Ansible

Configuration management tool and ad hoc solution

Marcel Nijenhof <marceln@pion.xs4all.nl>

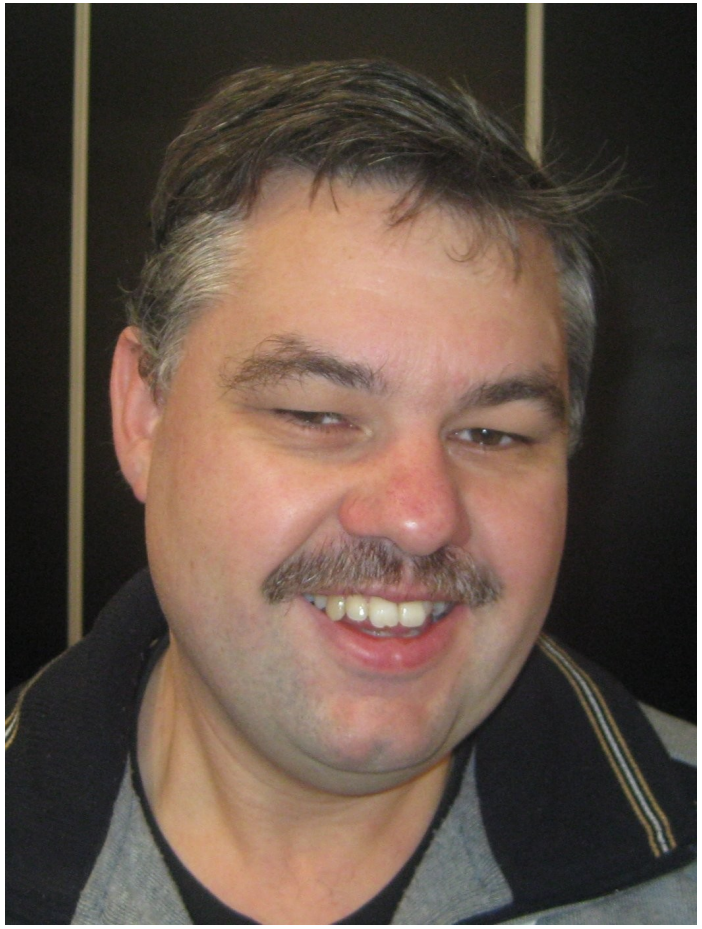
Index

- Introduction
- Installing & configuration
- Playbooks
- Variables
- Roles
- Ansible galaxy
- Configuration management tool
- Add hoc usage

Introduction

Marcel Nijenhof

- Computational physics
- 20 years unix & linux administrator
- Games
  - Britannica, 1861
  - http://megagame-makers.nl
- Proxy & Rijkswaterstaat



Introduction

Demos, Exercises & Questions

- Demos, Exercises
  - There is a lot time for demos & exercises
  - Based on CentOS 7

- Questions
  - Are welcome during the lecture

Introduction

Ansible

- A configuration management tool
  - Automatic deployment of system configurations
  - Maintain configuration during live cycle
  - Uses a dsl to specify the desired state
- Comparable with
  - Puppet, Cfengine, chef, …

Introduction

Differences with puppet

- Uses “ssh” for connection
  - No client software needed
    - Sshd
    - Sudo
    - python
- No master!
  - No default running components
    - Ssh
    - python
    - Crontab + authorized account
- Dsl is in yaml
- Written in python

Introduction

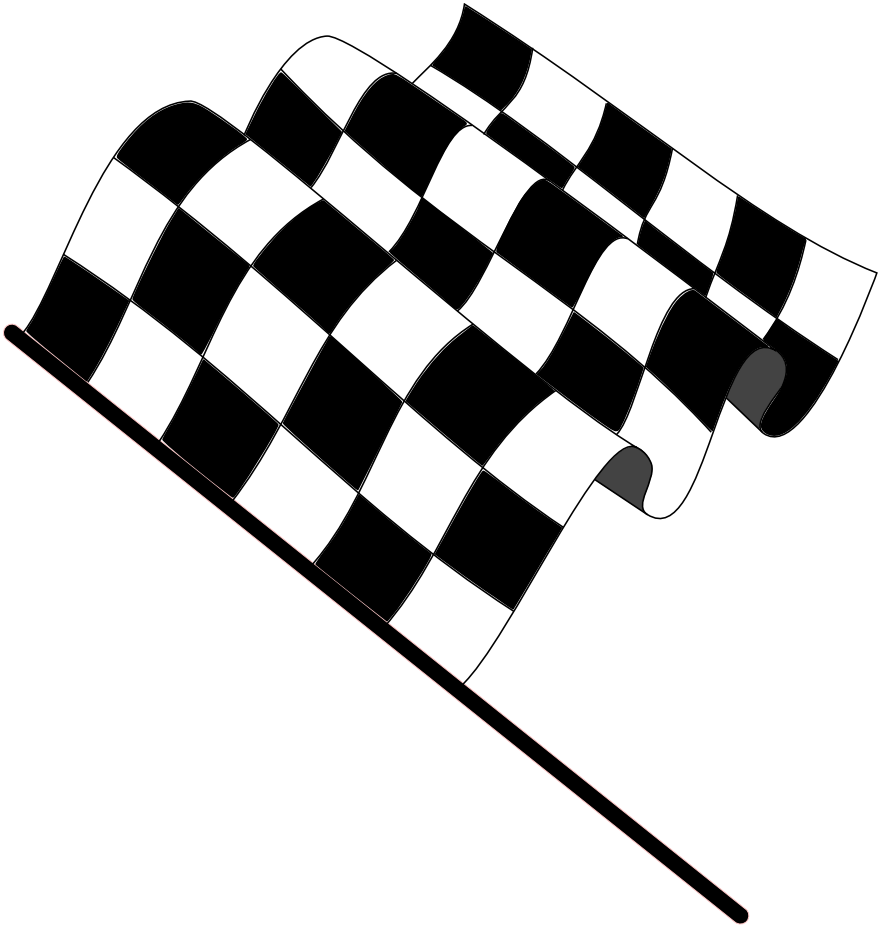
compare config management tools

https://en.wikipedia.org/wiki/Comparison_of_open-source_configuration_management_software

	Language	License	Agent-less	First release	Latest stable release
Ansible	Python	GPL	Yes	2012-03-08	2015-04-28 1.9.1
Chef	Ruby, Erlang	Apache	Yes	2009-01-15	2014-12-16 12.0.3
CFEngine	C	GPL, COSL	No	1993	2014-12-05 3.6.3
Puppet	Ruby	Apache	No	2005-08-30	2015-03-26 3.7.5

Installing & configuration

Finish



Installing & configuration

Ansible “master” (CentOS 7)

- Activate epel
  - https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
  - yum install epel-release-latest-7.noarch.rpm
- Install ansible
  - yum install ansible
    - Version: 1.9.2
- Debian 8:
  - apt-get install ansible
    - Version: 1.7.2

Installing & configuration

Master configuration

- Configuration files
  - /etc/ansible/ansible.cfg
    - Settings of ansible
      - Defaults work!
    - User version of the file
      - ~/.ansible.cfg
  - /etc/ansible/hosts
    - List of ansible clients
      - Group membership
      - Connection information
    - You need to configure your clients

Installing & configuration

ansible host file

- List of groups
  - Ini format file
  - List of hosts in the group
    - Hosts can be member of more then one group
    - Extra information per hosts is possible
      - Ssh port
      - connection information
      - variables
  - Groups of groups is possible

Installing & configuration

host file example

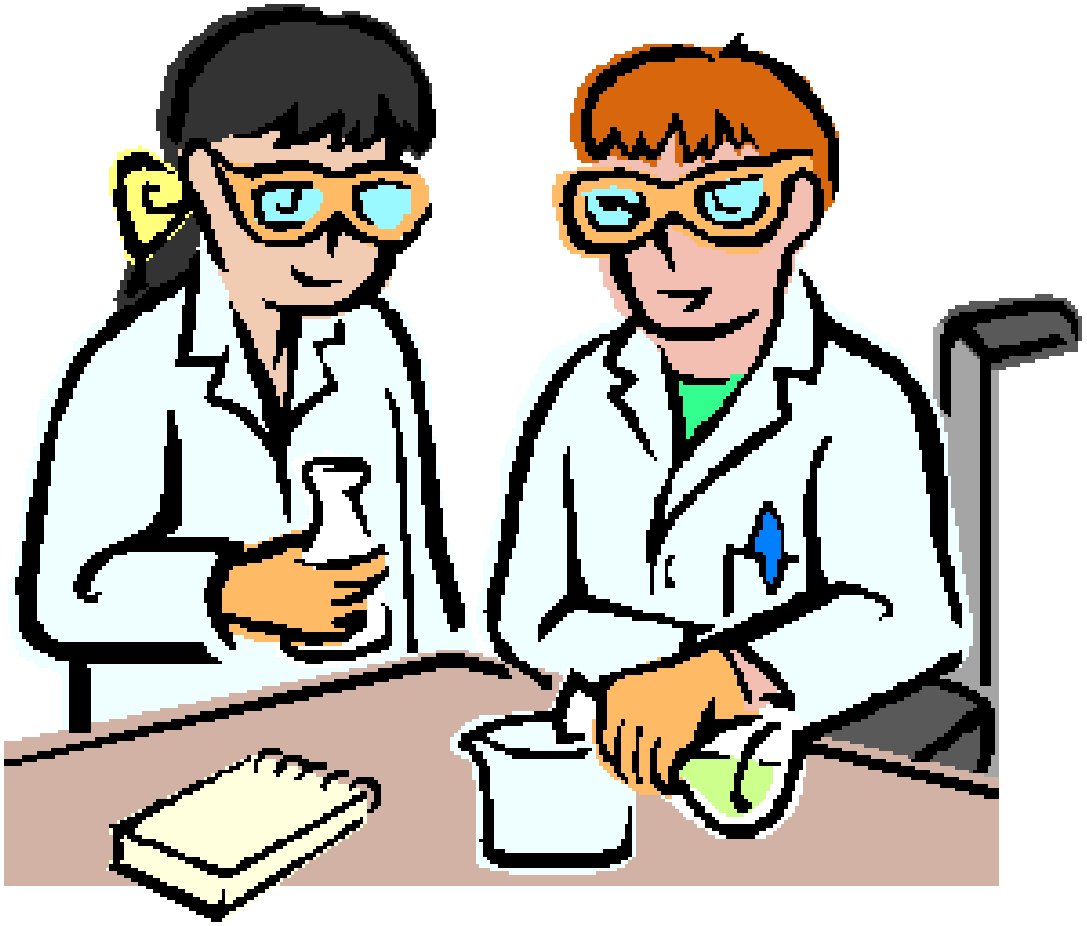
```
localhost
misc.example.lan

[db]
db-prim.example.lan
db-slave.example.lan

[web]
web*.example.lan
```

Installing & configuration

Demo: Ansible master



Installing & configuration

client requirements

- Software:
  - Ssh server
  - Sudo
  - python
- Privileges:
  - Login with ssh
  - Any sudo command

Installing & configuration

The client (CentOS 7)

- Add the client to ansible host file
- Login with ssh (with keys)
  - Ssh-copy-id
  - It's possible to use ansible with ssh passwords
- Make sure that sudo works for all commands
  - <user> ALL=(ALL) ALL
  - Sudo without tty is preferred
  - Without passwords
    - Or supply sudo password through ansible

Installing & configuration

Test the client

- ansible <client> -m ping
  - $ ansible d4-193 -m ping

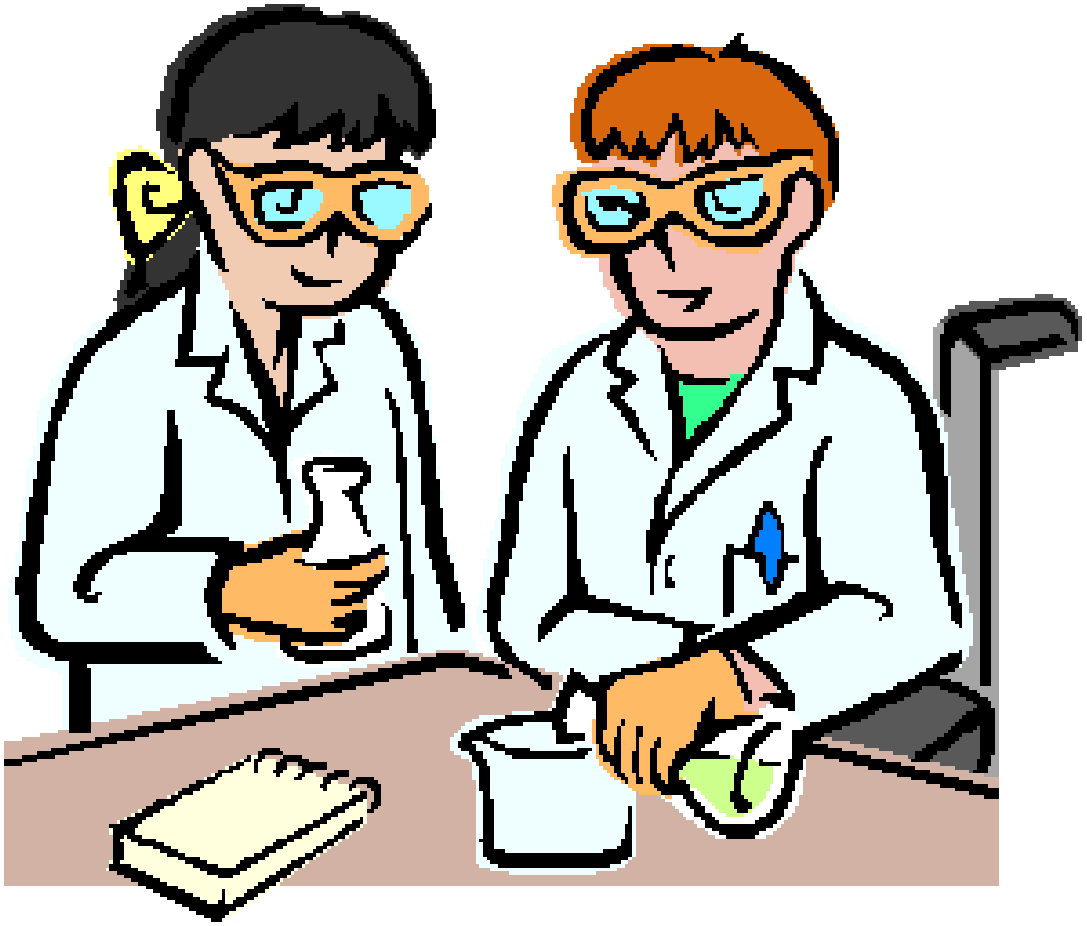
```
d4-193 | success >> {
    "changed": false,
    "ping": "pong"
}
```

- ansible <client> -m raw -s -a “id”
  - $ ansible d4-193 -m raw -s -a id

```
d4-193 | success | rc=0 >>
uid=0(root) gid=0(root) groups=0(root) ...
```

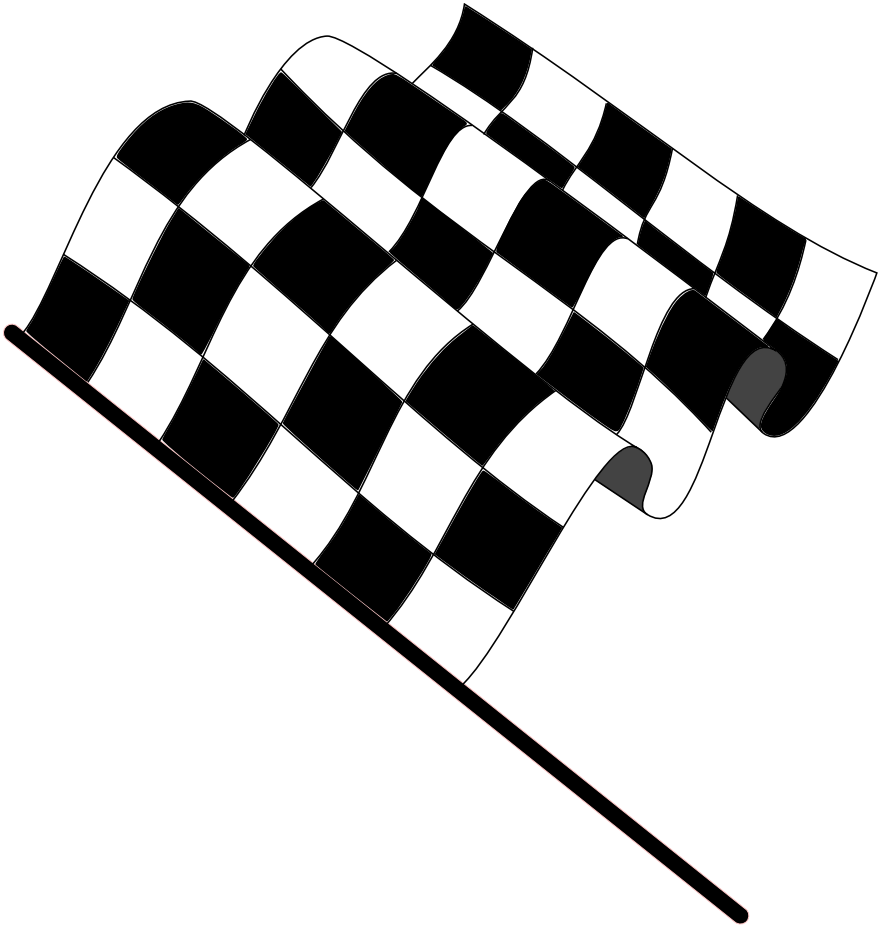
Installing & configuration

Demo: Ansible client



Installing & configuration

Finish



Playbooks

recipe to install and/or configure software

- Written in yaml
- Contains
  - options how to install
    - Host list
    - User for the installation
  - Program logic
    - Variables
    - Conditionals
    - Loops
  - Include statements
  - Calls to ansible modules
- Description of the desired state

Playbooks

Modules

- http://docs.ansible.com/ansible/list_of_all_modules.html
- Many (> 100)
  - Files:
    - File
    - Copy
    - Template
  - Packages:
    - Yum
    - Apt
  - User
  - ...

Playbooks

yaml

- https://en.wikipedia.org/wiki/YAML
- A format to serialize data
  - A yaml file start with “---”
    - One “os file” can contain multiple yaml files
  - Can contains hashes:
    - key: value
  - Can contain arrays
    - Each elemement start with a “-”
  - It's possible to nest data structures
    - Indention is used to specify nesting

Playbooks

yaml example

```
---
Persoon:
  - Name: Marcel Nijenhof
    Games:
      - Britannia
      - 1861
  - Name: ...
```

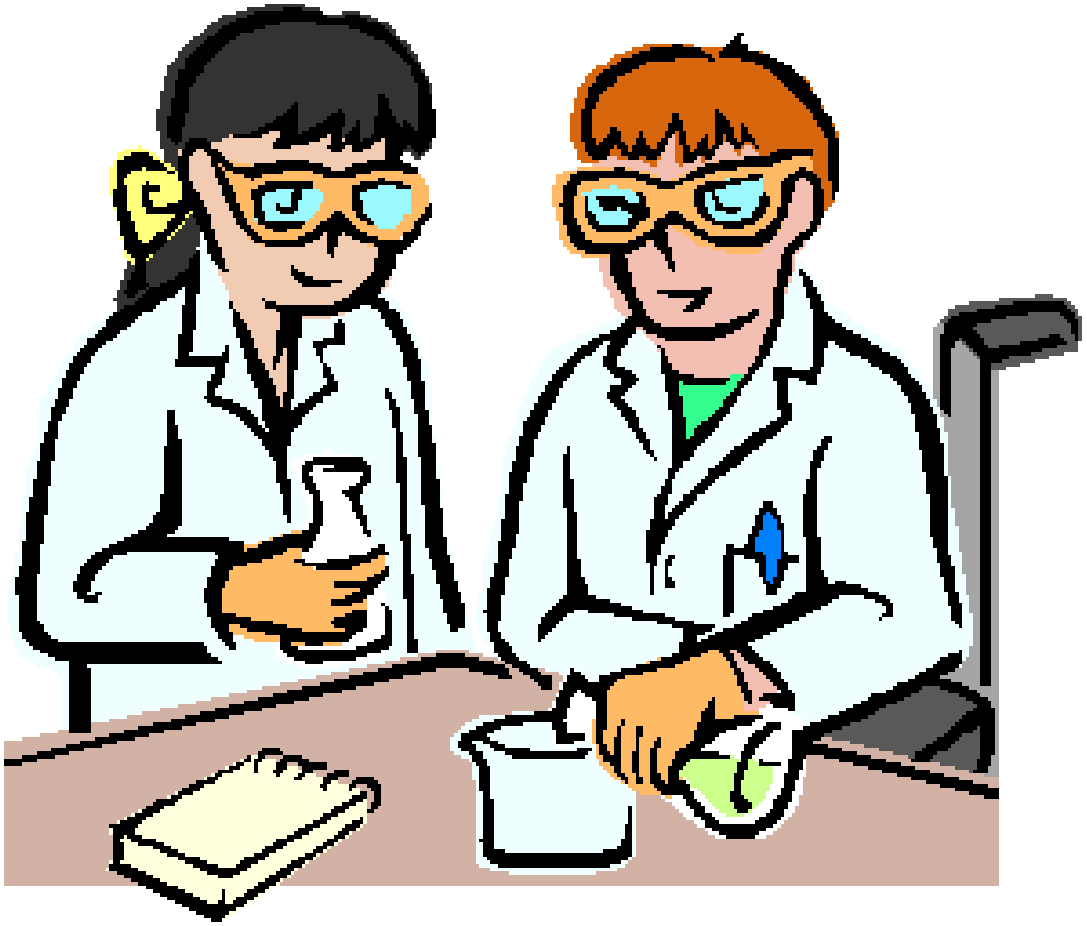
Playbooks

yaml & ansible

```
---
- hosts: localhost
  become: no
  tasks:
  - name: Ansible hello world
    debug:
      msg=”Hello world”
```

Playbooks

Demo: Ansible hello world



Playbooks

variables

```
- vars:
    name: marceln
    comment: Marcel Nijenhof
- name: Create a user
  user:
     comment={{ comment }}
     name={{ name }}
     …
```

NOTE:

This example is not usefull.

Playbooks

Conditionals

```
- name: install apache on redhat
  yum:
    name: http
    state: installed
  when: ansible_os_family == 'RedHat'
- name: install apache on Debian
    apt:
      name: apache2
      state: installed
    when: ansible_os_family == 'Debian'
```

Playbooks

Loops

- http://docs.ansible.com/ansible/playbooks_loops.html
- with_items
  - Loop over a array
- with_nested
  - Loop over 2 or more arrays
- with_dict
  - Loop over hashes
- ...

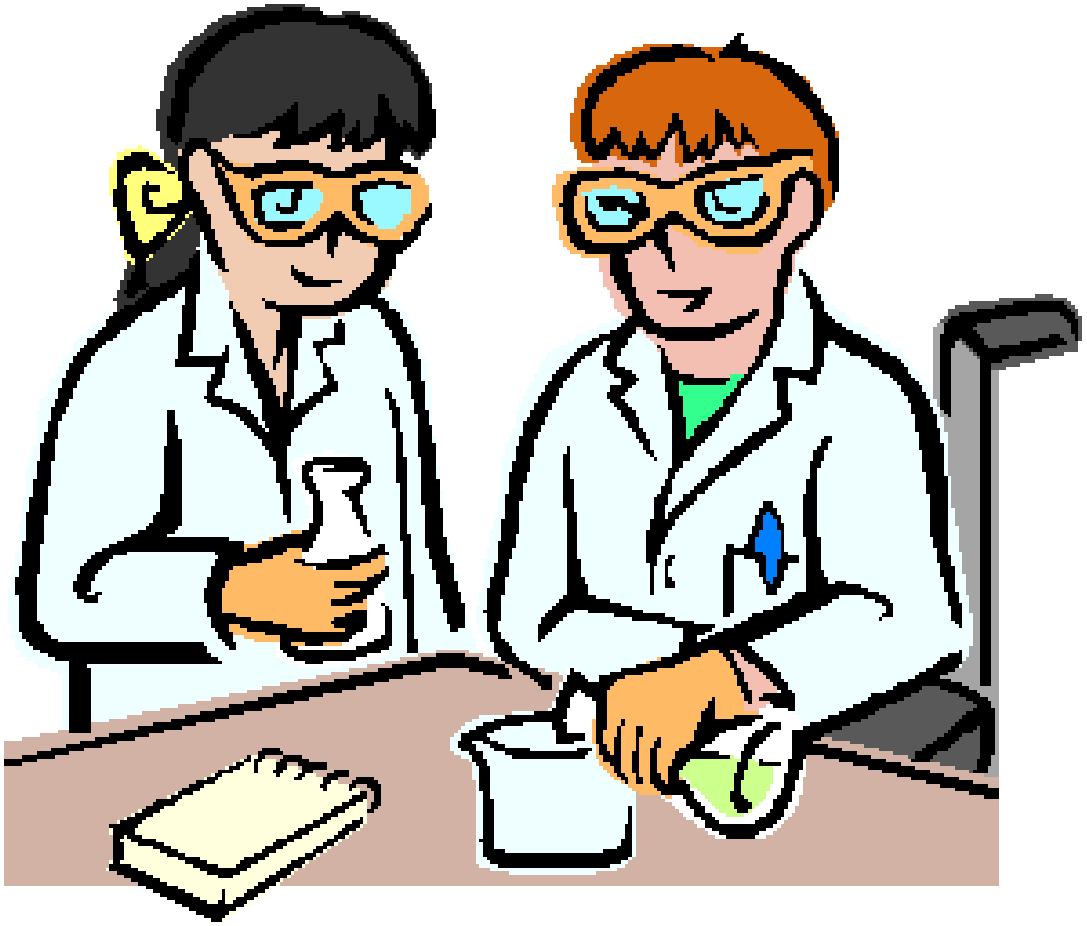
Playbooks

with_items

```
- vars:
    pkg:
      - httpd
      - php
- name: Install package list
  yum: name="{{ item }}" state=present
  with_items: "{{vars}}"
```

Playbooks

Demo: with_items



Playbooks

with_dict (definitie)

```
vars:
  users:
    marceln:
      uid: 1000
      comment: “Marcel Nijenhof”
    luc:
      uid: 1001
      Comment: “Luc Nieland”
```

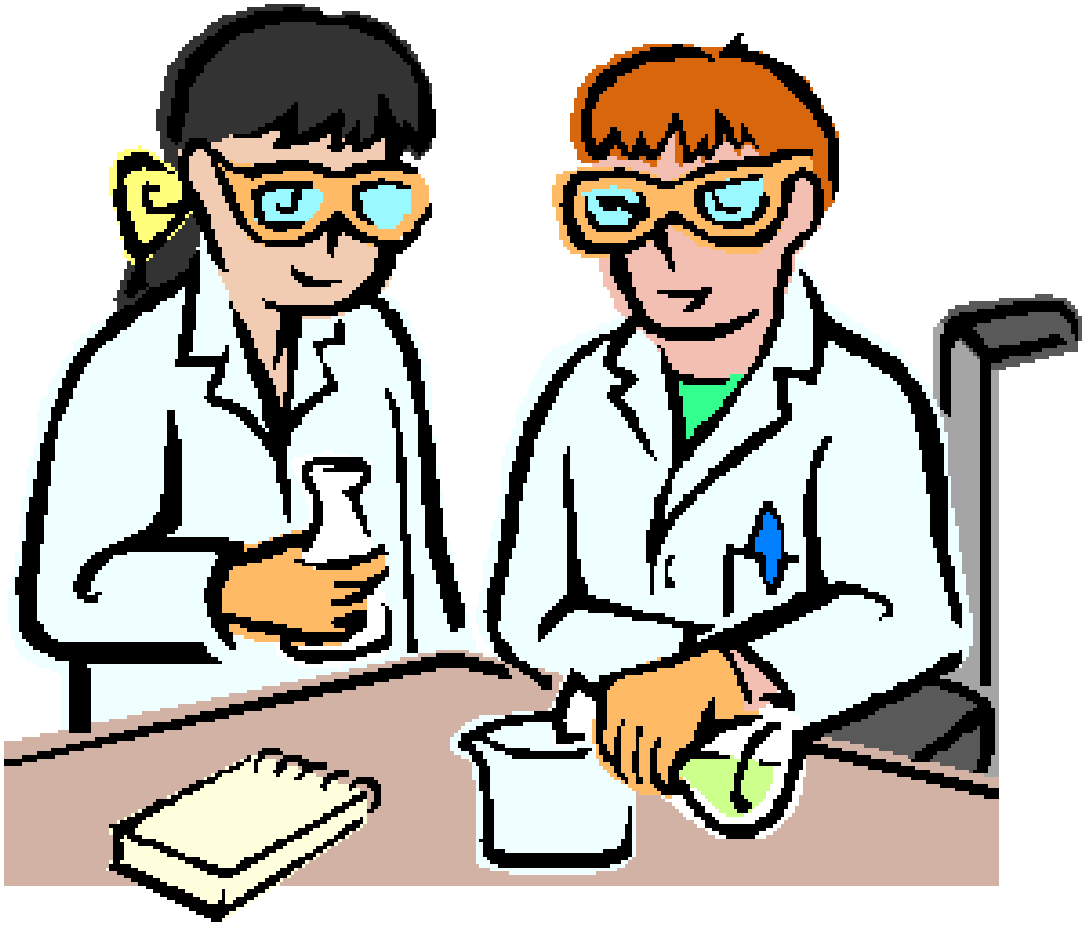
Playbooks

with_dict (code)

```
- name: Add users
  user:
    name=item.key
    comment=item.value.comment
    uid=item.value.uid
  with_dict: "{users}"
```

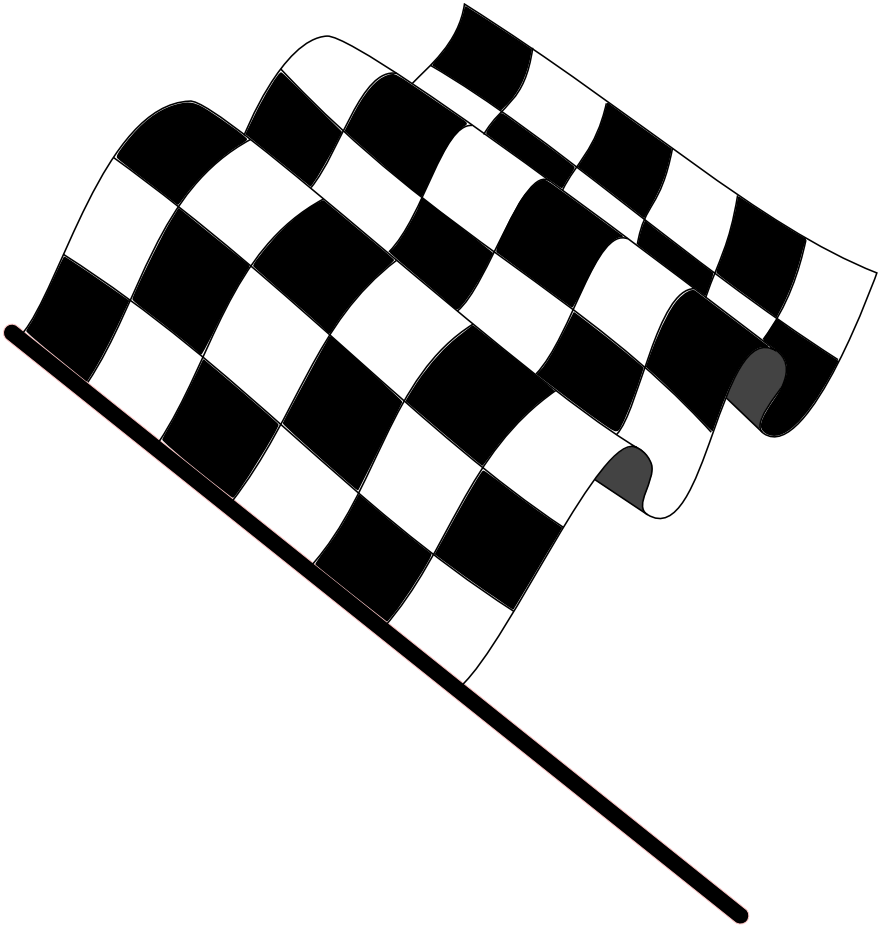
Playbooks

Demo: with_dict



Playbooks

Finish



Variables

sources

- Source of variables
  - Facts
    - Running os
  - Registered variables
    - Values from earlier tasks
  - Host declaration
    - Inventory file
    - host_vars
    - group_vars
  - From code
    - Default values
    - Standard values from code

Variables

sources

- Source of variables
  - Facts
    - Running os
  - Registered variables
    - Values from earlier tasks
  - Host declaration
    - Inventory file
    - host_vars
    - group_vars
  - From code
    - Default values
    - Standard values from code

Variables
facts (setup)

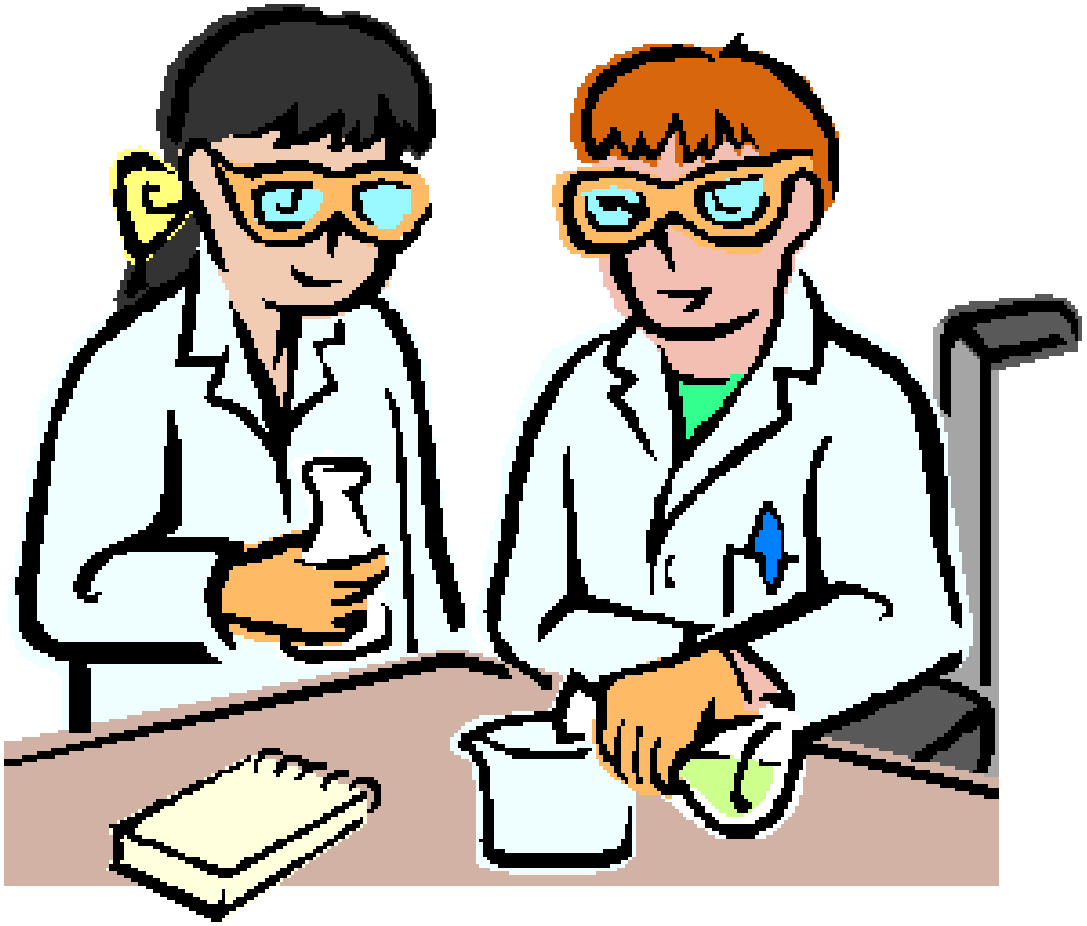
- ansible d4-193 -m setup
  - Gives a list of facts of the system
    - ansible_distribution
    - ansible_os_family
    - ansible_processor
  - These values can be used in playbooks
    - See “Playbooks conditionals”

Variables
facts (setup)

```
d4-193 | success >> {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "192.168.104.193"
        ],
...
        "ansible_architecture": "x86_64",
        "ansible_bios_date": "04/01/2014",
…
        "ansible_distribution": "CentOS",
        "ansible_distribution_major_version": "7",
...
```

Variables

Demo: setup



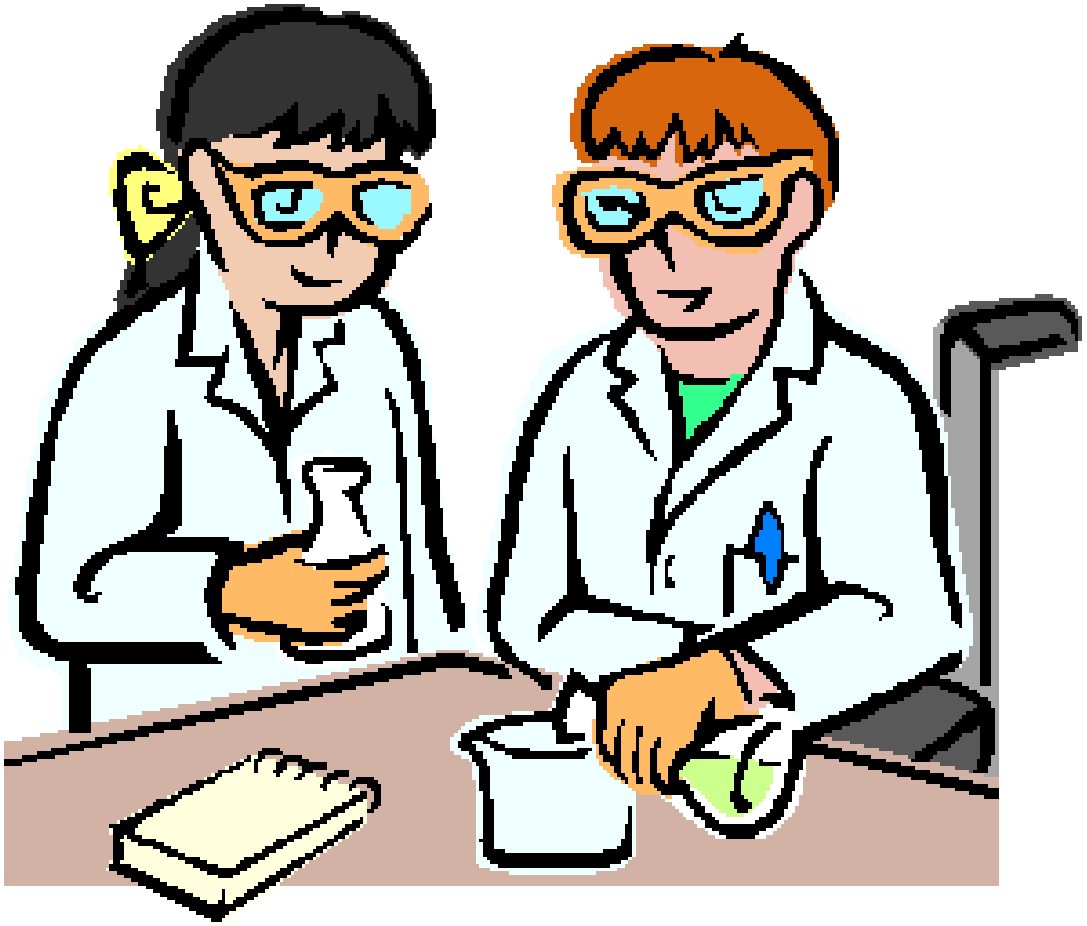
Variables

registered variables

- register: <name>
  - Store results of command in a variable
    - This is a data structure!
    - <name>.rc: Return code
    - <name>.stdout: Standard out
  - Use this result in later tasks
    - Debug: var=<name>

Variables

Demo: registered variables



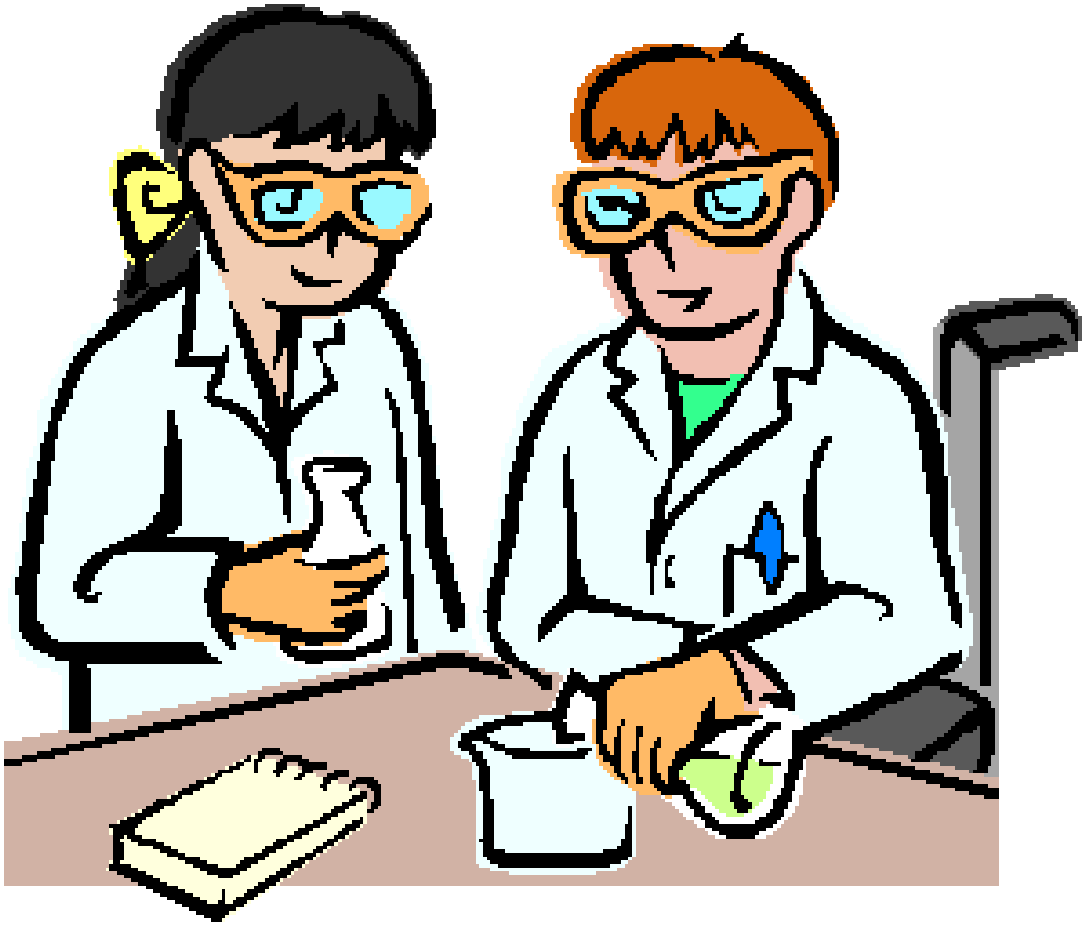
Variables

host_vars, group_vars

- host_vars/<hostname>.yaml
  - Variables for a specific host
- group_vars/<group name>.yaml
  - Variables for a specific group
- Just a yaml file with a data structure:
  - ---
    var: value

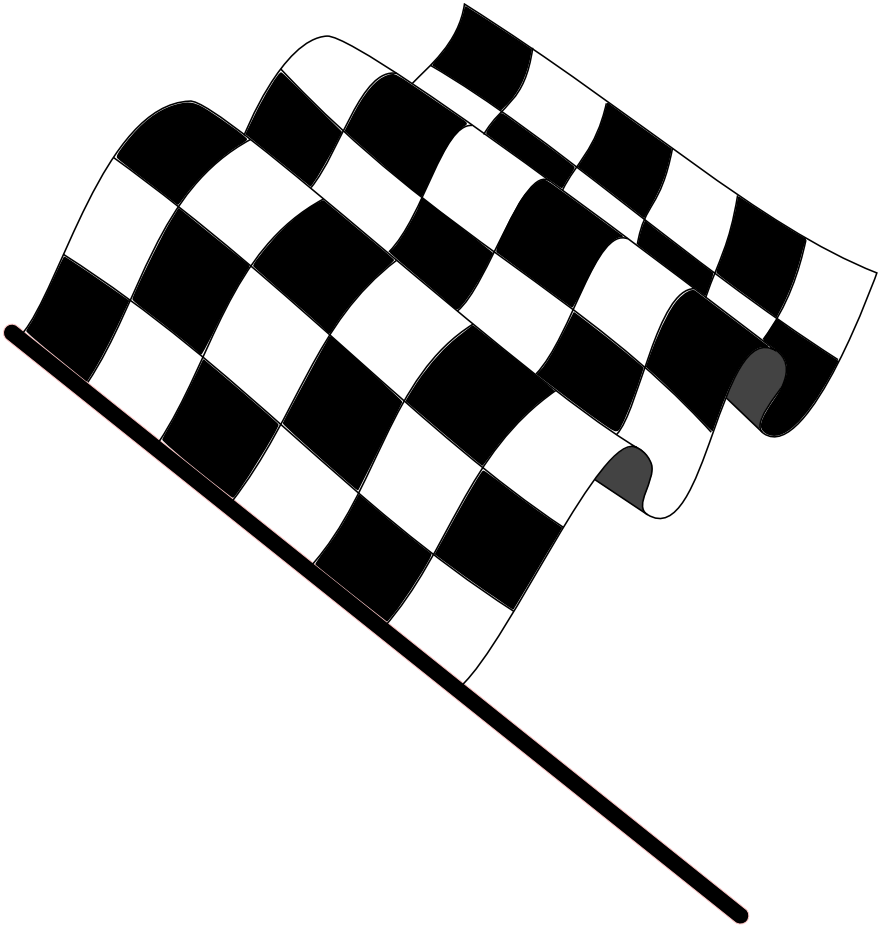
Variables

Demo: host_vars



Variables

Finish



Roles

set of files to implement one function

- Set of files to implement one function
  - Installing a apache server
  - Contains
    - Tasks to install en configure the server
    - Variables needed for the installation
      - Package names for RedHat or Debian
    - Default settings
      - Port 80
- There is a library with roles!!
  - First search before writing.

Roles

Layout

- README.md: Documentation of the module
- tasks: Task list for installation/configuration
- handlers: Task list for events
- defaults: Default values variables
- vars: Variables needed for install
- files: File store
- templates: Template store
- meta: Package information

Roles

Template role

- ansible-galaxy init <user>.<rolename>
  - Creates the default directory setup
  - Just fill in the details

Roles
README.md

- Documentation of the module
  - Standard github markup
  - Contains:
    - Requirements
    - Used variables
    - Dependencies
    - Example Playbook
    - License

Roles

tasks

- Contains:
  - main.yml
    May include:
    - include_vars: "{{ ansible_os_family }}.yml"
    - install.yml
    - config.yml
    - …
  - Contains tasks for install en configuration.
    - Notify command for handlers
      - Service reload

Roles

defaults

- main.yml
  - Contains default values for variables
  - Overrule with:
    - host_vars
    - group_vars
    - code

Roles

vars

- Defaults for various platforms
  - RedHat.yml
    - apache_pkg: httpd
    - apache_config_dir: /etc/httpd/conf.d
  - Debian.yml
    - apache_pkg: apache2
    - apache_config_dir: /etc/apache2

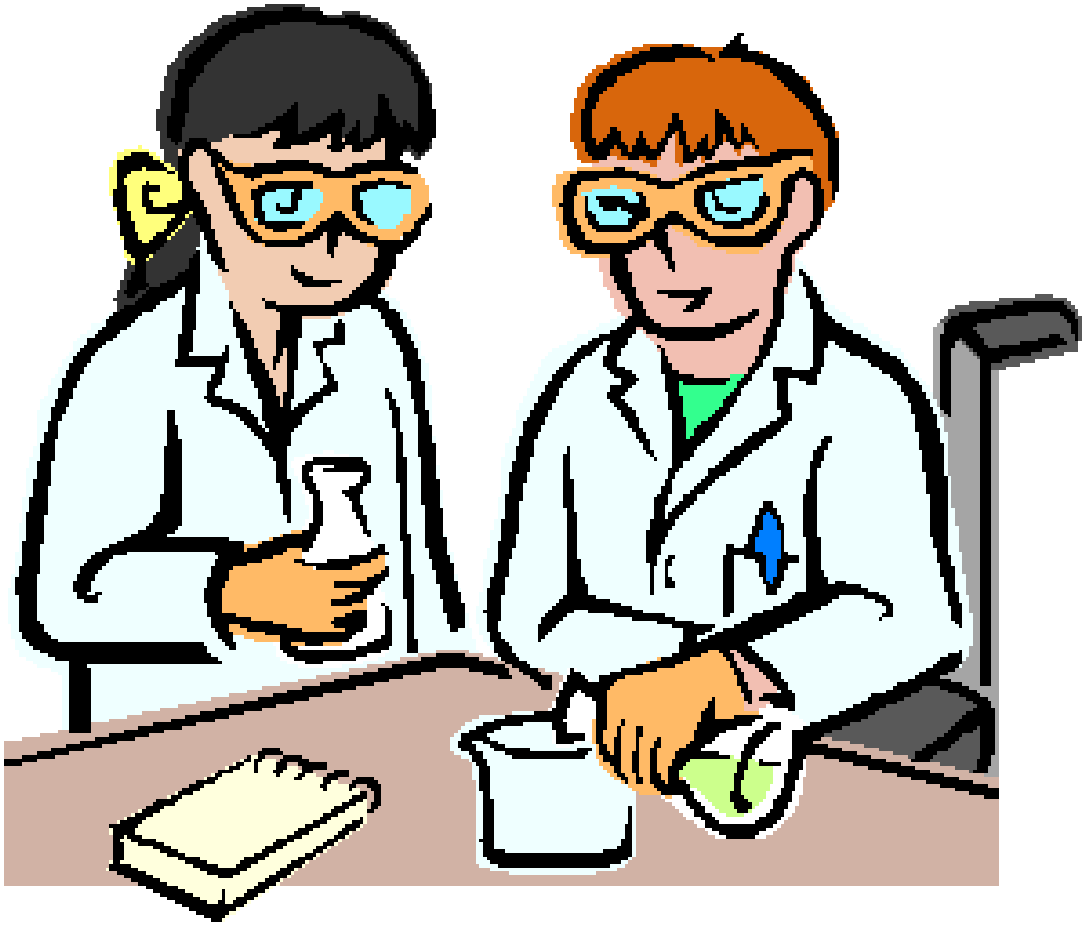
Roles

mydemocontent

- Distribute our content “index.html”
  - Uses vars for RedHat and Debian
    - In the correct directory
    - Under the correct user
- Create the module
  - mkdir roles
  - cd roles
  - ansible-galaxy init mydemocontent

Roles

Demo: mydemocontent



Roles

mydemocontent/tasks/main.yml

```
---
# tasks file for mydemocontent
- name: Include os dependend config
  include_vars: "{{ ansible_os_family }}.yml"
- name: Copy our content
  copy:
    src=index.html
    dest="{{ docroot }}"
    owner="{{ docowner }}"
```

Roles

mydemocontent/vars/RedHat.yml

```
---
docroot: /var/www/html
docowner: apache
```

Roles

mydemocontent/files/index.html

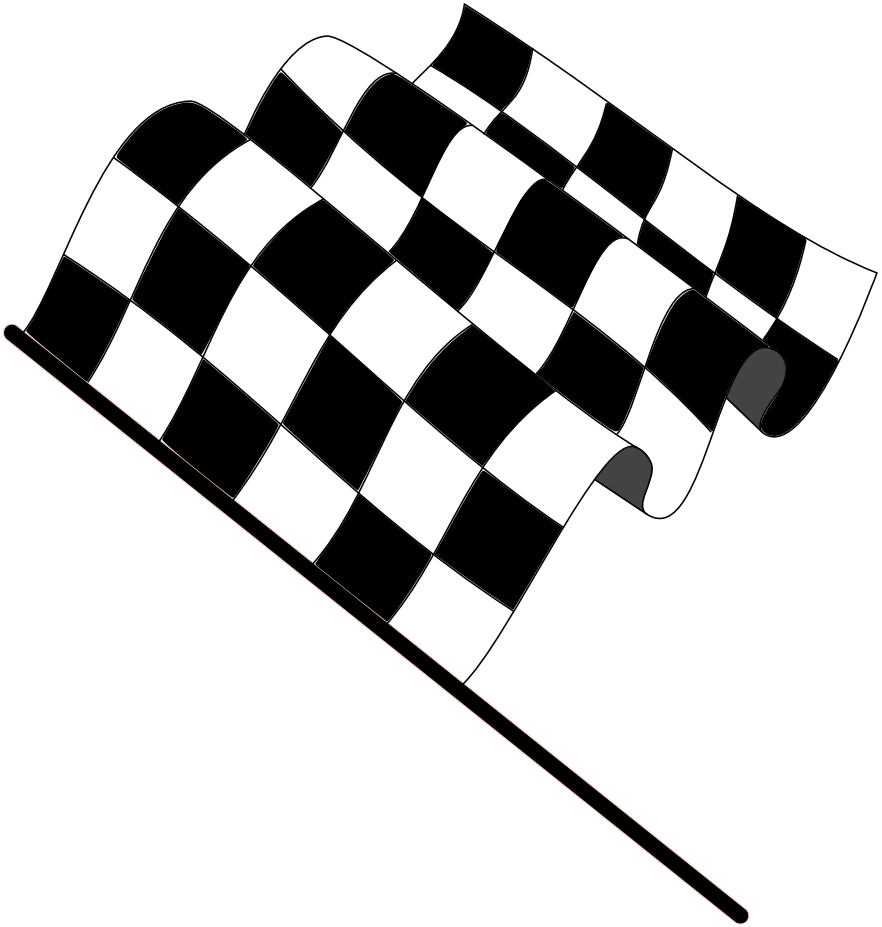
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta name="generator"
          content="HTML Tidy for Linux (vers 25 March 2009), see www.w3.org" />
  <title>Demo</title>
</head>

<body>
  <h1>Demo</h1>
  <p>Demo file placed by ansible</p>
</body>
</html>
```

Roles

Finish



Ansible galaxy

library

- Library for roles
  - https://galaxy.ansible.com/list#/roles
  - > 4000 roles
  - Not always easy to find the best role!
- Ansible-galaxy
  - install
  - init
  - ...

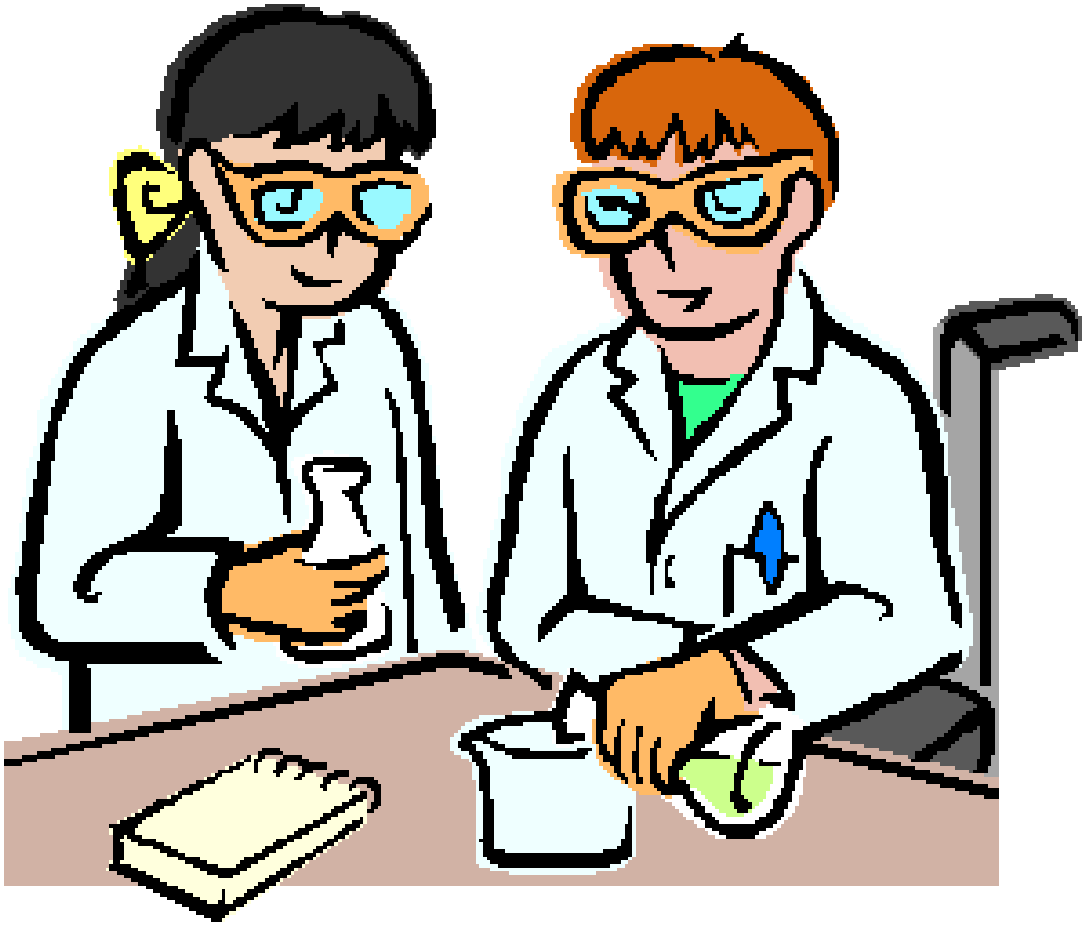
Ansible galaxy

Installing a role

- Search the website
  - marcelnijenhof.firewalld
- Install the role
  - Ansible-galaxy install marcelnijenhof.firewalld
    - Default install in /etc/ansible/roles
    - Use the “-p <dir>” for other locations

Ansible galaxy

Demo: Download a role



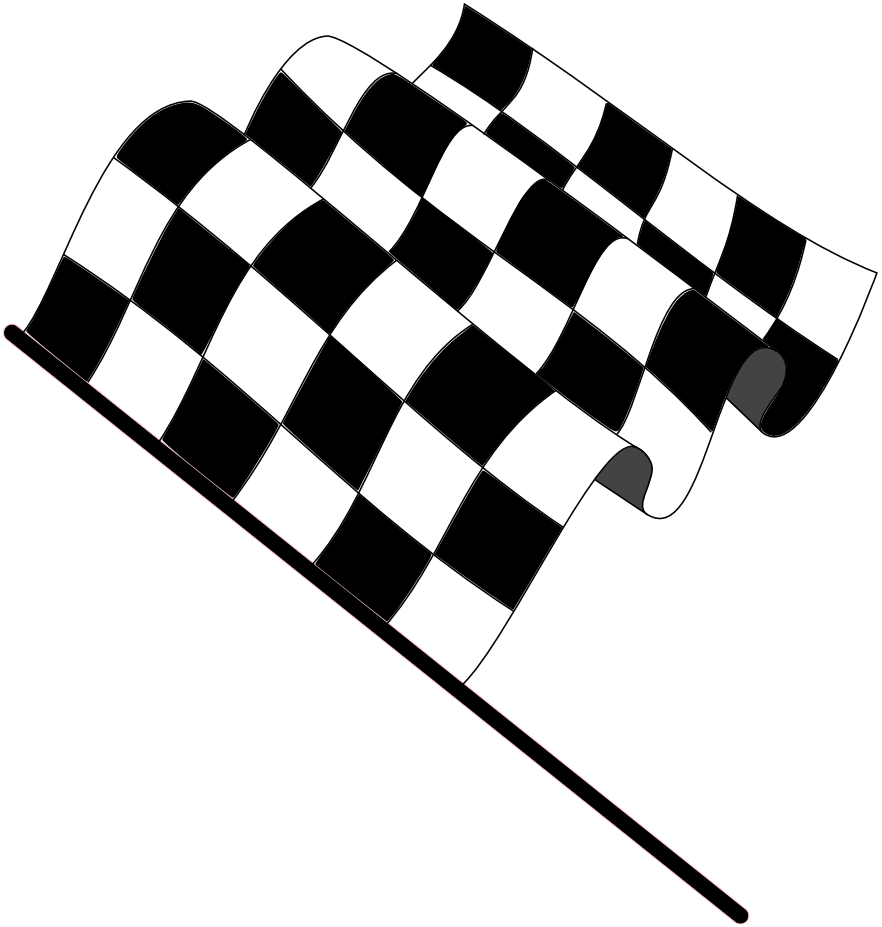
Ansible galaxy

adding your role

- Create the code
- Add it to github
- Login in galaxy with your github account
- Register your git project on galaxy

Ansible galaxy

Finish



Configuration management tool

Putting everything together

- site.yml: Includes all code for the site:
  - webservers.yml: Code for webservers
    - Include the correct roles
  - ...
- roles
- host_vars
- groups_vars

Configuration management tool

site.yml

```
---
- include: web.yml
- include: centos7.yml
```

Configuration management tool

web.yml

```
---
- hosts: web
  become: yes
  become_user: root
  roles:
    - geerlingguy.apache
    - marcelnijenhof.firewalld
    - mydemocontent
```

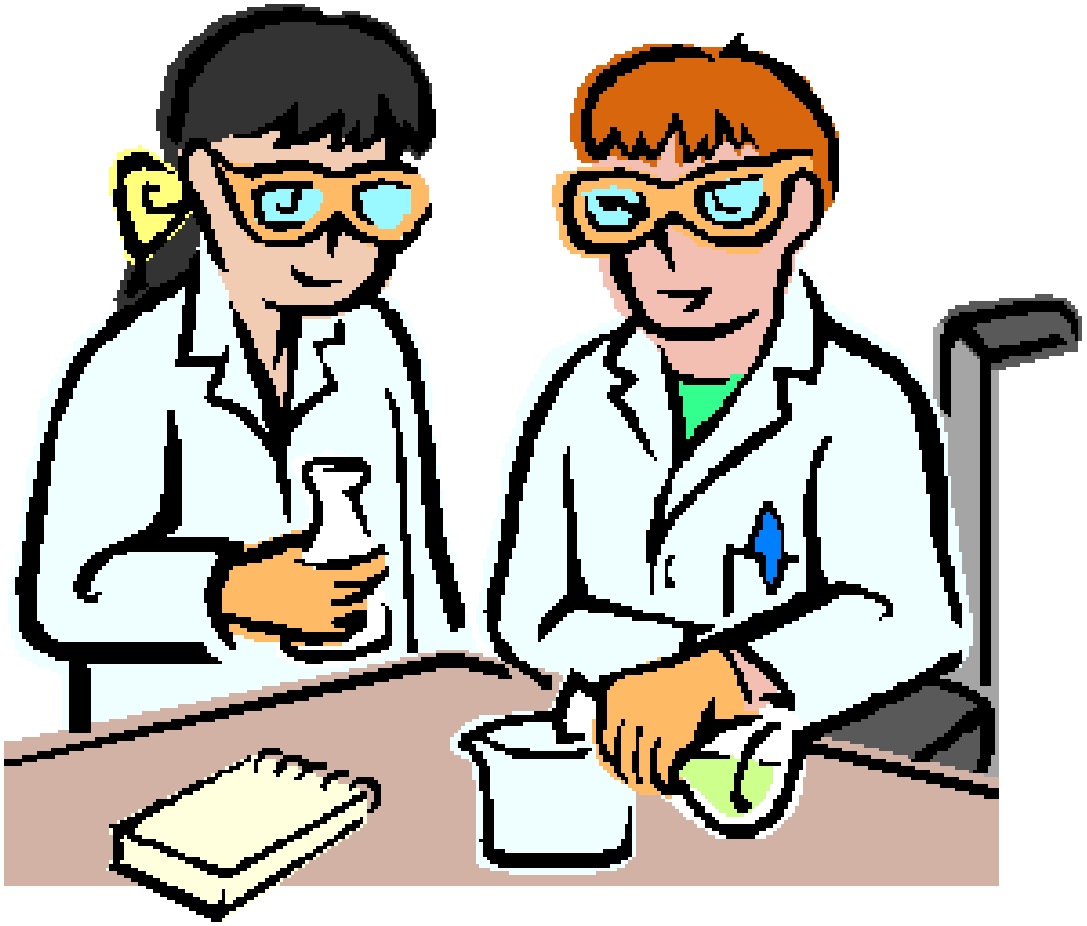
Configuration management tool

Running the code

- ansible-playbook site.yml
  - When users want
  - Through crontab by a dedicated user
  - ...

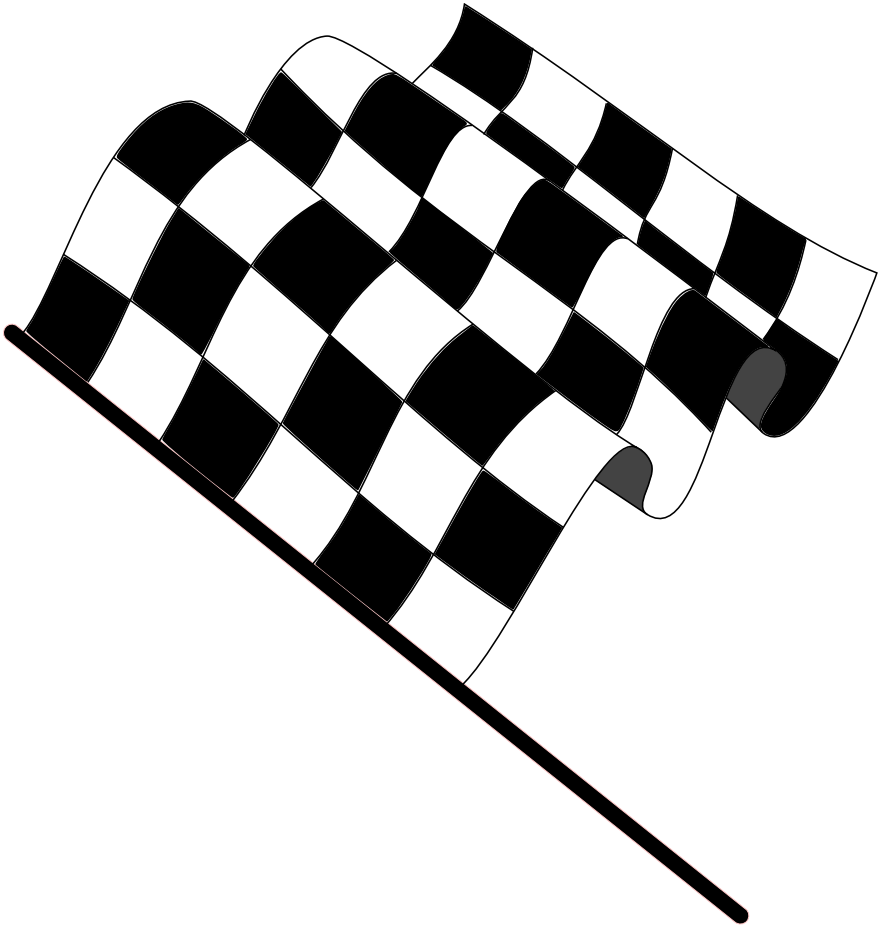
Configuration management tool

Demo: putting everything together



Configuration management tool

Finish



Add hoc usage

the reality

- Many different systems without central configuration management
  - Everything will be better after we rebuild everything
  - It's to cumbersome to implement central configuration management on old systems
- You are just hired to do one project
- But still we need more control now!!

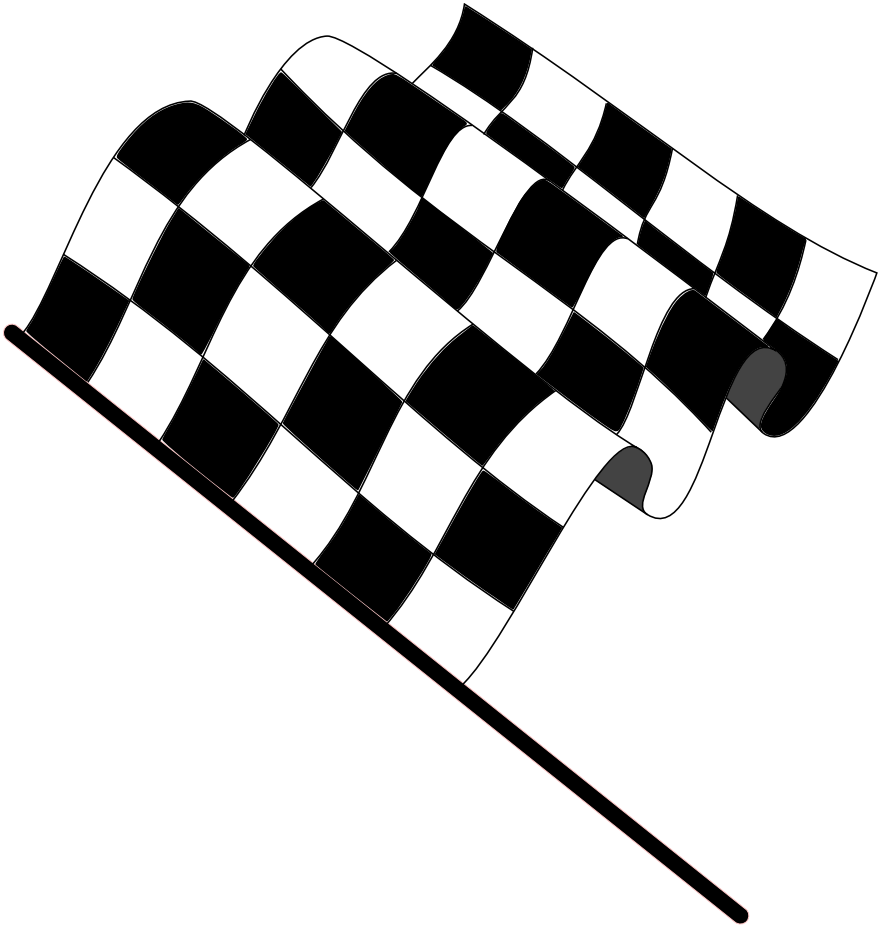
Add hoc usage

What if you just want to do one task

- Just one task
  - Add the same users on 200 systems
  - Update security settings
  - Upgrade a application
- It's possible
  - You don't need to set up anything on the client
  - Just one ansible server
    - And that could be your laptop!
  - A inventory
  - One small playbook

Add hoc usage

Finish



More info

- http://docs.ansible.com/
- http://pion.xs4all.nl/lezingen/ansible-2015.pdf

questions