Ansible Loops

What are Loops?

- Ansible loop is used to repeat any task or a part of code multiple times in an Ansible-playbook.
- Often you want to do many things in one task, such as create a lot of users, install a lot of packages, or repeat a polling step until a certain result is reached.

Types of Loops (Common)

- Standard Loops
- Nested Loops
- Looping over Hashes
- Looping over Files
- Looping over Fileglobs
- Do-Until Loops

Standard Loops

- To save some typing, repeated tasks can be written in short-hand like so:

```
- name: add several users
  user:
    name: "{{ item }}"
    state: present
    groups: "wheel"
  with_items:
     - testuser1
     - testuser2
```

Nested Loops

- Loops can be nested as well:

```
- name: Give users access to multiple databases
  mysql_user:
    name: "{{ item[0] }}"
    priv: "{{ item[1] }}.*:ALL"
    append_privs: true
    password: "foo"
  with_nested:
    - [ 'alice', 'bob']
    - [ 'clientdb', 'employeedb', 'providerdb']
```

Looping over Hashes

- Suppose you have the following variable:

```yaml
---
users:
  alice:
    name: Alice Appleworth
    telephone: 123-456-7890
  bob:
    name: Bob Bananarama
    telephone: 987-654-3210
```

Looping over Files

- **with_file** iterates over the content of a list of files, item will be set to the content of each file in sequence. It can be used like this:

```
---
- hosts: all
  tasks:
    # emit a debug message containing the content of each file.
    - debug:
        msg: "{{ item }}"
      with_file:
        - first_example_file
        - second_example_file
```

Looping over Fileglobs

- **with_fileglob** matches all files in a single directory, non-recursively, that match a pattern.
- It calls **Python's glob library**, and can be used like this:

```
---
- hosts: all

  tasks:

    # first ensure our target diectory exists
    - name: Ensure target directory exists
      file:
        dest: "/etc/fooapp"
        state: directory
    # copy each file over that matches the given pattern
    - name: Copy each file over that matches the given pattern
      copy:
        src: "{{ item }}"
        dest: "/etc/fooapp/"
        owner: "root"
        mode: 0600
      with_fileglob:
        - "/playbooks/files/fooapp/*"
```

Do-Until Loops

- Sometimes you would want to retry a task until a certain condition is met. So we use do-until loops.

Example:

```
- shell: /usr/bin/foo
  register: result
  until: result.stdout.find("all systems go") != -1
  retries: 5
  delay: 10
```

Types of Loops (More…)

- Looping over File trees
- Looping over Parallel Sets of Data
- Looping over Sub elements
- Looping over Integer Sequences
- Random Choices
- Loops and Includes in 2.0
- Writing Your Own Iterators
- Looping Over A List With An Index
- Using in file with a loop
- Flattening A List
- Using register with a loop
- Looping over the inventory
- Loop Control