Principes des Systèmes d'exploitation

Introduction to Ansible

Guillerme Duvillié

1. What is Ansible?
2. Prerequisites
3. Ansible first touch
4. Playbooks

Plan

1. What is Ansible?
2. Prerequisites
3. Ansible first touch
4. Playbooks

Configuration management

Ansible is (among other things) a **configuration management** tool.

Configuration management tool

Given a state description of a server, enforces the server to be effectively in that state.

- right packages installed,
- configuration files contain the expected values,
- tight services are running,
- …

Pros and cons

- Save time,
- Lower the risk,
- Improve control,
- Allows exact reproducibility,

- Error propagation,
- Automated tasks are no more eligible for junior administrator trainings.

Why Ansible?

- Many configuration management tools are available (Puppet, Chief, SaltStack, …),
- Ansible has some advantages:
  - Simple (human readable, no need to learn Python, low learning curve, …),
  - State Driven,
  - Secure (based on well tried-and-tested OpenSSH),
  - Simple tasks do not require scripts,
  - Idempotent (running the same script twice gives the same result as running it once),
  - Can be used as deployment software,
  - Large number of modules,
  - Almost nothing to install on remote hosts (OpenSSH only).

Plan

1. What is Ansible?
2. Prerequisites
3. Ansible first touch
4. Playbooks

Ansible prerequisites

You have to be familiar with:

- SSH connections,
- Interact with Bash command-line shell (pipes, stream redirections, …)
- Package installation,
- `sudo` command,
- File ownerships and permissions,
- Start and stop services,
- Environment variables set up and access,
- Python and bash scripting,
- Virtual machines management (network configuration, ports forwarding, …)

Architecture prerequisite

- Host machine with functional ansible (Linux/Unix distribution of your choice),
- A Debian 11 server,
- A Debian 10 server,
- An Arch Linux server,
- A Fedora 36 server,
- Working ssh connections for all the machines (based on ssh keys),
- Fixed network configuration for all the servers,
- A git repository to host all your playbooks.

Plan

1. What is Ansible?
2. Prerequisites
3. Ansible first touch
4. Playbooks

Inventory file

- Ansible only manage servers he is aware of,
- Use **inventory file**:
  - text file containing one host per line,
  - may define aliases,
  - may define ssh connections details,
  - may define virtual groups of machines,
  - …
- Default inventory file is located at `/etc/ansible/hosts`,
- Specific inventory file can be specified with `-i` parameter.

Example

```
1  [apt]
2  debian10 \
3      ansible_host=172.16.21.3 \
4      ansible_port=443 \
5      ansible_user=johndoe \
6      ansible_private_key=~/.ssh/ansible
7  www.myserver.com
8
9  [pacman]
10 RemoteHome
```

Listing 1: hosts file example

```
1 Host RemoteHome
2     Hostname www.mamaison.be
3     User johndoe
4     IdentityFile ~/.ssh/ansible
5     Port 443
```

Listing 2: ssh_config

Todo

Create your own inventory file in your dedicated git repository.

Test

Test connection with `$ ansible all -i path_to_hosts_file -m ping`.

Configuration file basics

Ansible looks for configuration at the following location (in the given order):

1. file pointed by `ANSIBLE_CONFIG` environment variable,
2. `$(pwd)/ansible.cfg`
3. `~/.ansible.cfg`
4. `/etc/ansible/ansible.cfg`

A configuration file should contain at least a `[defaults]` section.

Simplify inventory file

1. `inventory`: specifies the inventory file,
2. `remote_user`: the default user for login (if no other user is provided),
3. `private_key_file`: the default ssh private key.

```
1 [defaults]
2 inventory=hosts
3 remote_user=johndoe
4 private_key_file=~/.ssh/ansible
```

Listing 3: ansible.cfg

Todo

Simplify your inventory file by creating a configuration file.

Test

Test connection with `ansible all -m ping`.

Basic commands

- `ansible` is used to run single task on given hosts,
- `-m` options selects a module to run (`ping` module in the previous example),
- By default the module `command` is used (to run remote command),
- need a `-a` option to specify the command to run (in a double quoted string),
- `-b` allows command to be run as `root` (using `sudo`).

Todo

1. Get the uptime of all your machines,
2. Get the last 20 lines of your debian machines connection logs,
3. Create a file named `me` on the arch linux machine.

Ssh Agent

- Type key password quickly is annoying,
- Key without password (especially for `sudoers` is a security loophole),
- Use **`ssh-agent`**!

```
1  $ eval $(ssh-agent)
2  Agent pid xxxx
3  $ ssh-add ~/.ssh/ansible-key
4  Enter passphrase for ~/.ssh/ansible-key:
5  Identity added: ~/.ssh/ansible-key (xxx@yyy)
```

Plan

1. What is Ansible?
2. Prerequisites
3. Ansible first touch
4. Playbooks

Playbook overview

- Playbooks can be seen as configuration management script,
- They are:
  - repeatable,
  - reusable,
  - simple,
  - multi-machines,
  - written in YAML.
- Launch a playbook with `ansible-playbook` command.

YAML syntax (1)

- Begin with `---`,
- Comments are prefixed with `#`,
- Strings can (but don't need to) be quoted,
- Booleans may take several values
  (`yes`, `Yes`, `YES`, `on`, `On`, `ON`, `true`, `True`, `TRUE`, `y`, `Y`),
- Lists (sequences) are hyphenated dashed blocks:

```
1 list:
2     - This
3     - Is
4     - A List
```

YAML syntax (2)

- Dictionaries (mappings) are hyphenated key/value tuples:

```
1 dictionary:
2     key1: value1
3     key2: value2
```

- They can be described inline:

```
1 dictionary: {key1: value1, key2: value2}
```

- Use `>` to suppress carriage returns in the following block,
- Use `|-` to keep carriage returns in the following block, but consider it as a single block.

Playbook example

```yaml
---
- name: Display logs
  hosts: debian
  become: true
  tasks:
    - name: Displayer
      command: tail /var/log/auth.log
```

Playbook overview

- A playbook is a list of dictionaries (called plays),
- A play must contain:
  - A set of `hosts`
  - A list of `tasks`,
- Other common settings:
  - `name`: human readable name of the play (describes what the play is about),
  - `become`: indicates whether the play should be run as root,
  - `vars`: list of variables and values.
- Tasks must contains at least a module name.

Exercice (1)

Using ansible documentation for the `apt` package (Documentation link), write a playbook that upgrades debian hosts.

Exercice (2)

1. Modify your inventory file to create one group per distro type,
2. Using modules `dnf`, `apt` and `pacman`, write a playbook that upgrade all your virtual machines.

Conditional tasks

- Tasks may contain a `when` entry that creates conditions on when to apply the latter.
- Can use the variable `ansible_fact` to design conditions.

Exercice (3)

1. Reset your inventory to its previous state,
2. Using Conditions Documentation, modify your upgrade script to upgrade all your machines.