Ansible Curriculum

Ansible Basic Ideas

- Ansible is idempotent
    - An operation is idempotent if the result of performing it once is exactly the same as the result of performing it repeatedly without any intervening actions.
- Ansible Glossary

Inventory

- File locations
    - Default: /etc/ansible/hosts
    - Can specify with -i <path>
    - Can use multiple at the same time
- Can create dynamic inventory files
- Basics
    - Formats
        - Usually uses INI or YAML formatting
        - INI Example

```
mail.example.com

[webservers]
foo.example.com
bar.example.com

[dbservers]
one.example.com
two.example.com
three.example.com
```

            - Heading in brackets are group names
            - Everything not in brackets are hosts
        - YAML Example

```
all:
  hosts:
    mail.example.com:
  children:
    webservers:
      hosts:
        foo.example.com:
        bar.example.com:
    dbservers:
      hosts:
        one.example.com:
        two.example.com:
        three.example.com:
```

            - Headings one level under *children* are group names
            - Everything under *hosts* are hosts

- Hosts
  - Can put hosts in more than one group. Example below

```
all:
  hosts:
    mail.example.com:
  children:
    webservers:
      hosts:
        foo.example.com:
        bar.example.com:
    dbservers:
      hosts:
        one.example.com:
        two.example.com:
        three.example.com:
    east:
      hosts:
        foo.example.com:
        one.example.com:
        two.example.com:
    west:
      hosts:
        bar.example.com:
        three.example.com:
    prod:
      hosts:
        foo.example.com:
        one.example.com:
        two.example.com:
    test:
      hosts:
        bar.example.com:
        three.example.com:
```

  - Can add a range of hosts with or without a stride (increments between sequence numbers). You can also use letters instead of numbers. See examples below

In INI:

```
[webservers]
www[01:50].example.com
```

In YAML:

```
...
  webservers:
    hosts:
      www[01:50].example.com:
```

In INI:

```
[webservers]
www[01:50:2].example.com
```

In YAML:

```
...
  webservers:
    hosts:
      www[01:50:2].example.com:
```

```
[databases]
db-[a:f].example.com
```

- Adding variables to hosts

```
[atlanta]
host1 http_port=80 maxRequestsPerChild=808
host2 http_port=303 maxRequestsPerChild=909
```

In YAML:

```yaml
atlanta:
  hosts:
    host1:
      http_port: 80
      maxRequestsPerChild: 808
    host2:
      http_port: 303
      maxRequestsPerChild: 909
```

- Groups
    - Default groups
        - All: contains every hosts
        - Ungrouped: contains all hosts that don’t have another group aside from the group all
    - Can nest groups. Example below

```
all:
  hosts:
    mail.example.com:
  children:
    webservers:
      hosts:
        foo.example.com:
        bar.example.com:
    dbservers:
      hosts:
        one.example.com:
        two.example.com:
        three.example.com:
    east:
      hosts:
        foo.example.com:
        one.example.com:
        two.example.com:
    west:
      hosts:
        bar.example.com:
        three.example.com:
    prod:
      children:
        east:
    test:
      children:
        west:
```

- Adding variables to groups

```
[atlanta]
host1
host2

[atlanta:vars]
ntp_server=ntp.atlanta.example.com
proxy=proxy.atlanta.example.com
```

n YAML:

```
atlanta:
  hosts:
    host1:
    host2:
  vars:
    ntp_server: ntp.atlanta.example.com
    proxy: proxy.atlanta.example.com
```

        - Note that you may have a conflict if a host belongs to two groups and both of the groups have the same group variable set
    - Adding variables to nested groups

In INI:

```
[atlanta]
host1
host2

[raleigh]
host2
host3

[southeast:children]
atlanta
raleigh

[southeast:vars]
some_server=foo.southeast.example.com
halon_system_timeout=30
self_destruct_countdown=60
escape_pods=2

[usa:children]
southeast
northeast
southwest
northwest
```

In YAML:

```
all:
  children:
    usa:
      children:
        southeast:
          children:
            atlanta:
              hosts:
                host1:
                host2:
            raleigh:
              hosts:
                host2:
                host3:
          vars:
            some_server: foo.southeast.example.com
            halon_system_timeout: 30
            self_destruct_countdown: 60
            escape_pods: 2
        northeast:
        northwest:
        southwest:
```

- Host and group vars files

    - Ansible will look for the directories *host_vars* and *group_vars* relative to your inventory file. For example, if your inventory file is located at /etc/ansible/hosts and you have the group west and host web in your inventory file ansible will look in /etc/ansible/group_vars and /etc/ansible/host_vars for the files west and web respectively.
    - It is better to have the directories host_vars and group_vars in your playbook directory as well as your inventory file. This way everything is self contained in one place

Ad Hoc Commands

- Messing around with different groups
    - Ping all linux servers: ***ansible linux -m ping -i inventory.yml***
    - Ping only webservers: ***ansible webservers -m ping -i inventory.yml***
    - Ping only database servers: ***ansible database_servers -m ping -i inventory.yml***
- Messing around with different modules
    - Get hostnames of all linux machines: ***ansible linux -m command -a "hostname" -i inventory.yml***
    - Get available disk space of all linux machines: ***ansible linux -m command -a "df -h" -i inventory.yml***
    - Get memory stats of all linux machines: ***ansible linux -m command -a "free -h" -i inventory.yml***
    - Install htop across all linux machines: ***ansible linux -m package -a "name=htop state=present" -i inventory.yml -b --ask-become-pass***
    - Install nginx across all webservers: ***ansible webservers -m package -a "name=nginx state=present" -i inventory.yml -b --ask-become-pass***
    - Enable nginx with systemd across all webservers: ***ansible webservers -m systemd -a "name=nginx enabled=yes state=started" -i inventory.yml -b --ask-become-pass***

How to Read Ansible Documentation

- List of all ansible modules
- Module Documentation Sections
    - Table of Contents
    - Synopsis: short explanation of what this module does
    - Requirements: lists all requirement to run the module
    - Parameters: list of all valid parameters
        - Broken down in parameter name, possible parameter values, and an explanation
    - Notes: general information about the module
    - Examples: examples of the module being used

    - Return Values: list of return data presented similarly to parameters
- Inventory Documentation
- Playbook Documentation

Playbook Basics

- Definitions:
    - Playbook: List of plays
    - Play: List of tasks that are performed on a group of hosts
    - Task: A call to an ansible module
- Playbook Example

```
---
- hosts: webservers
  remote_user: root

  tasks:
  - name: ensure apache is at the latest version
    yum:
      name: httpd
      state: latest
  - name: write the apache config file
    template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf

- hosts: databases
  remote_user: root

  tasks:
  - name: ensure postgresql is at the latest version
    yum:
      name: postgresql
      state: latest
  - name: ensure that postgresql is started
    service:
      name: postgresql
      state: started
```

- Playbook Basic Components
    - Hosts and Users
        - hosts: list of one or more groups to run play on
        - remote_user: user to run tasks as
        - Use ***become: yes*** to run the play as root on the remote systems
            - Note: this will not login via SSH as root. Rather, it will login with a non-root user and escalate privileges after logging in
    - Variables
        - Vars can be specified at the top of the play and referenced anywhere later in the play
        - If a var named ***version*** was declared at the top of the play, you can reference it later with the following syntax: ***“{{ version }}”***
    - Tasks list
        - List of ansible modules to run which are executed in order
    - Handlers

        - List of tasks that only run if notified by a notifier
        - Will only run once per play at the end of the play no matter how many notifiers notify it
- Running a Playbook
    - Have to use the ansible-playbook command. Similar to the ansible command

Ansible Vault

- TODO

Intermediate Playbooks

- TODO

Variables and Jinja2

- TODO