:

1.1 Installing ansible-flow

`ansible-flow` is available on PyPI and can be installed using:

```
pip install ansible-flow
```

1.2 Configuration

ansible-flow looks for a `project.yml` file within the executing directory.

Each `project.yml` file contains three different sections:

- requirements
- environments
- targets

1.2.1 Requirements

The requirements section is where you define any python requirements (in a list) that you will need to execute your ansible playbooks. For simple use-cases you should only need to define `ansible` or `ansible==2.1.0`.

Example:

```
requirements:
    - ansible==2.1.0
```

1.2.2 Environments

The environments section is where you define specific for custom variable files, vault keys, and ansible configuration files on a per environment basis.

If you have a number of common values that you'd like to share across environments, then you can specify a `default` environment. If a `default` environment is specified then all other environments will just layer their settings on-top of the `default`.

Per Environment Options

- `vault-key`: Path to your vault password file
- `custom-var-files`: A list of YAML files to load when executing your playbooks. You can also use file globs in each entry.
- `directory`: A base directory for your `custom-var-files`
- `ansible-config`: Path to a ansible configuration file
- `shell-variables`: Key value pairs of environmental variables to be used when executing ansible

Example

A environments section containing two different environments each with a file encrypted with two different vault keys.

Directory Structure

```
envs/dev/
    - general.yml
    - auth.vault.yml
envs/test/
    - general.yml
    - auth.vault.yml
dev-vault-key
test-vault-key
project.yml
```

Environments section of `project.yml`

```
environments:
    default:
        custom-var-files:
            - general.yml
            - auth.vault.yml
    dev:
        directory: ./envs/dev
        vault-key: ./dev-vault-key
    test:
        directory: ./envs/test
        vault-key: ./test-vault-key
```

1.2.3 Targets

The targets section allows for you to define a set of playbooks to be executed.

Per Target Options

- `playbooks`: A list of playbooks to be executed in sequential order.
- `inventory`: The inventory script or ini file you wish to use.
- `tags`: Tags you wish to pass to ansible
- `ansible-options`: Custom cli arguments for ansible-playbook

Example

```
targets:
    ping:
        playbooks:
            - ping.yml
        inventory: ./inventory.ini
    bootstrap:
        playbooks:
            - bootstrap.yml
            - 2fa.yml
        inventory: ./inventory.ini
```

1.2.4 Example Configuration

```
---
requirements:
    - ansible==2.1.0

environments:
    default:
        custom-var-files:
            - general.yml
            - auth.vault.yml
    dev:
        directory: ./envs/dev
        vault-key: ./dev-vault-key
    test:
        directory: ./envs/test
        vault-key: ./test-vault-key

targets:
    ping:
        playbooks:
            - ping.yml
        inventory: ./inventory.ini
    bootstrap:
        playbooks:
            - bootstrap.yml
            - 2fa.yml
        inventory: ./inventory.ini
```

1.3 Using ansible-flow

1.3.1 Project virtual environment for ansible-flow

`ansible-flow` will execute all of its actions under a virtual environment. This allows for you to pin specific versions ansible and any other dependencies. `ansible-flow` can maintain the virtual environment for you using the venv sub-command.

Note: The packages installed into the virtual environment are defined in the requirements section of your `project.yml`

Working with ansible-flow's venv sub-command

```
# Create a fresh virtual environment
ansible-flow venv create

# Recreates the virtual environment (commonly used when you change dependencies)
ansible-flow venv recreate

# Completely deletes the virtual environment
ansible-flow venv delete
```

1.3.2 Running ansible-flow

Assuming you've written your `project.yml` configuration, you can execute a target against a given environment using the following command:

```
ansible-flow run ping --env dev
```

Note: ansible-flow is not affiliated with the ansible project in any way.

CHAPTER 2

Links:

- GitHub: https://github.com/jmvrbanac/ansible-flow

CHAPTER 3

Indices and tables

- genindex
- modindex
- search