Introduction to Ansible

https://github.com/ansible

https://www.ansible.com

Denice Deatrich

February 2016

- What is Ansible; Why use it?
- Idempotence
- Components & Features
  - Inventory
  - YAML
  - Commands
- Variables
- Bird's eye view
- Playbooks and Roles
- Test/production setups
- Examples
- Miscellany & Conclusions

What is Ansible?

- Ansible is an open-source configuration management tool for system managers
  - written in Python and implements commands via python modules (468 modules at Ansible version 2.x)
  - works over ssh
  - does not use any daemons
  - normally used in 'push' mode from one server to multiple nodes
  - can also work in 'pull' mode from a client (Kickstart!)
  - uses an inventory of nodes and playbook files written in YAML (YAML Ain't Markup Language)
- Compare to other open-source configuration management suites: Chef, CFEngine, LCFG, Puppet, Quattor, Salt, Spacewalk, etc.

Module Groupings

Module Index (Core and Extra)

- All Modules
- Cloud Modules
- Clustering Modules
- Commands Modules
- Database Modules
- Files Modules
- Inventory Modules
- Messaging Modules
- Monitoring Modules
- Network Modules
- Notification Modules
- Packaging Modules
- Source Control Modules
- System Modules
- Utilities Modules
- Web Infrastructure Modules
- Windows Modules

Why use Ansible?

- We want to manage *change* better
- Good Change Management tools are useful for system administration. They should provide or promote features like:
  - revision history
  - accountability
  - self-documenting
  - minimalization of syntax or command-line mistakes
  - better global reliability and efficiency
  - map from a testing environment to production
  - allow change roll-back
- The tools should not impede progress or be difficult to learn

Why use Ansible chez nous?

- Ansible seems like a good fit for the Tier-1 data centre:
  - an administrator only needs to learn a bit of YAML syntax
  - you do not need to know Python, unless you want to write custom modules – even then, Ansible allows other programming languages for that purpose
  - there are no daemons to maintain
  - there are no database back-ends
  - management is driven from an admin node, pushing changes to client nodes (much as we use *pdsh* now)
- You will need to use Git if you develop playbooks (but it won't kill you, as my Mother would say..)
- Becauses it is 2016 (and we need to manage change better)

Property of Idempotence

What does this unwieldy word mean?

*https://en.wikipedia.org/wiki/**Idempotence***

***Idempotence** (/ˌaɪdɪmˈpoʊtəns/ EYE-dəm-POH-təns) 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...*

In other words, an Ansible operation can be run over and over again, and it will not introduce an undesireable side-effect. Typically this means that change commands are only applied as needed, and re-running an operation will not trigger another change.

Have you ever tried to write a shell script which will do some operations like: install software, edit configuration file, restart service (if needed) … and then make that script safe to be run over and over again without bleeping out errors or warnings, yet not unnecessarily change configurations or restart services?

Ansible Versions

- The Tier-1 is using a patched version of Ansible 1.9.2
  - I built an RPM containing some fixes from Ansible v. 2:
    - transplanted the version 2 *yum* module which supports an 'excludes' option, important for the Tier-1
    - fixed a bug in the version 1.9.x *synchronize* module
    - enhanced the *copy* module to preserve timestamps by default (it just uses 'scp' so I added the '-p' options to the command)
- EPEL now has ansible-1.9.4 in the mirror
- Next version in the mirror will be 2.0.x (see next slide)
- If you are supporting RHEL 5 clients then you must install python-simplejson on those clients. Ansible works with contemporary RHEL versions, Fedora, most Linux distros (need Python 2.x interpreter installed), BSDs, OS X and Windows

Note upcoming version 2.0.0.x

Date: Mon, 18 Jan 2016 14:23:05 -0700

From: Kevin Fenzi <kevin@scrye.com>

Subject: EPEL-ANNOUNCE ansible 2.0 in epel

I thought I would send out an email with the plans for epel and ansible 2.0.

Since ansible 2.0 is (to a pretty high degree) compatible with the same playbooks as 1.9.x, we are going to just update the existing ansible package rather than create a parallel installable one.

Note that while playbooks should be compatible and require no changes, if you are using the ***ansible API, you will need to adjust your scripts/plugins***. See:

https://docs.ansible.com/ansible/developing_api.html

(The ansible api is not guaranteed stable between any versions)

To allow for additional testing and folks to check/port their scripts/plugins we are going to leave the ansible 2.0.0.x packages in epel-testing for a longer than normal time period. We will send another notice before the 2.0.0.x version goes to stable. Currently version 2.0.0.2 is available in epel6 and epel7 ***testing repositories***.

Components of Ansible

- A configuration file, *ansible.cfg*
  - a text file which specifies locations of roles and inventories, and configures runtime behaviour of commands.
  - Usually found in /etc/ansible/ ; but can exist anywhere in the Ansible working directories for override purposes
- An inventory
  - a text file listing hostnames usually grouped by functionality
- A few commands and a library of hundreds of modules:
  - ansible, ansible-playbook, ansible-vault, ansible-doc, ansible-pull and ansible-galaxy
- Playbooks and Roles (which you will write)
- A back-end revision control system is recommended
  - Git (others supported: subversion, hg, bzr, github_hooks)

Tier-1 ansible.cfg differences

```
# diff ansible.cfg ansible.cfg.1.9.2 | grep '<'
< inventory      = /ks/ansible/inventory/production
< #remote_tmp     = $HOME/.ansible/tmp
< remote_tmp = /var/tmp/.ansible
< #forks          = 5
< forks          = 32
< #transport      = smart
< transport      = ssh
< remote_port    = 22
< roles_path    = /ks/ansible/roles
< host_key_checking = True
< log_path = /var/log/ansible.log
< nocows = 1
< #fact_caching = memory
< fact_caching = jsonfile
< fact_caching_connection = /var/cache/ansible
< fact_caching_timeout = 86400
< retry_files_enabled = False
< ## removed -o ControlPersist=60s
< ssh_args=-o ControlMaster=auto -o ControlPath=/tmp/ansible-ssh-%h-%p-%r
< scp_if_ssh = True
```

```
 _________________
< Ansible is Fun! >
 -----------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Inventory Properties

- An inventory file is a list of hosts and groups in *INI-like* format
- Hosts can be grouped, and hosts can appear in more than one group
- There is a special group called '*all*', which is assigned by Ansible – of course, it holds a list of all hosts in the inventory
- Since the Tier-1 already uses pdsh environment variables for host grouping, we use the same names (without the n_ or nc_ prefixes) for groups in our inventory file
- You can use self-explanatory alphanumeric patterns to specify a large range of hostnames
- You can create groups of groups
- You can embed 'variables' in the file, but better to put them elsewhere (see ahead)

Inventory snippets

```
[dchead]
srm
dcache
dcache-acc
hsmhead
proxy
proxy2
```

```
[dpool]
dpool[01:08]
dpool[15:30]
zpool[03:11]
zpool[13:16]
zpool[18:21]
```

```
## 425 worker nodes
[workers]
wn[028:173]
wn[175:453]
```

```
[hsmpool]
hsmpool[01:12]

## group of groups named 'storage'
[storage:children]
dpool
hsmpool
```

YAML Syntax I

- YAML is a simple list-oriented data structure
- YAML files can optionally begin with **---** and end with **...**
- All members of a list are lines beginning at the **same *indentation level***, starting with a "- " (a dash and a space)
- A list:
  - can be comprised of single items
  - can contain a list of key:value pairs (commonly called a “hash” or a “dictionary”)
- key/values pairs are separated with a *colon and a manditory space*, or sometimes an *equals sign* separates key/value pairs
- Dictionaries and lists can also be represented in an abbreviated form (example from docs.ansible.com)

```
employees:
  -  martin: {name: Martin D'vloper, job: Developer, skill: Elite}
fruits: ['Apple', 'Orange', 'Strawberry', 'Mango']
```

YAML Syntax II

- There is also a 'folded scalar' syntax (using the '>' character) allowing arguments to a module to appear on succeeding lines
- boolean values (true/false) are valid in several forms: yes, no, True, false, TRUE
- There are some special characters that you need to be careful about, especially inside of strings – “quote them”:
  [] {} : > |
- A good reference on YAML best practices with Ansible is at:
  http://www.jeffgeerling.com/blog/yaml-best-practices-ansible-playbooks-tasks
- Another good reference on YAML syntax is at:
  https://gist.github.com/halberom/82d1280d14d85d04e956

YAML Example

```
---
# This playbook uses the ansible scheduler to allow you to fork off
# a set number of rdiff-backup runs against an array of backup targets
# Taken from: https://infrastructure.fedoraproject.org/cgit/ansible.git
# See the Ansible Glossary for help with some directives or key words
- name: rdiff-backup
  hosts: backup_clients
  user: root
  gather_facts: False
  serial: 10

  vars:
  - global_backup_targets: ['/etc', '/home']

  tasks:
  - name: run rdiff-backup hitting all the global targets
    local_action: "Some ghastly command”[1]
    with_items: global_backup_targets
    when: global_backup_targets is defined
```

[1] "shell rdiff-backup --remote-schema 'ssh -p {{ ansible_ssh_port|default(22) }} -C %s rdiff-backup --server' --create-full-path --print-statistics {{ inventory_hostname }}::{{ item }} /fedora_backups/{{ inventory_hostname }}/`basename {{ item }}` | mail -r sysadmin-backup-members@fedoraproject.org -s 'rdiff-backup: {{ inventory_hostname }}:{{ item }}' sysadmin-backup-members@fedoraproject.org"

Ansible Commands I

- ansible
  - used to run ad-hoc commands

```
ansible <host-pattern> [-f forks] [-m mod_name] [-a args]
$ ansible t1-pps10 -m setup | less
$ ansible work3 -m ping
$ ansible ppsworkers -a “/bin/echo hello world”
```

- ansible-doc
  - shows documentation on Ansible modules

```
$ ansible-doc yum
$ ansible-doc ping
$ ansible-doc template
$ ansible-doc -l
```

- ansible-galaxy
  - manage roles using galaxy.ansible.com
    Galaxy is Ansible’s official community hub for sharing Ansible roles

Ansible Commands II

- ansible-playbook
  - run an ansible playbook

```
ansible-playbook <filename.yml> … [options]
$ ansible-playbook playbooks/testrole.yml \
  -e "target='t1-pps10' in_kickstart=True"
$ ansible-playbook playbooks/yum/update/do-update-glibc-only.yml \
  -e "target='pps'"
```

- ansible-pull
  - set up a remote copy of ansible (from a repository) on a managed node

```
ansible-pull -d some_path -U URL [options] [<filename.yml>]
$ ansible-pull -d /var/tmp -U git://git.lcg.triumf.ca/ansible
```

- ansible-vault
  - manage encrypted YAML data

```
ansible-vault [create|decrypt|edit|encrypt|rekey|view] \
 [options] file_name
$ ansible-vault –vault-password-file=/some/secret_file \
  encrypt root_pass
$ ansible-vault view root_pass ### you are prompted for the password
```

Variables

- Ansible variables are used to deal with differences between systems
- Variable names are composed of letters, numbers and underscores, and must always start with a letter
- Variables can occur:
  - in the inventory; ansible knows to look for variable files (in yaml format) under the inventory sub-directories:
    - ***group_vars/*** – variable files named after groups of hosts
    - ***host_vars/*** – variable files named after hosts
  - in playbooks, and/or from 'include' files
  - from 'facts' – see: $ ansible-doc setup
  - from the results of some command; known as *registered* variables
  - on the command-line

Bird's eye view of working dir.

```
# cd /ks/ansible
# ls -a
./   .git/       .gitignore  Archive/    library/    roles/
../  .githooks/  .gitmeta    inventory/  playbooks/  scripts/
# cat .gitignore
Archive
*~
.*.swp
.gitvault
ansible.cfg*
# ls inventory/
group_vars/  host_vars/  production testing  secrets/
# ls inventory/group_vars/
adm            gridmon    panglia     sb       work14  work22  work3
all            hsmpool    pooldcs     storage  work15  work23  work30
bdii           kvm-iscsi  poolddn     tr-orac  work16  work24  work31
...
# ls playbooks/yum/
check/  share/  update/
# ls roles/
common/   ks/         logs/  ntpd/        root/  syslog/   unpriv/
ganglia/  logrotate/  mail/  pdsh-hosts/  ssh/   tcpwrap/  yaim/ ...
# ls roles/common/
files/  meta/  tasks/
# ls -a inventory/secrets/ | head -4
./               roc-nagios-sb/  wn030/  wn119/  wn209/  wn298/  wn387/
../              roc-policy/     wn031/  wn120/  wn210/  wn299/  wn388/
.gitvault        roc-policy-sb/  wn032/  wn121/  wn211/  wn300/  wn389/
build/           roc-sam/        wn033/  wn122/  wn212/  wn301/  wn390/
```

Playbooks and Roles

- A *playbook* is a yaml file that lists one or more plays to run against a host or group of servers. Each play might be a list of one or more 'tasks'. A playbook might also invoke one or more 'roles'; it might also include other playbooks.
- A *role* is a set of files that encapsulate a theme – e.g. - apache-server, or client-ssh-setup, or create-users.
- A role is comprised of at least a tasks/main.yml file, but might contain one of more of:

```
tasks/              #
    main.yml        #  <-- tasks file can include smaller files if warranted
handlers/           #
    main.yml        #  <-- handlers file
templates/          #  <-- files for use with the template resource
    eg.conf.j2      #  <------- templates end in .j2 (Jinja2 Python template language)
files/              #
    bar.txt         #  <-- files for use with the copy resource
    foo.sh          #  <-- script files for use with the script module
vars/               #
    main.yml        #  <-- variables associated with this role
defaults/           #
    main.yml        #  <-- default lower priority variables for this role
meta/               #
    main.yml        #  <-- role dependencies
```

Playbook command-line options

- There are some command-line options that can be helpful:
  - `--check`
    - – don't make any changes; instead, try to predict some of the changes that may occur
  - `--syntax-check`
    - – perform a syntax check on the play, but do not execute playbook
  - `--list-hosts`
    - – output list of matching hosts; do not execute playbook
  - `--list-tasks`
    - – list all tasks which would be executed
  - `--skip-tags=some_tag`
    - - only run plays and tasks which exclude these tags

Notes about playbooks

- So far for our data centre, there are 3 general kinds of playbooks:
  - Playbooks used during installation that describe a node mostly by enumerating a bunch of roles
  - Playbooks used to invoke a role for reconfiguration; e.g. reconfigure ntpd globally, or update iptables on one node
  - Playbooks used for miscellaneous maintenance; e.g. patch nodes, or restart public-facing services
- As the implementation evolves there will be other kinds

Test / Production setups

- The model the Tier-1 is using:
  - One node is the *production* node – no development is done on this node: gridadm:/ks/ansible/
    - The config file is /etc/ansible/ansible.cfg
    - It uses the production inventory by default
    - Any 'git pulls' into this tree are done carefully and are logged in the logbook. An admin can assume that this tree is always in working order
  - Another admin node is used for *development* work: gridbackup:/ks/git/your_name/ansible/
    - Use `/root/bin/set_up_ansible_workplace.sh your_name` to set up a local working copy; see recipe: https://gridweb.triumf.ca/recipes/showentry.php?rid=264
    - The config file is local at /ks/git/your_name/ansible.cfg
    - It uses the testing inventory by default
    - Let me know when you have pushed updates to Git

Example playbook: for Kickstart

```
---
# Playbook for generic kvm virtual host
- name: Set up a KVM guest host
  hosts: "{{target}}"
  gather_facts: true
  user: root

  roles:
  - ssh
  - logs
  - pdsh-hosts
  - unpriv
  - root
  - yaim
  - ks
  - ganglia
  - mail  ## and so on
```

- invoke in kickstart '%post' section with:

```
ansible-pull -d /var/tmp -U git://git.lcg.triumf.ca/ansible
cd /var/tmp/ansible; ## fiddle with timestamps, inventory location, etc.
export ANSIBLE_ROLES_PATH=/var/tmp/ansible/roles
ansible-playbook playbooks/vmguest.xml \
   --vault-password-file=/root/.ssh/.gitvault -c local \
   -e "target=$this_shost ansible_dir=/var/tmp/ansible in_kickstart=true"
```

Example playbook: Update glibc

```
---
# Usage: ansible-playbook THIS_FILE -e "target='host1;host2'"
- name: Do only glibc updates, excluding other updates
  hosts: "{{target}}"
  gather_facts: true
  user: root
  tasks:
  - name: Run updates on SL 6 systems
    yum: >
      name='glibc,glibc-devel,glibc-headers'
      enablerepo=sl,sl-errata
      disablerepo=*
      state=latest
    when: ansible_distribution== "Scientific" and ansible_distribution_major_version|int <= 6
  - name: Run updates on Red Hat 6 systems
    yum: >
      name='glibc,glibc-devel,glibc-headers'
      enablerepo=rhel-6-server-rpms
      disablerepo=*
      state=latest
    when: ansible_distribution == "RedHat" and ansible_distribution_major_version|int <= 6
  - name: Run updates on SL7 systems
    yum: >
      name='glibc,glibc-devel,glibc-headers,glibc-static'
      enablerepo=sl,sl-errata
      disablerepo=*
      state=latest
    when: ansible_distribution_major_version|int > 6
```

Example playbook: Check updates

```
---
# Usage: ansible-playbook THIS_FILE -e "target='host1;host2'"

- name: Check for kernel updates only
  hosts: "{{target}}"
  gather_facts: false
  user: root

  tasks:
  - name: Run check
    shell: '/usr/bin/yum list available kernel warn=no || test $? -eq 1'
    register: yumoutput
    ignore_errors: true
    changed_when: no

  - name: List of available updates
    debug: var=yumoutput.stdout_lines
    when: "'kernel' in yumoutput.stdout"
```

Example role: ganglia client I

```
---
## Global ganglia configuration: ganglia/tasks/main.yml
- name: Tasks for gmond
  hosts: "{{target}}"
  user: root

  include: gmond-config.yml
  tags:
  - gmond
```

```
---
## Global ganglia configuration: ganglia/handlers/main.yml
- name: Restart gmond
  service: name=gmond state=restarted
  when: in_kickstart == false
```

```
[ inventory]# grep gr_ext group_vars/* | head -5
group_vars/adm:gr_ext: 'adm'
group_vars/all:gr_ext: 'default' ## file extension for default config files
group_vars/dchead:gr_ext: 'dcache'
group_vars/dpool:gr_ext: 'dcache'
group_vars/hsmpool:gr_ext: 'hsm'
```

Example role: ganglia client II

```
---
## Global ganglia configuration: ganglia/tasks/gmond-config.yml

- name: Install ganglia client
  yum: pkg=ganglia-gmond state=installed

- name: Be sure gmond is enabled on boot
  service: enabled=yes name=gmond

- name: Copy in /etc/ganglia/gmond.conf
  copy: >
    src={{item}}
    dest=/etc/ganglia/gmond.conf
    owner: root
    group: root
    mode: '0644'
  with_first_found:
    - files:
      - etc/ganglia/gmond.conf.{{ gr_ext }}
      - etc/ganglia/gmond.conf.pps
  notify:
    - restart gmond
```

Example role: ganglia client III

```
[ roles]# ls ganglia/files/etc/ganglia/
gmond.conf.adm      gmond.conf.work15  gmond.conf.work3
gmond.conf.dcache   gmond.conf.work16  gmond.conf.work30
gmond.conf.hsm      gmond.conf.work17  gmond.conf.work31
gmond.conf.midware  gmond.conf.work18  gmond.conf.work32
gmond.conf.nat      gmond.conf.work19  gmond.conf.work33
gmond.conf.ora      gmond.conf.work20  gmond.conf.work34
gmond.conf.pps      gmond.conf.work21  gmond.conf.work35
gmond.conf.roc      gmond.conf.work22  gmond.conf.work4
gmond.conf.sb       gmond.conf.work23  gmond.conf.work5
gmond.conf.vm       gmond.conf.work24  gmond.conf.work6
gmond.conf.work10   gmond.conf.work25  gmond.conf.work7
gmond.conf.work11   gmond.conf.work26  gmond.conf.work8
gmond.conf.work12   gmond.conf.work27  gmond.conf.work9
gmond.conf.work13   gmond.conf.work28
gmond.conf.work14   gmond.conf.work29
```

Miscellaneous notes

- Privilege escalation: *become*
  - at this time I have set up our Ansible environment to expect root to run commands, but there is the option of implementing privilege escalation; see ansible.cfg:

```
[privilege_escalation]
#become=True
#become_method='sudo'
#become_user='root'
#become_ask_pass=False
```

- jinja-2 variable {{}} format
  - When you use a variable in an Ansible yaml file it is *evaluated* when it appears in double curly brackets; e.g.

```
--enablerepo=pg{{ pg_major_ver }}{{ pg_minor_ver }}
```

  - It doesn't matter if you use spaces around the variable name
  - YAML syntax requires that if you *start* a key value with `{{ foo }}` you quote the whole line, since it wants to be sure you aren’t trying to start a YAML dictionary

Miscellaneous notes – debugging

- debugging
  - there is a debug module; however it is a bit ugly

```
[ ansible]# grep debug playbooks/testrole-debug.yml
  - debug: var=ansible_default_ipv4
```

```
GATHERING FACTS ***************************************************************
ok: [gridvm2]

TASK: [debug var=ansible_default_ipv4] ****************************************
ok: [gridvm2] => {
    "var": {
        "ansible_default_ipv4": {
            "address": "206.12.1.116",
            "alias": "br0",
            "gateway": "206.12.1.252",
            "interface": "br0",
            "macaddress": "34:40:b5:ac:61:fc",
            "mtu": 9000,
            "netmask": "255.255.255.0",
            "network": "206.12.1.0",
            "type": "bridge"
        }
    }
}
```

Miscellaneous notes - Issues

1. Only encountered a few times: get_mount_facts timeout
   https://github.com/ansible/ansible/issues/10779

```
GATHERING FACTS ***************************************************************
failed: [t1-webmon] =>
{"cmd": "/bin/lsblk -ln --output UUID /dev/mapper/vg0-data", "failed": true, "rc": 257}
msg: Traceback (most recent call last):
File "/var/tmp/.ansible/ansible-tmp-1453921386.73190028277031634/setup",
 line 1677, in run_command rfd, wfd, efd = select.select(rpipes, [], rpipes, 1)
File "/var/tmp/.ansible/ansible-tmp-1453921386.73-190028277031634/setup", line 1807,
 in _handle_timeout  raise TimeoutError(error_message)
TimeoutError: Timer expired
```

2. disk space consumption by yum option 'update_cache=yes'
   I turned this option off, because it eats disk space in /var/

3. Issues with 1.9.2 previously mentioned on Slide 7 (Ansible versions)

Example script output

Next I show some example output from playbook runs. The colour output is useful, so it is best to have a dark terminal background.

Colour output meanings:

green – success ; red – failure ;yellow – something changed ; cyan - skipped

```
# ansible-playbook --check playbooks/testrole.yml -e "target='vmtest1;wn021;pps04'"

GATHERING FACTS ***************************************************************
ok: [wn021]
ok: [pps04]
fatal: [vmtest1] => SSH Error: ssh: connect to vmtest1 port 22: No route to host..
...
TASK: [autofs | Modify /etc/sysconfig/autofs file] ****************************
changed: [wn021]
ok: [pps04]

TASK: [autofs | Copy in /etc/auto.master file on servers] *********************
skipping: [wn021] =>
 (item=/ks/git/denice/ansible/roles/autofs/files/etc/auto.master.servers)
ok: [pps04] =>
 (item=/ks/git/denice/ansible/roles/autofs/files/etc/auto.master.servers)

PLAY RECAP ********************************************************************
pps04                      : ok=7    changed=0    unreachable=0    failed=0
vmtest1                    : ok=0    changed=0    unreachable=1    failed=0
wn021                      : ok=8    changed=1    unreachable=0    failed=0
```

Finally ..

```
Remember what the cow says:

 _____________________
< Ansible is Fun! >
 ---------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Though in the early stages, we anticipate that Ansible will be useful, and should also be fun to implement.

Not everything will be ansible-ized; however daily maintenance on dozens of services and hundreds of worker nodes will be easier and more efficient using Ansible.

TRIUMF-only access to these resources:

```
$ git clone git://git.lcg.triumf.ca/ansible
```

https://gridweb.triumf.ca/documents/System/Ansible_presentation.pdf