2. Configuration Management with Ansible

Infrastructure Automation
HOGENT applied computer science
Bert Van Vreckem & Thomas Parmentier
2024-2025

Configuration management

Learning goals

- Understanding the concept of cfg mgmt systems
  - declarative, idempotent
  - advantages over scripting
- Setting up network services with Ansible
  - applying basic concepts: playbooks, variables, modules, roles
  - writing playbooks
  - using existing roles

What’s wrong with scripting?

```
dnf install -y httpd
systemctl enable --now httpd
firewall-cmd --add-service http --permanent
firewall-cmd --add-service https --permanent
firewall-cmd --reload
```

Adding a user

```
adduser admin
```

Run this script twice:

```
$ sudo ./setup-server.sh
$ sudo ./setup-server.sh
useradd: user 'admin' already exists
```

What about...

- small changes between hosts?
- maintaining config files?
- maintaining a large Bash code base?
- configuration drift?

Bash doesn’t scale!

Enter configuration management

- 1993: CFEngine by Mark Burgess
- Declarative
- Idempotent

Declarative

- Describe the desired state of the system
  - DSL, existing language
- Cfg mgmt system brings system to desired state
  - independent of initial state
  - in one pass
  - abort on fail

Example: CFEngine DSL

Domain Specific Language (DSL)

```
body common control {
    bundlesequence => { "install_packages" };
    inputs => { "libraries/cfengine_stdlib.cf"
};
}

bundle agent install_packages {
    vars:
        "desired_packages"
            slist => { "httpd", "mod_ssl" };
    packages:
        "$(desired_packages)"
            package_policy => "add",
            package_method => generic;
}
```

Example: Puppet manifest (DSL)

```
package { 'httpd':
  ensure => installed,
}

service { 'httpd':
  ensure => running,
}
```

Example: Chef recipe (Ruby)

```ruby
packages = ["httpd", "mod_ssl"]

packages.each do |pkg|
  package pkg do
    action: install
  end
end

service "httpd" do
  action [:start, :enable]
end
```

Example: Ansible playbook (YAML)

```yaml
- hosts: srv001
  vars:
    packages:
      - httpd
      - mod_ssl
  tasks:
    - name: Ensure packages are installed
      package:
        name: "{{ packages }}"
        state: installed
    - name: Ensure the service is running
      service:
        name: httpd
        state: started
        enabled: true
```

Idempotence

- Single pass
- End state is guaranteed
  - or run aborted!
- Only necessary changes

Advantages

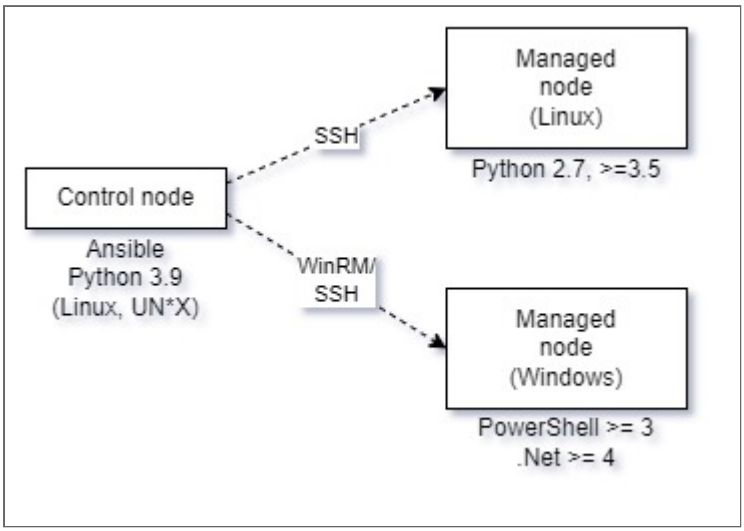
- Easier to reuse
- Readable
- Scaleable
- Config file templates
- Manage configuration drift
- Cfg mgmt = disaster recovery plan!

Recommendation

- Manage your entire infrastructure using a config management system
- Use revision control system!
- Never make manual changes to a production system!

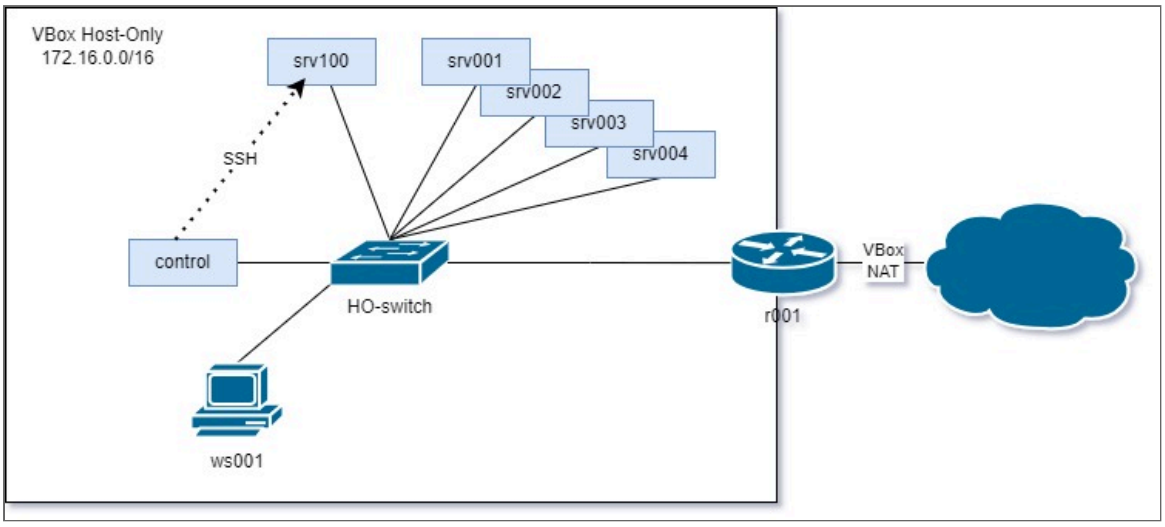
Ansible demo

Ansible control/managed nodes



Requirements for Ansible control/managed node

Lab assignment setup



Complete environment for the lab assignment

`vmlab` environment

```
> cd infra-labs-23-34-USERNAME/vmlab
> vagrant up control
> vagrant ssh control
> cd /vagrant/ansible
```

Add a new VM

In `vagrant-hosts.yml` (*before* the control node!):

```
- name: srv100
  ip: 172.16.128.100
  netmask: 255.255.0.0
```

and run `vagrant up srv100`

The inventory file

```yaml
# inventory.yml
---
servers:
  vars:
    ansible_user: vagrant
    ansible_become: true
  hosts:
    srv100:
      ansible_host: 172.16.128.100
      ansible_ssh_private_key_file:
         ../.vagrant/machines/srv100/virtualbox/private_key
```

Connecting to managed hosts

Try this:

```
> ansible -i inventory.yml srv100 -m ping
> ansible -i inventory.yml srv100 -m setup
```

Main playbook

```yaml
# ansible/site.yml
---

- name: Configure srv100
  hosts: srv100
  tasks:
    - name: Ansible demo
      ansible.builtin.debug:
        msg: "Hello from host {{ ansible_fqdn
         }}!"
```

Let’s try out the example playbook!

Running a playbook

```
[vagrant@control ansible]$ ansible-playbook -i
inventory.yml site.yml

PLAY [Configure srv100]
*********************************************

TASK [Gathering Facts]
*********************************************
ok: [srv100]

TASK [Ansible demo]
*********************************************
ok: [srv100] => {
    "msg": "Hello from host srv100!"
}

PLAY RECAP
*********************************************
```

Installing a role

```
> ansible-galaxy install bertvv.rh-base
```

Add a section `roles:` to `site.yml`:

```
# site.yml
---
- name: Configure srv100
  hosts: srv100
  roles:
    - bertvv.rh-base
  tasks:
    # ...
```

and run the playbook again.

Play it again, Sam!

After the first run:

```
PLAY RECAP
*********************************************************
srv100                     : ok=33   changed=13
unreachable=0    failed=0    skipped=19
rescued=0    ignored=0
```

After the second one:

```
PLAY RECAP
*********************************************************
srv100                     : ok=31   changed=0
unreachable=0    failed=0    skipped=19
rescued=0    ignored=0
```

Idempotency at work!

Roles: reusable playbooks

- **https://galaxy.ansible.com/**
- e.g., the rh-base role:
  - Galaxy page: **https://galaxy.ansible.com/bertvv/rh-base**
  - Github: **https://github.com/bertvv/ansible-role-rh-base**

  Role behaviour can be changed by setting (role) variables. See the README!

Initialising variables

- In the playbook
- `host_vars/srv001.yml`
- `group_vars/servers.yml`
- `group_vars/all.yml`
- …

```yaml
# ansible/group_vars/servers.yml
---
rhbase_install_packages:
  - bind-utils
  - tree
```

That’s enough for now!

Resources

- **Ansible documentation**
- **Ansible directory layout**
- Recommended books:
  - Geerling, J. (2020) ***Ansible for Devops***
  - Sesto, V. (2021) ***Practical Ansible***

Time to get started!

- Continue with the lab assignment