Ansible:

Overview

- **Ansible** is open source software that automates software provisioning, configuration management, and application deployment.
- **Ansible** connects via SSH, remote PowerShell and other remote APIs.
- Some of **Ansible** design principals are:
  - Simple setup process and a minimal learning curve
  - Manage machines very quickly and in parallel
  - Avoid custom-agents and additional open ports
  - Describe infrastructure in a language that is both machine and human friendly
  - Be the easiest IT automation system to use, ever
- For example, let’s say we have 10 servers, which all need to have Nginx up and running.
- The “old” process will be: SSH into the machine → apt-get install Nginx → start Nginx **X 10**

https://docs.ansible.com/

Setup

- Ansible uses python interpreter to run its modules, therefore we will need to install Python on the host in order for Ansible to communicate with it.

```
$ sudo apt-get update

$ sudo apt-get install python
```

- Next we will install Ansible

```
$ sudo apt update

$ sudo apt install software-properties-common

$ sudo apt update

$ sudo apt install ansible
```

- To check Ansible is installed correctly, let’s run the following command:

```
$ ansible –version
```

- Install sshpass which allows us to provide the ssh password without using the prompt:

```
$ sudo apt-get install sshpass
```

Host / Inventory

- Ansible keeps track of all of the servers that it knows about through a hosts file (also known as inventory file).
- We need to set up this file first before we can begin to communicate with our other computers (servers).

```
$ sudo nano /etc/ansible/hosts
```

- For our practice, Let's start a VM, and obtain its IP address (using ip a)
- Let's add our server information including IP address, privileged user name, and user password one under the other and save:

```
10.0.1.2 ansible_ssh_pass=123456 ansible_ssh_user=root
```

- Let's ping our servers, to validate connection:

```
$ ansible -m ping all
```

Host group

- Ansible lets us use host groups which are used in classifying systems and deciding what systems you are controlling at what times and for what purpose.
- For example:

```
[webservers]
10.0.1.2 ansible_ssh_pass=123456 ansible_ssh_user=root
 [dbservers]
10.2.8.0 ansible_ssh_pass=123456 ansible_ssh_user=root
```

- Using it like below:

```
$ ansible -m ping webservers
```

Playbook

- Playbooks allow you to organize your configuration and management tasks in simple, human-readable files.
- Each playbook contains a list of tasks (‘plays’ in Ansible parlance) and are defined in a YAML file.
- Playbooks and roles are large topics so I encourage you to read the docs.
- Let’s create a simple example playbook, which will ping all servers:

```
$ sudo nano ping.yml
```

- containing the following:

```
- hosts:
    - all
  tasks:
    - action: ping
```

- To run it simply run:

```
$ ansible-playbook ping.yml
```

- ** Remember yml files can be annoying, so I suggest using http://www.yamllint.com

Ansible tower

- One of the major gripes from Ansible users is that it didn’t have a proper GUI.
- And that’s putting it mildly--the GUI was so bad that in the early days it wasn’t even properly synced to the CLI, meaning that the CLI and GUI could give you 2 different query results about the state of a certain node.
- Ansible itself was (and still is) rather new, so most of its users were by definition new users.
- Ansible Tower, previously called the AWX project, is the fix to this problem.
- It is a web-based UI for Ansible, containing the most important Ansible features, especially those that render better as graphical rather than text-based output.
- Ansible tower is not free, and the base plans are starting from 5,000-17,500$/year.