Ansible for Windows

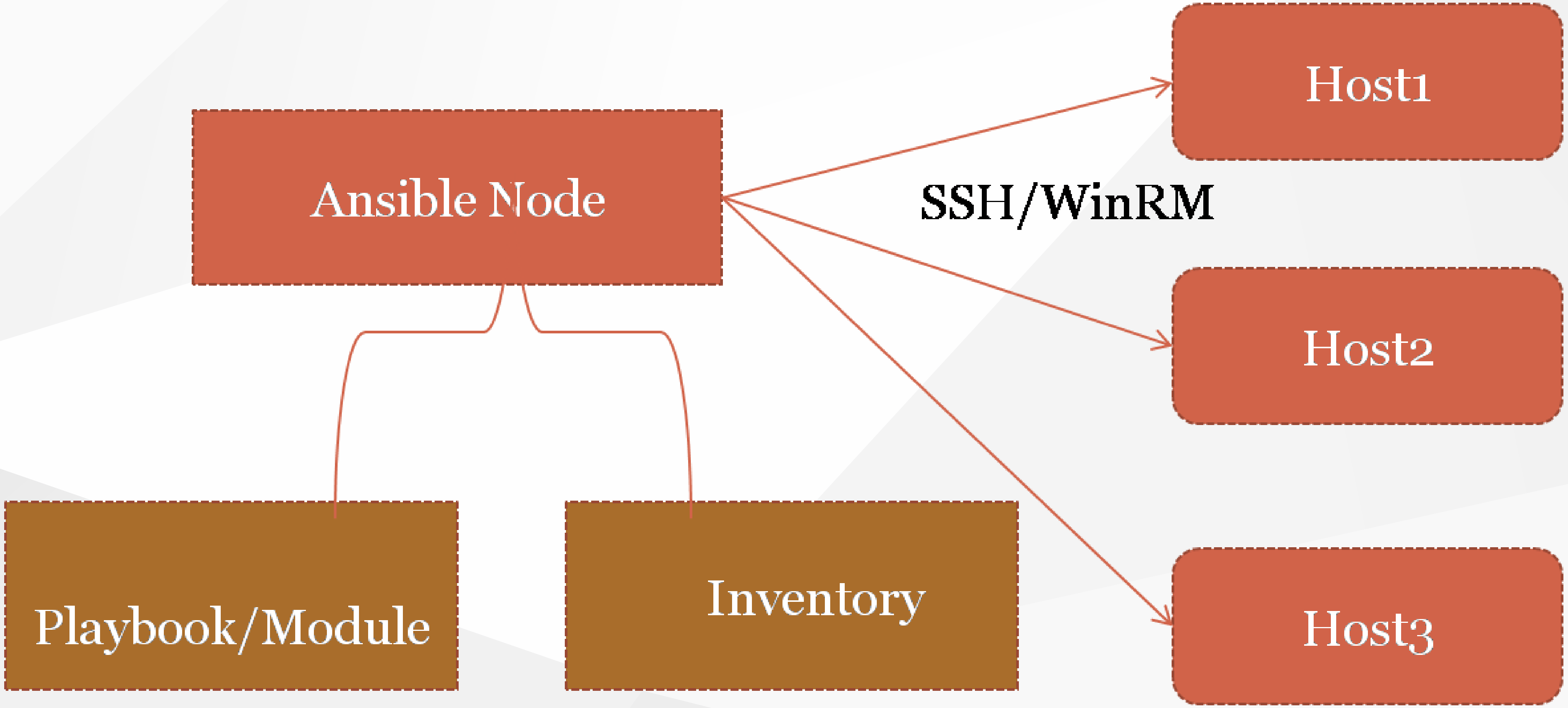
Created by Hocine Hacherouf (@hocinehacherouf)

Ansible Concepts

ANSIBLE

What is Ansible ?

- Open source
- Based on Python
- Created in 2012 and acquired by Red Hat in 2015
- Tool for machine provisioning, configuration and deployment
- Agentless
- Idempotence stateless



[Group A]
Host 1
[Group B]
Host 2
Host 3

Ansible Communication Protocoles

- Ansible + Linux 👉 SSH
- Ansible + Windows 👉
  - WinRM: **Win**dows **R**emote **M**anagement
  - SSH (⚠️ Experimental)

WinRM

- **Win**dows **R**emote **M**anagement: Used by Ansible to to communicate with Windows machines.
- WinRM default ports
  - HTTP : 5985
  - HTTPS : 5986 (SSL certificats required)
- ℹ️ The python module `pywinrm` is required by Ansibe for Windows support.

WinRM and Authentification

- Basic
- Certificate
- NTLM
- Kerberos
- CredSSP

WinRM and Authentification: Basic

- The simplest authentication options to use,
- Can only be used for local accounts (not domain accounts).

WinRM and Authentification: Certificate

- Authentication uses certificates as key, similar to SSH key pairs
- Not enabled by default on a Windows host

WinRM and Authentification: NTLM

- Enabled by default on the WinRM service
- Support local and domain users
- More secure than Basic

ℹ We will use **NTLM** on our labs

WinRM and Authentification: Kerberos

- Recommended when running on a domain environment
- Support message encryption over HTTP and credential delegation
- The most secure available on WinRM service
- Ansible controller require configuration
- The wrapper **pywinrm[kerberos]** is required

WinRM and Authentification: CredSSP

- New authentication protocol that allows credential delegation
- Support message encryption over HTTP
- The wrapper **pywinrm[credssp]** is required

Ansible ❤️ PowerShell

- Ansible can execute PowerShell commands/files
  - Ansible required powershell 4.0+ on target hosts
- Ansible modules for Windows are based on PowerShell 👉 `ansible.windows`

Templating with Jinja2

Ansible Installation

🧪 Lab 01

Configure testing environement

🧪 Lab 02

Ansible Project

Ansible Configuration

- Ansbile comes with a default configuration `/etc/ansible/ansible.cfg` .
- This configuration can be overriden with the follwing precedence:
  - The env variable: `ANSIBLE_CONFIG`
  - The file `ansible.cfg` on your current directory where ansible is executed
  - User home directory: `~/.ansible/ansible.cfg`
  - Default `/etc/ansible/ansible.cfg` file.

Inventories

- File that describes your infrastructure: Contant machines and their variables
- Machines can be grouped by type: db, frontend, backend...
- Multiple format to define an inventory:
  - ini
  - yaml
  - json

Inventory structure

```
├── inventories
│   ├── dev
│   │   ├── group_vars
│   │   │   └── all.yml
│   │   │   └── backend.yml
│   │   ├── host_vars
│   │   │   └── db01.yml
│   │   │   └── backend01.yml
│   │   │   └── backend02.yml
│   │   │   └── frontend01.yml
│   │   └── hosts
│   │
```

Inventory hosts file

```
[db]              # group name
db01              # machine name

[backend]
backend0[1:2]     # use range to simplify backend01 and backend02

[frontend]
frontend01
```

Secret Management with Ansible Vault

- To deal with sensitive data such as passwords, tokens, certificats..., we have to use the cli `ansbile-vault` to encrypt them.
- The enrypted data can be distributed or placed in `source control`.

Let's create an Ansible inventory with secrets

🧪 Lab 03

Playbook

- An ansible playbook is a yaml file that triggers the actions to be performed by ansbile on an inventory
  👉 Orchestrate

```yaml
---
- name: My awesome playbook
  hosts: all
  gather_facts: yes
  roles:
    - create-vm
    - install-database
```

```
ansible-playbook deploy.yml -i inventories/dev
```

Roles

Roles let you automatically load related vars, files, tasks, handlers, and other Ansible artifacts based on a known file structure. After you group your content in roles, you can easily reuse them and share them with other users.

Role's structure

- meta: Metadata of the role
- defaults: Default variables of the role
- vars: Other variables used by the roles and can be overridden by a user
- tasks: Set of tasks used by the role
- handlers: Handlers triggered used by the tasks of the role
- files: Static files used by the role
- templates: Templates based on jinja and used by the role

Create a role

- You can create the skeleton of an ansible role using the cli `ansible-galaxy` :

```
ansible-galaxy init roles/create-vm
```

```
├── roles
│   └── create-vm
│       ├── README.md
│       ├── defaults
│       │   └── main.yml
│       ├── files
│       ├── handlers
│       │   └── main.yml
│       ├── meta
│       │   └── main.yml
│       ├── tasks
│       │   └── main.yml
│       ├── templates
│       └── vars
│           └── main.yml
│
```

Ansible variable precedence

See 👉 Understanding variable precedence

Jinja2

Live DEMO

Let's play with Ansible

🧪 Lab 04

Ansible Collections

- With Ansible Collections, you can use ansible playbooks/roles created and maintained by the community.
- Example: Microsoft provides an ansible collection `azure.azcollection` that allows to provisionne and configure Azure ressources using Ansible

Ansible Galaxy

Ansible Galaxy or Galaxy is a hub for finding and sharing Ansible content

How to install ansible collections ?

Install a role

```
ansible-galaxy install geerlingguy.java
```

Install a collection

```
ansible-galaxy collection install azure.azcollection
```

Or install multiple collections/roles with a requirements file

```
ansible-galaxy install -r requirements.yml
```

```yaml
---
roles:
  # Install a role from Ansible Galaxy.
  - name: geerlingguy.java
    version: "1.9.6" # note that ranges are not supported for roles

collections:
  # Install a role from Ansible Galaxy.
  - name: install azure.azcollection
    version: ">=1.14.0" # usage of ranges
  # Install a role from Ansible Galaxy.
  - name: awx.awx
    version: 21.11.0
    source: https://galaxy.ansible.com
  # Install a role from a git branch/tag/commit
  - name: https://github.com/organization/repo_name.git
    type: git
    version: develop
```

Let's play with Ansible Collections

🧪 Lab 05

Best Practices

Usage of tags

If you have a large playbook, it may be useful to run/skip only specific parts of it instead of running the entire playbook. You can do this with Ansible tags:

```
---
- name: Install all critical and security updates
  win_updates:
    category_names:
    - SecurityUpdates
    state: installed
  tags: patching
```

```
ansbile-playbook deploy.yml --tags "patching"
ansbile-playbook deploy.yml --skip-tags "patching"
```

Smoke tests

- After a deployment on a target node, it's a good practice to run checks on new/updated compnents to verify that they are running and healty.
- The goal is to get information as soon as possible.
- Theses tests are called `smoke tests`
- An example of smoke tests : After deploying an api with ansible, the playbook should
  - Check if the authentication endpoint works
  - Call the api to query data
  - ...

Testing roles with Molecule

- Molecule is a wrapper for ansible that allows to tests and lint ansible roles.
- Tests are done by Molecule on disposable environements using drivers (supported by ansible): docker, vagrant…
- The usage of molecule is not only for local testing, it can/must be used on continuous integration pipelines.

Molecule Live Demo

Molecule

Automation with Ansible Tower/AWX

Ansible Tower/AWX

- Ansible Tower : UI web solution for ansible projects management and automation.
- AWX : An open source version of Ansible Tower

AWX

Live Demo

👌 Thanks