Make your environment sane with Ansible Automation

Fabio Alessandro Locati

November 3, 2023

EMEA Associate Principal Specialist Solutions Architect

About me - Fabio Alessandro Locati

- 20 years in ICT, majority in infrastructure consulting
- 10 years using Ansible
- Working as Associate Principal Specialist Solutions Architect at Red Hat
- Author of 4 books, 3 of which on Ansible:
  - Learning Ansible
  - Learning Ansible 2.7
  - Practical Ansible
- RHCA IV

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
- 1k+ of scripts around (5 per instance), theoretically all copies of the same base scripts

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

Automate Application Server deployment

Deploying new application servers

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

Example - jas.yaml

```
---
 - hosts: jas
   user: ansible
   tasks:
     - name: Ensure we have Java installed
       ansible.builtin.yum:
         name: java-1.8.0-openjdk
         state: present
     - name: Ensure that the glassfish user exists
       ansible.builtin.user:
         name: glassfish
         state: present
     - name: Ensure we have unzip installed
       ansible.builtin.yum:
         name: unzip
         state: present
     - name: Ensure Payara installer is present
         ansible.builtin.get_url:
         url: "https://system.example.org/pms/payara-4.1.1.154.zip"
         dest: /opt/payara-4.1.1.154.zip
 ...
```

Example - jas.yaml (cont.)

```
...
    - name: Ensure Payara is unarchived
      ansible.builtin.unarchive:
        src: /opt/payara-4.1.1.154.zip
        dest: /opt
        remote_src: True
    - name: Ensure the Payara files have the correct ownership
      ansible.builtin.file:
        path: /opt/payara41
        owner: glassfish
        group: glassfish
        recurse: True
    - name: Ensure Payara service file is present
      ansible.builtin.copy:
        src: glassfish.service
        dest: /etc/systemd/system/glassfish.service
```

Some considerations

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

Automate users

The lifecycle of a user

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

- Delete a user on all machines

User v1 - create_user.yaml

```
---
 - hosts: {{ tgthosts }}
   user: ansible
   tasks:
     - name: 'Ensure user {{ name }} is present'
       ansible.builtin.user:
         name: '{{ name }}'
         shell: /bin/bash
         groups: '{{ groups }}'
         uid: '{{ uid }}'
         state: present
     - name: Ensure SSH Keys are available
       ansible.posix.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"
```

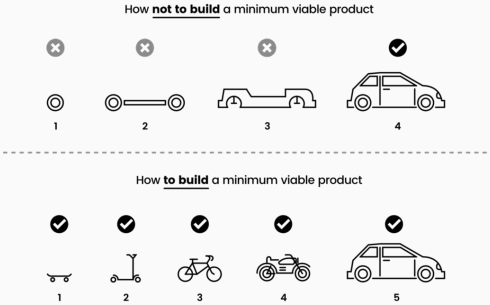
User v1 - remove_user.yaml

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

```
ansible-playbook -i inventory remove_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



Users v2 - users.yaml

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

Users v2 - create_users.yaml

```
---
 - hosts: {{ tgthosts }}
   user: ansible
   tasks:
     - name: 'Ensure user {{ item.name }} is present'
       ansible.builtin.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
       ansible.posix.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"
```

Wrapping up

Wrapping up

- Ansible can provide you with a simple way to automate distributed processes
- Start with the low-hanging fruits
- If possible, rethink processes instead of just automating them
- Aim for complete lifecycle automation, but work towards it in a gradual way
- 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

Thanks

fale@redhat.com