From Puppet to Ansible

Who am I?

- Currently Platform Engineering Manager @ ContentSquare
- Widely adopted and used Puppet in several companies
- Got troubles
- Then decided to move to Ansible
- Lots of benefits
- But still got troubles :-)

Scope: systems configuration, applications deployment

Puppet issues (1/2)

- Duplicated code in some modules
- Not-fully functionnal modules (UNIX users...)
- Implement services discovery?
- CI integration (Jenkins)
- Infrastructure resources management:
  - EC2: instances, ELB, security groups
  - Network: VPC, subnets, firewall rules
  - Route 53: domain names

Puppet issues (2/2)

- Enterprise ecosystem
- Mix of Puppet-DSL and Ruby
- Complex production setup (master, CA, agents...)
  Puppetization of PuppetMaster?
- Random ordering (soup of `require`)
- Need a custom external node

ANSIBLE

Scope: Infrastructure provisionning, systems configuration, applications deployment

Ansible

Decision to move away from Puppet was driven by:

- CI tools integration (Jenkins)
  (thanks to *Push*, not *Pull*, paradigm)
- Properly manage infrastructure resources
  (servers, load balancers, domain names, firewall rules...)
- Overall team's motivation

Examples

Time to show real-life code!

(or almost real-life)

Example 1: File (Puppet)

```
file {$myscript:
  content => template('path/to/mytemplate.erb')
  owner   => $app_user,
  group   => $app_group,
  mode    => '0755',
  require => Archive['hbase'],
}
```

Template :

```
<% myvar.each do |key, value| -%>
    <%= key %>=<%= value %>
<% end %>
```

Example 1: File (Ansible)

```
- name: "Update config file in {{ app_dir }}"
  become: yes
  template:
    src: myfile.j2
    dest: "{{ app_dir }}/application.conf"
    owner: "{{ app_user }}"
    group: "{{ app_group }}"
    state: present
  tags:
    - app
```

```
{% for key, value in myvar.iteritems() %}
        {{ key }}={{ value }}
{% endfor %}
```

Example 2: Loop (Puppet)

(Old-style syntax)

```
define puppet::binary::symlink ($pkg = $title) {
  file {"/usr/bin/${pkg}":
    ensure => link,
    target => "/opt/puppetlabs/bin/${pkg}",
  }
}

$pkgs = ['facter', 'hiera', 'mco', 'puppet', 'puppetserver']

puppet::binary::symlink { $pkgs: }
```

Example 2: Loop (Ansible)

```
- name: "Installing useful set of packages"
  apt:
    name: "{{ item }}"
    state: present
  become: yes
  with_items:
    - htop
    - curl
    - jnettop
```

Example 3: Vars management (Ansible)

```yaml
- name: "List old releases"
  become: yes
  shell: "ls -t {{ app_user.home }}/ | tail -n +5"
  changed_when: False
  register: ls_output

- name: "Remove oldest releases"
  become: yes
  file:
    name: "{{ app_user.home }}/{{ item }}"
    state: absent
  with_items:
    - ls_output.stdout_lines
```

Conclusion

Ansible concerns

Not a paradise-island...

- Variables factorization:
  Vars? Default? Group vars? Extra vars?
- Variables scope & precedence
- Fast development cycles

Benefits

- No issues with repeated instructions
- Clear dependencies management (`meta/`)
- Easy guidelines-checking
- Tags management
- Ansible Vault

Current state

- Management of all AWS resources
- Jenkins integration
- Set of blocking guidelines
- Dynamic inventory for ES/Kafka IPs
- Automatic CIDR for subnets definitions (Jinja filter)

Next steps

- Advanced tags management
- Playbooks tests