Make your environment sane with Ansible Automation

Fabio Alessandro Locati

Initial situation

- 300 GlassFish installations
- A good mix of versions 4.0, 4.1, 4.1.1
- Same(ish) application running on it
- ~250 running on 25 EC2 running CentOS 6.x (~10 instances/server)
- ~50 running on ~50 bare metal systems running CentOS 5.x (1 instance/server)
- 300 instances of MySQL running
- Thousands of scripts around (5 per instance), theoretically all copies of the same base scripts

About me - Fabio Alessandro Locati

- 15+ years in ICT, majority in infrastructure consulting
- 7+ years using Ansible
- 150+ contributions in github.com/ansible/ansible
- Author of 4 books, 3 of which on Ansible:
  - Learning Ansible
  - Learning Ansible 2.7
  - Practical Ansible
- Now working for Red Hat as Senior Solution Architect supporting Global System Integrators (GSI) partners in EMEA

The research

- Automation system that is:
  - Simple
  - Can coexist with legacy processes
  - Does not change the security model
  - Is self-documenting(ish)
  - Idempotent

Idempotence

Idempotence: is the property of certain operations in mathematics and computer science, that can be applied multiple times without changing the result beyond the initial application.

Idempotent examples:

- X = 100 (always 100)
- X = X^0 (always 1)
- echo “TEST” > /root/example

Non-idempotent examples:

- X = X*2
- echo “TEST” >> /root/example

Idempotence - tricky/edge cases

- yum update
- yum install ...
- wget ...
- echo “$x” > /root/test

Ansible

- Agent-less
- Connects to managed machines via SSH
- Does not care about the state of the rest of the system
- Applies changes in a sequential way
- It has a very gentle learning curve
- Playbooks can be easily read by non-technical people (i.e.: auditors)
- It is very simple setup
- It is a swiss-knife tool (configuration, deployment, orchestration)

Initial setup

- Create SSH keys
- Distribute SSH keys
- Create git repository
- Create inventory

How to select processes to automate

- Non critical operations
- Very well understanded operations
- Easy to test

Deploying new application servers

- Install Java
- Create the glassfish user
- Install unzip
- Download Payara
- Unarchive Payara
- Set Payara file ownership
- Create systemd unit

Examples

```
- name: Ensure we have Java installed
  yum:
    name: java-1.8.0-openjdk
    state: present

- name: Ensure that the glassfish user exists
  user:
    name: glassfish
    state: present

- name: Ensure we have unzip installed
  yum:
    name: unzip
    state: present

- name: Ensure Payara installer is present
  get_url:
    url: "https://system.otelia.eu/pms/payara-4.1.1.154.zip"
    dest: /opt/payara-4.1.1.154.zip
```

Examples

```
- name: Ensure Payara is unarchived
  unarchive:
    src: /opt/payara-4.1.1.154.zip
    dest: /opt
    remote_src: True

- name: Ensure the Payara files have the correct ownership
  file:
    path: /opt/payara41
    owner: glassfish
    group: glassfish
    recurse: True

- name: Ensure Payara service file is present
  copy:
    src: glassfish.service
    dest: /etc/systemd/system/glassfish.service
```

Some considerations

- If allowed, redesign your infrastructure while automating
- Simpler is better

How to select processes to automate - 2

- There is a boring or dirty job to do
- You have a little bit of time available to automate it

The lifecycle of a user

- Create a user on certain machines
- Add user to groups
- Add SSH keys

- Delete a user on all machines

Examples

```
- hosts: {{ tgthosts }}
  user: ansible
  tasks:
    - name: 'Ensure user {{ name }} is present'
      user:
        name: '{{ name }}'
        shell: /bin/bash
        groups: '{{ groups }}'
        uid: '{{ uid }}'
        state: present
    - name: Ensure SSH Keys are available
      authorized_key:
        user: '{{ name }}'
        key: "{{ lookup('file', name + '.pub') }}"
```

Examples

```
- hosts: {{ tgthosts }}
  user: ansible
  tasks:
    - name: 'Ensure user {{ name }} is present'
      user:
        name: '{{ name }}'
        shell: /bin/bash
        groups: '{{ groups }}'
        uid: '{{ uid }}'
        state: present
    - name: Ensure SSH Keys are available
      authorized_key:
        user: '{{ name }}'
        key: "{{ lookup('file', name + '.pub') }}"
```

```
ansible-playbook -i inventory create_user.yaml --extra-vars "name=fale
groups=wheel uid=1500 tgthosts=webservers"
```

Examples - Inventory

```
[webservers]
ws01.example.com
ws02.example.com

[jas]
js01.example.com
js02.example.com
```

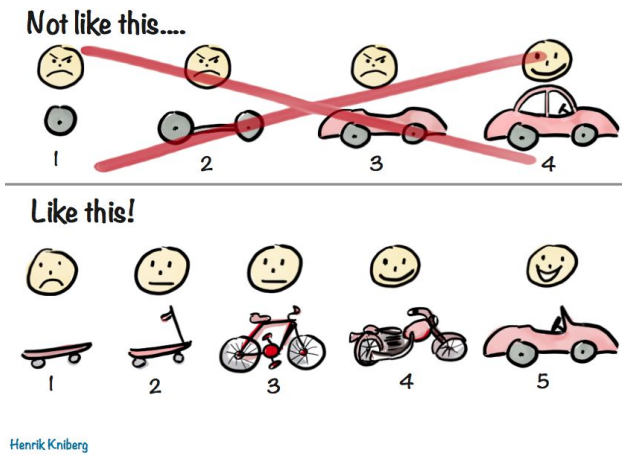
Examples

```
- hosts: all
  user: ansible
  tasks:
    - name: 'Ensure user {{ name }} is absent'
      user:
        name: '{{ name }}'
        state: absent
```

```
ansible-playbook -i inventory create_user.yaml --extra-vars
"name=fale"
```

Some considerations

- With Ansible it is easy to create “distributed bash” scripts
- Ansible will improve consistency in the environment
- Automate step by step



Red Hat

Users v2 - users.yaml

```
---
users:
  - name: fale
    uid: 1500
    groups:
      - wheel
  - name: admin
    uid: 1501
    groups:
      - wheel
```

Users v2

```
- hosts: {{ tgthosts }}
  user: ansible
  tasks:
    - name: 'Ensure user {{ item.name }} is present'
      user:
        name: '{{ item.name }}'
        shell: /bin/bash
        groups: '{{ item.groups }}'
        uid: '{{ item.uid }}'
        state: present
      with_items: '{{ item.users }}'
    - name: Ensure SSH Keys are available
      authorized_key:
        user: '{{ item.name }}'
        key: "{{ lookup('file', item.name + '.pub') }}"
      with_items: '{{ users }}'

ansible-playbook -i inventory create_user.yaml --extra-vars "@users.yaml"
```

The lifecycle of a server

- Create a machine on AWS
- Update the OS
- Harden the OS
- Configure users
- Install application
- Correctly set the load balancer/DNS/…

- Update application

- Update users

- Remove machine from load balancer/DNS/…
- Destroy machine from AWS

Some considerations

- Start with the low-hanging fruit
- Aim for complete lifecycle automation, but work towards it in a gradual way

Additional processes

- Workstation management
- Office automation
- Administrative processes automation
  - Download files from certain locations
  - Upload files to certain locations
  - Send documents via emails
- Creation and destroyment of lab environments

Summary

- Ansible can provide you with a simple way to automate processes with multiple systems involved
- Start with the low-hanging fruit
- If possible, rethink processes instead of just automating them
- Aim for complete company automation, but work towards it in a gradual way
- Automation has a huge impact on people, maybe even more than processes

"Any improvement made after the bottleneck is useless, because it will always remain starved, waiting for work from the bottleneck. And any improvements made before the bottleneck merely results in more inventory piling up at the bottleneck." - Gene Kim, The Phoenix Project

Q&A

Website: fale.io
Email: me@fale.io

Thanks

Website: fale.io
Email: me@fale.io

Additional resources

- Official documentation: http://docs.ansible.com
- Videos: https://www.ansible.com/videos
- Whitepapers: https://www.ansible.com/whitepapers
- Ebooks: https://www.ansible.com/ebooks