Ansible: The Automation Era

Hamdi BENSALAH

linkedin

16th July, 2022

How to handle configuration management?

The challenges

Configuration Hell

A business relies on multiple systems

- Version Control Systems
- Continuous Integration
- Continuous Delivery
- Multiple environment setup
- Security configuration
- Cloud resources

The solution

Automation

At first you may think it is the way to go…

Theoretically, automation should solve the problem of multiple system provisioning since:

It saves time, make maintenance doable and offer traceability and logs.

It can be achieved with different scripting tools:

SHell, Python, Ruby, UI etc

But **practically…**

- Systems are configured differently
- No universal tools and methodologies
- Tools diversity open the door to new hell since it requires different expertise and dedicated management.

For example, we can end up with dozen of SHell and python scripts, unmaintable, not modular, can't follow business development.

A beam of light

One solution for all problems

“Ansible”

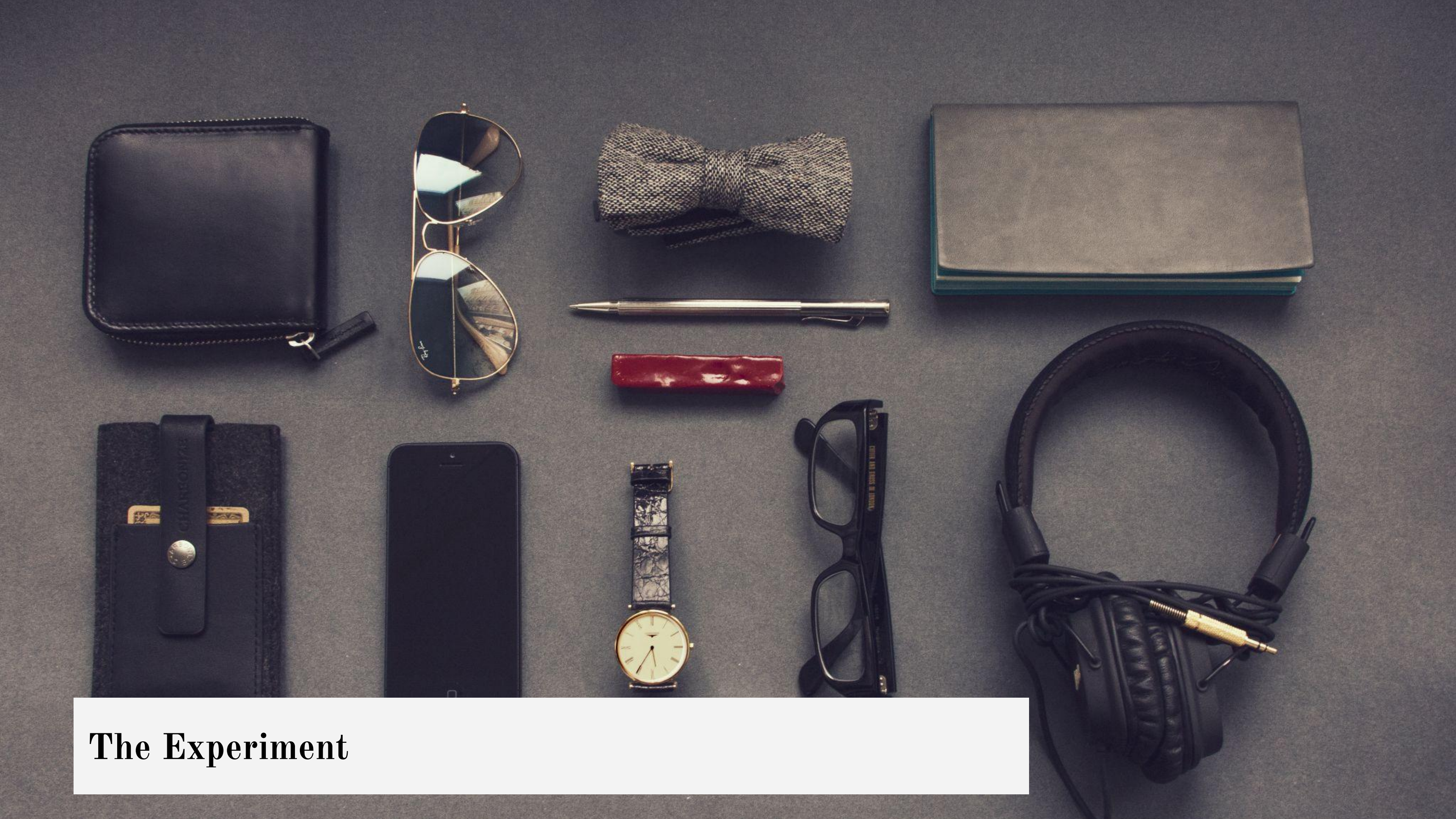
Ansible

What is it...

- OSS Scripting abstraction tool by RedHat
- Organized in **Steps**
- No agent dependency
- Mature in industry
- Playbooks/Role written in Yaml
- Short learning curve
- Reusability
- Modularity and Ansible Galaxy

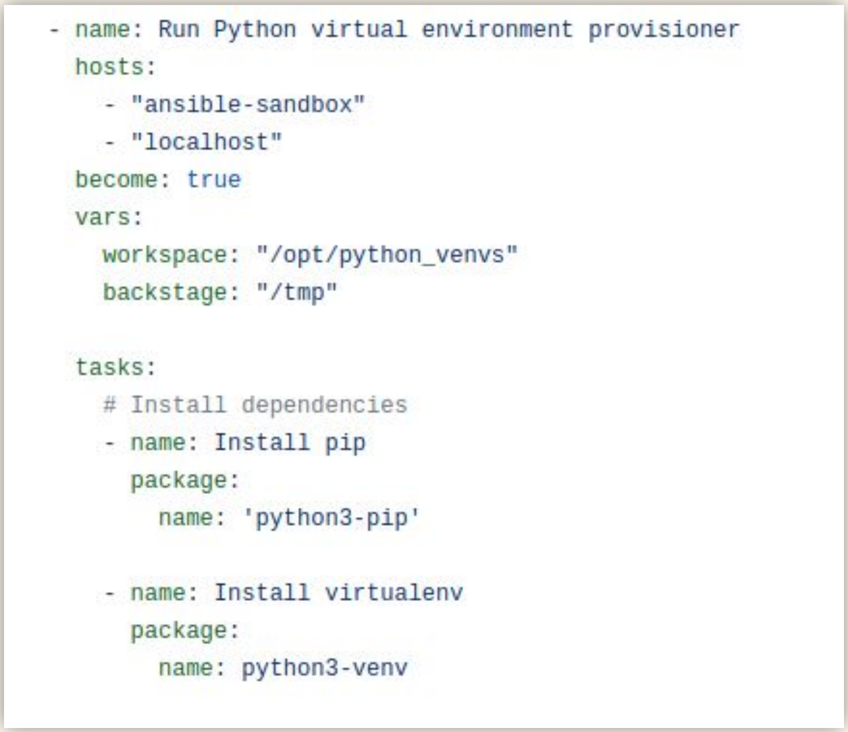
So one single way to

- Configuration Management
  Prepare dependencies, manage services...
- Application Deployment
  Deploy product in multiple nodes with Inventory
- Orchestration
  Deploy backend, frontend, configure databases...
- Security and Compliance
  Setup firewall, handle users...
- Cloud Provisioning
  Setup bare-metals nodes, configure cloud resources



The Experiment

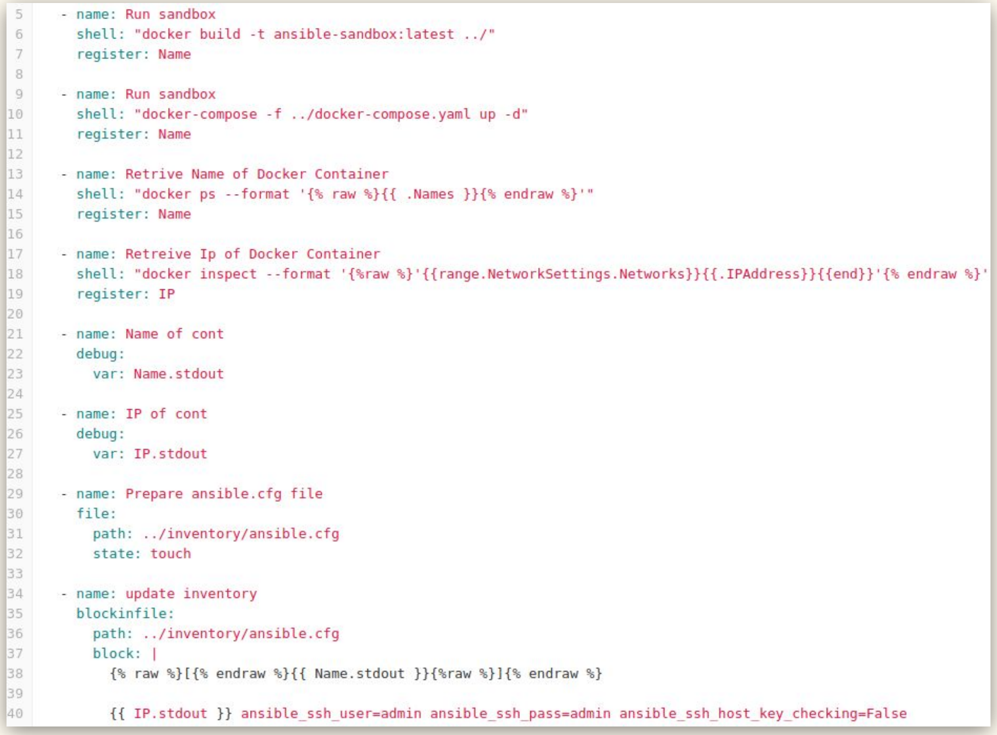
A sample playbook



```yaml
- name: Run Python virtual environment provisioner
  hosts:
    - "ansible-sandbox"
    - "localhost"
  become: true
  vars:
    workspace: "/opt/python_venvs"
    backstage: "/tmp"

  tasks:
    # Install dependencies
    - name: Install pip
      package:
        name: 'python3-pip'

    - name: Install virtualenv
      package:
        name: python3-venv
```



```yaml
  - name: Run sandbox
    shell: "docker build -t ansible-sandbox:latest ../"
    register: Name

  - name: Run sandbox
    shell: "docker-compose -f ../docker-compose.yaml up -d"
    register: Name

  - name: Retrive Name of Docker Container
    shell: "docker ps --format '{% raw %}{{ .Names }}{% endraw %}'"
    register: Name

  - name: Retreive Ip of Docker Container
    shell: "docker inspect --format '{%raw %}'{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}'{% endraw %}'
    register: IP

  - name: Name of cont
    debug:
      var: Name.stdout

  - name: IP of cont
    debug:
      var: IP.stdout

  - name: Prepare ansible.cfg file
    file:
      path: ../inventory/ansible.cfg
      state: touch

  - name: update inventory
    blockinfile:
      path: ../inventory/ansible.cfg
      block: |
        {% raw %}[{% endraw %}{{ Name.stdout }}{%raw %}]{% endraw %}

        {{ IP.stdout }} ansible_ssh_user=admin ansible_ssh_pass=admin ansible_ssh_host_key_checking=False
```

Conclusion

Ansible allowed teams and business to automate theirs infrastructure efficiently and reliably.

Beside the IAC role, Ansible presented the term of “**The Democratization of Scripting**” since it opened the door for everybody to create scripts nevertheless the level of experience, the type of the infrastructure, the used technologies and so on.