# DevOps with Ansible

## 26 August 2015

kapil.thangavelu at capitalone dot com

# Software Trends

- Clouds
- Microservices
- Containers
- Continous deployment

Lots more resources to manage on a dynamic basis

# iotw automation is mandatory

## But what do we need to automate?

# Config Management

At a high level breaks down into 3 key resources

- Service / Process management
- Files (templates, copies, attributes)
- Distribution Packages

# Application Deployment

- app source (vcs) deployment to machines
- databases
- load balancers
- message queues
- caches

# IAAS Management

- provision servers
- load balancers
- volumes
- databases
- network topology
- autoscale groups

# So what are tools

# cfg mgmt history

CFEngine (1993)
- Convergence (Describe Goal state, agents converge to it)
- Idempotence

Puppet (2005)
- Stronger resource model (files, packages, services)
- Dependency Graph sort

Chef (2009)
- Switch DSL to Ruby
- Sequential Flow

Ansible (2011)

Lots of others (Salt, bcfg2)

# Ansible

## Goals

- Simplicity / Pragmatic / Flexible

## Why

- address common concerns of multi-node deployments working in concert
- address disconnect between config tools and deployment tools
- avoiding agent issues (cve, is it up, ntp, certs/ca, resource consumption)

# Ansible

## How

- agent-less
- push based default (optional pull)
- ssh for node communications
- yaml DSL
- small core / extensible in any language

# Example

```
$ cat web.yml
```

```
- hosts: webservers
  tasks:
    - yum: name=httpd state=installed
    - template: src=httpd.conf.j2 dest=/etc/httpd/httpd.conf
    - service: name=httpd state=running
```

```
$ ansible-playbook -i myhosts web.yml
```

# Inventory

- Hosts, Groups, Variables

Can source from

- Text Files (static)
- Any CMDB
- AWS / GCE / Openstack / Digital Ocean
- Any Script that can return JSON

# Variables

- from inventory
- from host_vars, group_vars
- from register (ie. output of any task)
- facts automatically gathered from nodes (plus ohai and facter if present).
- cli variables
- lists, dicts/maps/hash table, booleans, integer, strings, etc.

# Ansible DSL / Playbook

- YAML
- Variable interpolation {{ my_var }}
- Loops (with_items)
- Conditionals (when)
- Blocks (v2)
- Notification handlers

# Ansible DSL / Playbook

- A playbook is a collection of plays
- A play is a collection of tasks

# Modules

Batteries included

Hundreds of modules with broad and deep coverage.

Unix & Windows, Load balancers, Package management, Cloud

Containers, Databases, VCS, monitoring, hipchat/irc, etc.

docs.ansible.com/ansible/list_of_all_modules.html (http://docs.ansible.com/ansible/list_of_all_modules.html)

Written in *any* language as long as it can input/output json

# Roles

Provide a standard structure for making playbooks reusable and distributable.

Ansible Galaxy provides a public distribution channel

```
- hosts: webservers
  roles:
    - base-configuration
    - apache
    - webapp

- hosts: dbservers
  roles:
    - base-configuration
    - database
```

# Roles Structures

```
/roles/webserver/
   /files/
   /tasks/
     main.yml
   /meta/
   /templates/
   /defaults/
   /handlers/
   /vars/
```

# Ad Hoc Mode

## Great for quick exploration, one off task, ie. pragmatic

```
$ ansible -i ec2.py -m ping us-east-1
 172.94.160.165 | success >> {
    "changed": false,
        "ping": "pong"
    }
 172.94.160.166 | success >> {
    "changed": false,
        "ping": "pong"
      }
 172.94.160.167 | success >> {
    "changed": false,
       "ping": "pong"
      }
```

# Application Deployment

```
# This playbook deploys the whole application stack in this site.

# Apply common configuration to all hosts
- hosts: all
  roles:
    - common

# Configure and deploy database servers.
- hosts: dbservers
  roles:
    - db

# Configure and deploy the web servers
- hosts: webservers
  roles:
    - base-apache
    - web

# Configure and deploy the load balancer(s).
- hosts: lbservers
  roles:
    - haproxy
```

# Rolling Update

```
- hosts: webservers
  remote_user: root
  serial: 1

  pre_tasks:
  - name: disable the server in haproxy
    haproxy: 'state=disabled backend=myapplb host={{ inventory_hostname }} socket=/var/lib/hapro
xy/stats'
    delegate_to: "{{ item }}"
    with_items: groups.lbservers

  roles:
  - web

  post_tasks:
  - name: wait for webserver to come up
    wait_for: 'host={{ inventory_hostname }} port=80 state=started timeout=80'

  - name: enable the server in haproxy
    haproxy: 'state=enabled backend=myapplb host={{ inventory_hostname }} socket=/var/lib/haprox
y/stats'
    delegate_to: "{{ item }}"
    with_items: groups.lbservers
```

# Provisioning

```
- hosts: localhost
  tasks:
    - name: Loading Config
      include_vars: config.yml
    - name: Provision
      ec2:
        id: "{{ provision_token }}"
        region: "{{ region }}"
        key_name: "{{ key }}"
        group_id: "{{ security_groups }}"
        instance_type: "{{ instance_type }}"
        image: "{{ image }}"
        wait: yes
        wait_timeout: 350
        count: 3
        vpc_subnet_id: "{{ subnet }}"
        instance_tags: "{{ tags }}"
      register: ec2
    - name: Wait for SSH to come up
      wait_for: host={{ item.private_ip }} port=22 timeout=320 state=started
      with_items: ec2.instances
```

# Containers

```
- hosts: myserver
  tasks:
   - name: Database
     docker:
       name: database
       image: postgres:9.4
       state: started
   - name: My application
     docker:
       name: web
       image: quay.io/myapp/minimal-sinatra:latest
       pull: always
       state: reloaded
       env:
         SOMEVAR: value
         SHH_SECRET: "{{ from_the_vault }}"
       link:
         - "database:database"
```

# Comparing to Chef

- Pull vs Push mode

```
Speed
```

- Observation

```
Are we there yet?
```

- Coordination

```
Sequenced operations across nodes
```

- Security

```
Auto Accept :-(
```

- Simplicity

# Links

## Ansible Docs

docs.ansible.com (http://docs.ansible.com)

## This Presentation

github.com/kapilt/ansible-presentation (https://github.com/kapilt/ansible-presentation)

## Michael DeHaan's Presentations

## Config Management Concepts and Futures

speakerdeck.com/mpdehaan/systems-management-concepts-and-futures

(https://speakerdeck.com/mpdehaan/systems-management-concepts-and-futures)

## Ansible Reasons for Creation, Lessons Learned, etc

speakerdeck.com/mpdehaan/etc (https://speakerdeck.com/mpdehaan/etc)

## Ansible Presentation - Origin Story / Call for simplicity

speakerdeck.com/mpdehaan/ansible (https://speakerdeck.com/mpdehaan/ansible)

# Thank you

kapil.thangavelu at capitalone dot com
http://github.com/kapilt (http://github.com/kapilt)
@kapilvt (http://twitter.com/kapilvt)