Ansible - The Future is Now!

Presentation by Norman J. Harman Jr. <njharman@gmail.com>

Code and Source: https://github.com/njharman/presentations

Who

Michael DeHann <michael.dehaan%40gmail.com>

- Created Cobbler.
- Co-author of Func, Ansible's precursor.

What

Simple, (not simplistic), Python based

- Configuration management (Chef, Puppet, cfengine)
- Process automation aka deployment (Fabric, Capistrano)
- Adhoc remote execution (mcollective, Func)

Why

Why automation at all?

Even if you only have one server...

- correctness, repeatability, confidence
- documentation ~ "code" is the best type of doc
- lazyness, agility(tm), delegation to peons
- ad-hoc remote execution changed (for the better) how I work

Why Ansible in particular?

[mostly from webpage http://ansible.github.com/]

- You have a smallish (<20 number of systems).
- You have many systems spread across many "customers".
- You have to work on systems you don't / don't wanna "own".
- Python!
- One tool, three applications
- Low, low barrier to entry
- No additional software required on client boxes
- No server or client daemons; use existing SSHd out of the box
- Supports Kerberized SSH, jump hosts, forwarding, etc
- Pluggable transports (SSH is just the default)
- Can be easily run from a checkout, no installation required
- Modules are idempotent, but you can also easily use shell commands
- Modules can be written in ANY language
- Does not have to run remote steps as root
- Source host info & variables from files or external software
- Parallel by default

Gettin It

The usual suspects pip, rpm, setup.py, make? Also, running from checkout:

```
git clone git://github.com/ansible/ansible.git
cd ./ansible
source ./hacking/env-setup
export ANSIBLE_HOSTS=~/.ansible_hosts  # or use -i (--inventory)
vim $ANSIBLE_HOSTS
```

Bam!, ready to remotely manage hosts.

Ad Hoc

Expects, rightfully so, that you are using ssh keys. But if you're lame, use -k (--ask-pass). Use --private-key=path_to_key if ssh key is in "wrong" place.

Parralizes across 10 hosts by default, alter with -f (--forks).

```
ansible local -a "lsb_release -a"

# default login is root
ansible local -u nharman        -a "echo ssh as user"
ansible local -u nharman -s     -a "echo ssh as user, sudo root"
ansible local -u nharman -U bob -a "echo ssh as user, sudo bob"
```

The above are all using the "command" module, the default. It runs command directly, not in a shell, therefor no pipes, redirections, env vars. Use "shell" for that. Even lower level is raw:

```
ansible local -m shell -a "echo 'Use me some $SHELL vars and pipes.' | cat"
ansible all -m raw -a "yum install python-simplejson"
```

More on Inventory

Select more than one, all but one, all:

```
ansible localhost:var -a "hostname"
ansible var:!bar -a "hostname"
ansible all -a "hostname"
```

Select targets by wildcard (hosts only not groups):

```
ansible ec2* -a "hostname"
```

/home/njharman/work/presentations/awpug-2012-July/playbooks/hosts

Pluggable

Native SSH vs Paramiko

Paramiko is python ssh lib, just works. Native SSH supports advanced OpenSSH features (Kerberized, jumphosts, other shenanigans).

```
ANSIBLE_TRANSPORT='ssh'  # or -c (--connection) command line arg
ANSIBLE_SSH_ARGS=''  # defaults to using ControlMaster, you want ControlMaster
```

Playbooks

Combine hosts, varibles, tasks. Used to declare configurations. Used to define series of steps, possibly involving serveral systems, specific sequence, etc.

Full power of jinja in templates.

Introspection of host included. But supports ohai, facter if you wanted to install a buch of Ruby crap on all over your servers.

Vars everywhere

- instrospected ansible_vars
- inventory, playbooks
- vars_files
- vars_prompt
- command line

<Demo Time>

- sflow/setup.yml
- nharman/setup.yml

Can run playbooks locally (not via ssh)

```
ansible-playbook playbook.yml --connection=local
```

Default Ansible is push (run from central location). Pull mode is supported via ansible-pull. Supports massive scale, automatic config/setup/deploy when server is provisioned say from AWS.

Modules

Written in any language. Return JSON. Most are idempotent (with obvious exceptions raw, shell, command).

Serveral modules ship: apt, copy, file, template, git, user, group, etc. /home/njharman/work/ansible/library

Development

Modules are copied and excuted on remote machine.

/home/njharman/work/ansible/library/time /home/njharman/work/ansible/library/time2

What Sucks

- immature, churn, hard to use moving target, maybe shiny pants
- opinionated (good unless you disagree with opinion ;)
- command line only see rundeck for how aweome alternative can be
- not enterprisey, no ACL, not complicated, doesn't use XML - probably not right tool if you have 100's, 1000's of systems and dozens of admins