ANSIBLE

DEPLOYMENT MADE EASY

by Nico Oosterwijk

TRADITIONAL WEBSERVER INSTALLATION

- Setup physical of virtual machine using an ISO file like CentOS or Ubuntu
- Configure firewall rules to access the new server(s)
- Push your ssh public key to the servers
- Install pre-required software using yum (or apt-get)
- Setup required users and groups
- Install and configure TomCat/Apache

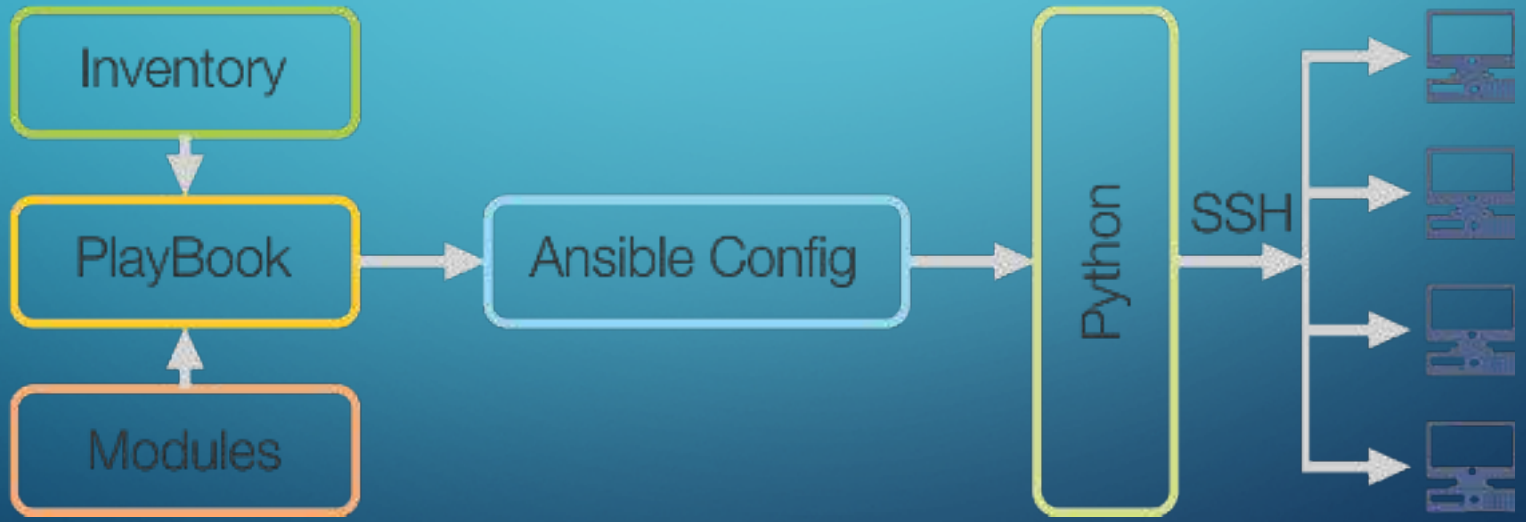
WHAT IS ANSIBLE?

Ansible is a radically simple IT automation engine that automates cloud provisioning, configuration management, application deployment, intra-service orchestration, and many other IT needs.

- Bare-metal -> Datacentre management
- Infrastructure -> network switches, routers, firewalls
- Cloud ready -> OpenStack, AWS, Azure
- Team development -> Ansible Tower
- Application deployment -> Playbooks

ANSIBLE MASTER

- Who is the master in an Ansible configuration?
- What defines the master?
- How is the communication between master and clients?



DIGITALINFO

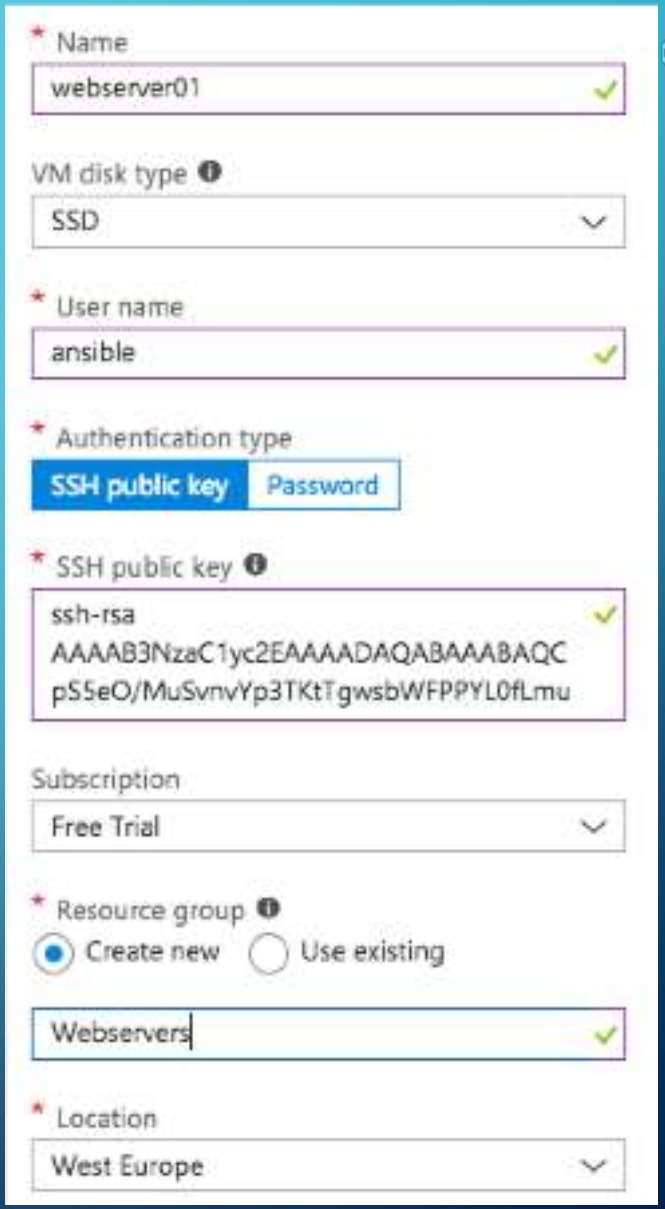
INSTALLATION

- MacOS:
  - pip install ansible
  - brew install ansible
- Linux:
  - sudo apt-get install –y ansible
  - sudo yum install –y ansible
- Windows 10:
  - Use Linux subsystem for Windows: install Ubuntu from the Microsoft Store

CONTROLLER AUTHENTICATION

- Ansible is agent-less!
- Prefer using ssh-keys!
- Create user ‘ansible’ on every host with useradd
- Create key pair for user ‘ansible’ with key-gen
- Add public key with ‘ssh-copy-id hostname’ to every host
- Create a file /etc/sudoers.d/ansible with the following line:

  ansible ALL-(ALL) NOPASSWD: ALL



DIGITALINFO

CONFIGURATION

ANSIBLE.CFG

- Configuration example:

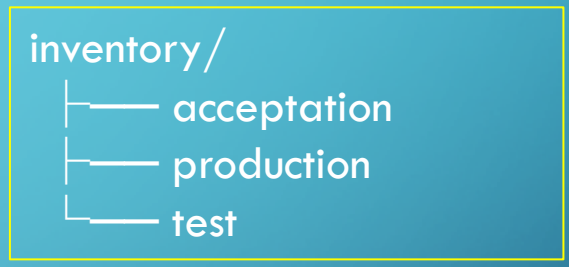
```
[defaults]
inventory = ./inventory
retry_files_enabled = False
nocows = True
```

INVENTORY

INVENTORY

- Can be a file or a directory
- An inventory file holds the node(s) to be controlled.
- Inventories can be nested into groups.
- Groups can be nested into supergoups.
- Example:

```
[tst]
tst-server01 ansible_host=10.0.0.1 ansible_user=ansible
tst-server02 ansible_host=10.0.0.2 ansible_user=ansible
```



INVENTORY

- Example:

```
[tst]
tst-server01 ansible_host=10.0.0.1 ansible_user=ansible
tst-server02 ansible_host=10.0.0.2 ansible_user=ansible
[musicplayers]
Huiskamer      ansible_host=192.168.178.21
Demoruimte     ansible_host=192.168.178.31
Badkamer       ansible_host=192.168.178.41
Tuin           ansible_host=192.168.178.51
```

MODULES

MODULES

- Downloaded from galaxy.ansible.com
- Modules are ‘libraries’ that are executed by ansible directly or by ansible-playbooks.
- Users can write their own modules.
- Modules are written in Python, therefore needs to be present on targets.
- Example:

```
ansible –m ping localhost
```

ADD-HOC COMMANDS

ansible-doc command

ADD-HOC ANSIBLE COMMANDS

- ansible -m ping tst
- ansible all -a who
- ansible tst -a '/sbin/reboot' -f 10
- ansible tst-server01 -m shell -a 'echo $TERM'

-m = module

-a = arguments

-f = fork (# processes parallel)

FILE TRANSFER

- ansible tst –m copy –a ‘scr=/etc/hosts dest=/tmp/hosts’

- ansible webservers -m file -a "dest=/var/html/index.html mode=600 owner=www-data group=www-data”

- ansible webservers -m file -a "dest=/var/web state=absent"

MANAGING PACKAGES

- ansible tst –m yum –a ‘name=ntpd state=present’ -k -b
- ansible tst –m yum –a ‘name=ntpd state=latest’ -k -b
- ansible tst –m yum –a ‘name=ntpd state=absent’ -k –b

-k = ask for connection password

-b = become (sudo)

MANAGING USERS AND SERVICES

- ansible tst -m user -a ‘name=ansible password=ansible’ -k -b
- ansible tst –m service –a ‘name=ntpd state=started’ -k -b
- ansible webservers -m service -a ‘name=httpd state=started’ -k -b

FACTS

setup

GATHERING FACTS

- ansible tst –m setup
- Facts represent discovered variables about a system. These can be used to implement conditional execution of tasks or to get ad-hoc information about the system.


```
"ansible_python_version": "2.7.10",
"ansible_real_group_id": 20,
"ansible_real_user_id": 501,
"ansible_selinux": {
    "status": "Missing selinux Python library"
},
"ansible_selinux_python_present": false,
"ansible_service_mgr": "launchd",
"ansible_stf0": {
    "device": "stf0",
```

PLAYBOOKS

PLAYBOOK

- Playbooks are script with deployment instructions.
- Playbooks are yaml-files.
- Playbooks can include other playbooks.
- Playbooks contains plays
- Playbooks can contain multiple plays

PLAYBOOK EXAMPLE

```
---
- hosts: webservers
  vars:
     http_port: 80
- name: Ensure apache is the latest version
  yum: name=httpd state=latest
- name: Write the apache config file
  template: src=/srv/httpd.j2 dest=/etc/httpd.conf
  notify:
    - restart apache
- name: Ensure apache is running
    service: name=httpd state=started enabled=yes
handlers:
  - name: restart apache
    service: name=httpd state=restarted
```

EXECUTING A PLAYBOOK

- ansible-playbook playbook.yml {options}

Options can be limited hosts, or asking for SUDO password, or only using a specific tagged task, etc…

Use '`ansible-playbook --help`' for more options.

VARIABLES

GROUP_VARS AND HOST_VARS

- Create a directory named ‘group_vars’ and/or ‘host_vars’
- Create a file called ‘all’ (or ‘test’, ‘production’, etc… for groups)
- Example group_vars/all:

```
---
download_folder: /data/download
install_folder: /data/install
```

DIGITALINFO

USE VARIABLES

- {{ download_folder }}
- {{ install_folder }}

- Example:

```
- file: path={{ install_folder}} state=directory
```

VARIABLES PRIORITY

- “role defaults”, which lose in priority to everything and are the most easily overridden
- variables defined in inventory
- facts discovered about a system
- “most everything else” (command line switches, vars in play, included vars, role vars, etc.)
- connection variables (ansible_user, etc.)
- extra vars (-e in the command line) always win

http://docs.ansible.com/ansible/latest/playbooks_variables.html#variable-precedence-where-should-i-put-a-variable

ROLES

ROLES - DIRECTORIES

- `defaults`
- `files`
- `handlers`
- `tasks`
- `templates`
- `vars`

'ansible-galaxy init demo'
will create a role-structure

ROLES EXAMPLES

```
---
- name: Installing required packages
  become: yes
  yum:
     name: "{{ item }}"
     state: present
  with_items: "{{ packages.required }}"
```

```
---
packages:
  required:
    - unzip
    - libselinux-python
```

ANSIBLE DEMO – TOMCAT INSTALL

- Create VirtualBox VM with Vagrant
- Git pull tomcat ansible project
- Run tomcat ansible-playbook

ANSIBLE

- Questions?

More information can be found at: **www.digitalinfo.nl** or **www.ansible.com** or E-mail your question to: **nico@digitalinfo.nl**

DIGITALINFO