Configuration management with

Ansible

Lab environment

- Subnet **192.168.56.0/24** exists in VirtualBox (“File”=>”Host Network Manager” or “Ctrl+H” in main window)
- `mkdir ansible_lesson01`
- `cd ansible_lesson01\`
- `git clone https://bitbucket.org/astrukov/ansible_lab.git`
- `cd ansible_lab\`
- `vagrant up`

Ansible installation

Control node:

- `yum install -y epel-release`
- `yum install -y ansible`
- `ansible --version`
  - should be 2.9 (latest)

Connection configuration

- All nodes:
  - `useradd ansible`
  - `echo password | passwd --stdin ansible`
  - `echo "ansible ALL=(ALL) NOPASSWD:ALL" > /etc/sudoers.d/ansible`
- Control node:
  - `su - ansible`
  - `ssh-keygen`
  - `ssh-copy-id node1.example.com && ssh-copy-id node2.example.com`

Inventory

- Create inventory file “inventory” in new directory (e.g. “lesson1”)
- Add node1 and node2 to group “nodes”
- `ansible all –i inventory --list-hosts`

Ansible configuration file

- ansible.cfg
  - The generic file /etc/ansible/ansible.cfg
  - The user specific file ~/.ansible.cfg
  - The ansible.cfg file in the project directory (takes precedence)
- It’s common practice to use ansible.cfg file in the project directory
- $ANSIBLE_CONFIG environment variable
- The ansible.cfg file that is used should contain all environment variables
- `ansible –v` shows which configuration file is used

Ansible configuration file

ansible.cfg contents

- become
- become_user
- become_ask_pass
- Inventory
- remote_user

Adding managed host

- ansible user account
- ssh keys
- Logon on managed host to copy public key
- sudo configuration
- Python
- Update the inventory file

Ansible architecture

- Managed hosts, running SSH
  - Python 2.7 or later
- Controller host
  - Inventory
  - ansible.cfg
  - Python 2.7 or later
- Playbooks
  - Jinja2 templates
  - Modules
  - Plugins

Infrastructure deployment with Ansible

- Installation is taken care of by other utils
- Ansible can be used to:
  - Configuration of software repositories
  - Application installation
  - Files modification
  - Firewall configuration
  - Services configuration (start\disable)
  - Application testing

Modules

- Modules are programs that Ansible runs to perform specific tasks on host
- Included in playbooks, or referred to when running ad-hoc commands
- Ansible comes with hundreds of modules, and administrators can write their own modules as well

Module types

- Core modules
- Extra modules
- Custom modules
- Module location depends on Linux distro
  - /usr/lib/python2.7/site-packages/ansible/modules

Ad-hoc commands

Modules

- command: runs a command on a managed host
- shell: runs a command on managed host through the local shell
- copy: copy a file, change content on a remote host in a target file
- raw

Using modules

- `ansible –m <modulename>`
  - `ansible –m ping all`
- Playbook:

```
tasks:
- name: Install a package
   yum:
     name: vsftpd
     state: latest
```

Module documentation

- http://docs.ansible.com
- `ansible-doc –l`
  - `ansible-doc <modulename>`
  - `ansible-doc –s <modulename>`

Playbooks

- YAML
- Indentation; spaces
- Do **NOT** use tabs for indentation!
- ---
- …

YAML

- key: value
- `- list`
- `ansible-playbook --syntax-check example.yaml`

Playbook structure

- Collection of plays. Each play defines a set of tasks that are executed on the managed hosts.
- Tasks are performed by using Ansible modules
- Ordering is important
- Desired state
- Idempotent
  - Avoid using modules like command, shell and raw

The task attribute

- The most important attribute is the task attribute:

```
tasks:
- name: run service
   service: name=vsftpd enabled=true
```

- «-» marks the beginning of a list of attributes
- If multiple tasks are defined, each first attribute of the task starts with a «-»

Other attributes

- name
- hosts
- remote_user
- become
- become_method
- become_user

Running playbooks

- `ansible-playbook`
- `ansible-playbook --syntax-check`
- `ansible-playbook -C` – dry run
- `ansible-playbook --step` – step by step execution

Variables

- Variable is a label that can be referred to from anywhere in the playbook, and it can contain different values, referring to anything
- Variable names must start with a letter and can contain letters, underscores and numbers
- Variables can be defined at a lot of different levels

Arrays

- An array is a variable that defines multiple values, including their specific properties

```
cities1:
  - Moscow
  - StPetersburg
  - Saratov
  - Kazan
cities2: [Moscow, StPetersburg, Saratov, Kazan]
```

Variables scopes

- Global scope: command line on ansible config file
- Play scope
- Host scope
  - This can be done through the inventory file
- Higher level wins
  - Global scope wins from host scope

Variable precedence

- include_vars
- Global scope
- Playbook variables
- Host level variables

Variables

- Variables can be defined in a playbook or included from external files
- Defining variables in a playbook

```
- hosts: all
  vars:
    user: alex
    home: /home/alex
```

Variable files

- YAML with variables
  - This file uses a path relative to the playbook path
- This file is called from the playbook , using **vars_files**

```
- hosts: all
  vars_files:
     - vars/users.yml
```

Variables

- In the playbook, the variable is referred to using double curly braces
- If the variable is used as the first element to start a value, using double quotes is mandatory

```
tasks:
  - name: Creates the user {{ user }}
    user: “{{ user }}”
```

Host and group variables

- A host variable is a variable that applies to one host that is defined in the inventory file
- A group variable applies to multiple hosts as defined in a group in the inventory file
- The recommended method is to use group_vars and host_vars directories

group_vars and host_vars

- Create directories **group_vars** and **host_vars** in project directory, which contains inventory file
- As example, if you have a host group webservers that is defined in the inventory file, create a file with the name **group_vars/webservers** and in that file define the variable
- Similar for individual host variables: create a file with the name of the host and put it in **host_vars**
- Variables can be overwritten from the command line, using the –**e “key=value”** command line option from the **ansible-playbook**

Facts

- A fact contains discovered information about a host
- Facts can be used in conditional statements
- The **setup** module is used to gather fact information
  - `ansible -m setup`

Filters

- Facts provide a lot of information
- Filters can be applied on the level 1 information that is displayed by the facts
- `ansible –i <inventory> -m setup –a ‘filter=ansible_kernel’`

Custom facts

- Custom facts can be created by administrators to display information about a host
- Custom facts must be defined in a file using the INI or JSON format and the .fact extension, and stored in the /etc/ansible/facts.d directory and will be shown as an “ansible_local” fact

```
ansible <host> -m setup –i <inventory> -a
‘filter=ansible_local’
```

Inclusions

- Using inclusions makes it easy to create a modular Ansible setup
- Tasks can be included in a playbook from an external YAML file using the **include** directive
  - Separate files for different tasks, which can be managed independently
- Variables can be included from a YAML or JSON file using the **include_vars** directive
  - Using this method overrides any other method to define variables
  - If you want to do this, make sure the **include_vars** happens before the actual usage of the variables

Flow control

- Flow control works with loops and conditionals to process items
- A loop is used to process a series of values in an array
- A conditional is a task that is executed only if specific conditions are met

Simple loops

- **with_item**

  -
    ```
    -yum:
      name: “{{ item }}”
      state: latest
     with_items:
       - nmap
       - net-tools
    ```

Nested loops

- **with_nested**
- In a nested loop, a loop inside a loop is called. If these are used, Ansible iterates over the first array, and applies the values in the second array to each item in the first array.

Over loop types

- https://docs.ansible.com/ansible/2.9/user_guide/playbooks_loops.html
- with_file
- with_fileglob
- with_sequence
- with_random_choice

Conditionals

- Conditionals can look at different items
  - Values of registered variables
  - Ansible facts
  - Output of commands
- Conditional operators

Conditionals

- **when** statement
  - Must be indented outside a module, at the top level of the task
- Multiple conditions

Jinja2 templates

- Python-based templates that are used to put host specific data on hosts, using generic YAML and Jinja2 files
    - Jinja2 templates are used to modify YAML files before they are sent to the managed host
    - Jinja2 can also be used to reference variable in playbooks
    - As advanced usage, Jinja2 loops and conditionals can be used in templates to generate very specific code
    - The host specific data is generated through variables or facts