binlogic DataOps

" A journey between **infrastructure** and **data** "

JUNE 20-21 2019 WTCBCN

Ivan Groenewold

Pythian

Automating MySQL with Ansible

Agenda

- About me
- What is Ansible?
- Why Ansible?
- Installing & configure Ansible
- Using Ansible
  - Ad-hoc mode
  - Playbooks
  - Upgrade MySQL
  - Streaming Backup/restore
- Q&A

About

@igroenew

Lead DB Consultant @ Pythian - OSDB practice

Based in Argentina

Pythian's 400+ IT professionals help companies adopt and manage disruptive technologies to better compete

What is Ansible?

What is Ansible?

- Automation engine
- SSH-based
- Open source
- Paid support option

What is Ansible?

- AWX
  - Web interface for Ansible
  - REST API
  - Open source
  - https://github.com/ansible/awx
- Ansible Tower
  - AWX-based (license required)

Why Ansible?

Why Ansible?

- Easy to deploy
- No agent required
- No firewall rules required
- YAML
- Secure

Installing & configuring Ansible

Installing Ansible

- Control machine
  - Can be your laptop
  - Ansible “server”
- Managed nodes
- Requirements
  - Python 2.7 or 3.5+
  - SSH

Installing Ansible

- Package manager
  - Check EPEL repo for the latest and greatest
- pip
- Build from source
- https://github.com/ansible/ansible

Configuring Ansible

- Generate ssh key pair
- Configure passwordless ssh
  - Add public key to .ssh/autorized_keys on each host
- Keep private key in the control machine
- Review ansible.cfg
  - `host_key_checking = False`
  - `timeout = 20`

Configuring Ansible

- Running Ansible through a jump host
- Modify ~/.ssh/config on the control machine

```
User igroenewold

Host bastion1.example.com
    Hostname 1.2.3.4
    ProxyCommand none

Host *.example.com
    StrictHostKeyChecking no
    IdentityFile ~/.ssh/id_rsa
    ProxyCommand ssh bastion1.example.com nc %h %p
```

Configuring Ansible

- Ansible Inventory
  - Static
    - ini format
    - YML format
  - Dynamic
    - Scripts available for most cloud providers
    - Write your own

Configuring Ansible

- Static inventory example

```
[webservers]
www.myhost.com
www.example.com
[databases]
db-[a:f].example.com
[atlanta]
dba.example.com http_port=80 maxRequestsPerChild=808
[atlanta:vars]
ntp_server=ntp.atlanta.example.com
```

Group name

Host vars

Group vars

Using Ansible

Using Ansible - ad-hoc commands

- Run simple commands across multiple hosts
  - `ansible [host-pattern] -a [command] --limit [expr]`
- The first filter applied is `host-pattern`
- Use `--limit` to further refine
- Default inventory is `/etc/ansible/hosts`

Using Ansible - ad-hoc commands

- Example

Use limit to further refine host pattern

```
ansible webservers -a "df -h” --limit *.example.com
    www.example.com | SUCCESS | rc=0 >>
    Filesystem            Size  Used Avail Use%
    Mounted on
    rootfs                1008G  9.5G  948G   1% /
    devtmpfs                15G  152K   15G   1% /dev
    tmpfs                   15G     0   15G   0%
    /dev/shm
    /dev/xvda1            1008G  9.5G  948G   1% /
```

Using Ansible - modules

- Building blocks for playbooks
- Idempotent
- Examples:

```
ansible example -m ping
    www.example.com | SUCCESS => {
        "changed": false,
        "ping": "pong"
    }

ansible example -m service -a "name=httpd state=started"
```

Using Ansible - playbooks

- Orchestrate steps
- Composed of one or more “plays”
- YML format
- Best kept in source control

Using Ansible - playbooks

- Playbook example: Ensure that both apache and postgres are installed up to the latest version available in the company's repository

Using Ansible - playbooks

- Sample inventory

```
$ vi /etc/ansible/hosts

[webservers]
web[01:10].example.com

[databases]
db[01:10].example.com
```

Using Ansible - playbooks

```
$ vi latest.yml
---
- hosts: webservers
  tasks:
    - name: ensure apache is at the latest version
      yum:
        name: httpd
        state: latest

- hosts: databases
  tasks:
    - name: ensure postgresql is at the latest version
      yum:
        name: postgresql
        state: latest
```

Indentation is very important!

Using Ansible - playbooks

- Line by line:

YML marker

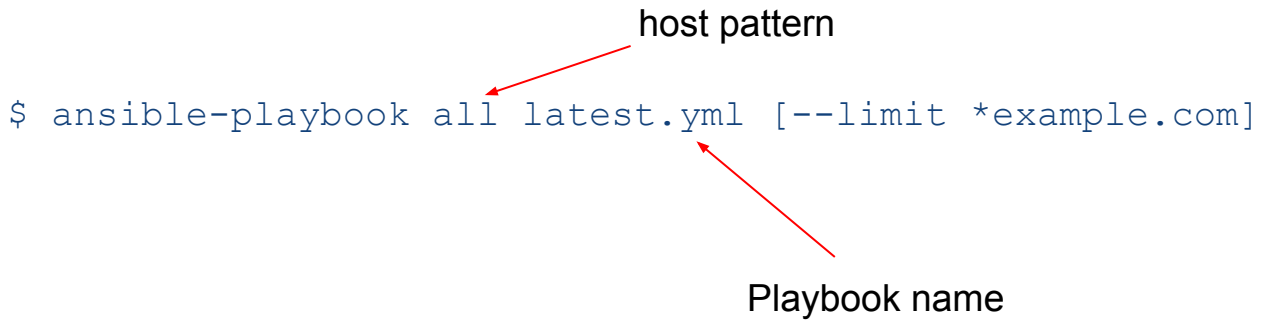
Host groups as per inventory file

modules

```
---
- hosts: webservers
  tasks:
  - name: ensure apache is at the latest version
    yum:
      name: httpd
      state: latest

- hosts: databases
  tasks:
  - name: ensure postgresql is at the latest version
    yum:
      name: postgresql
      state: latest
```

Using Ansible - playbooks

- Run playbooks with ansible-playbook command
- Similar syntax to ad-hoc mode



Using Ansible - playbooks

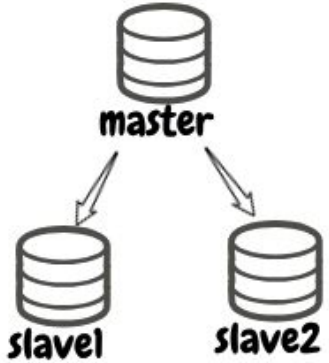
```
PLAY [all]
*********************************************************************
*

TASK [check if specified os user exists]
*********************************************************************
*changed: [mysql1]
ok: [mysql2]

PLAY RECAP
*********************************************************************
*mysql1 : ok=1    changed=1    unreachable=0    failed=0
mysql2 : ok=1    changed=0    unreachable=0    failed=0
```

Practical examples

- MySQL upgrade
- Streaming backup/restore

Upgrade MySQL

- Rolling approach
  - Upgrade slaves in sequence
  - Promote one slave as master
  - Upgrade old master



Upgrade MySQL

- Pre-tasks
  - Verify health
  - Figure out the master
  - Modify parameters
  - Downtime monitoring

Upgrade MySQL

- Verify health

```
- name: Ensure the cluster is healthy
  shell: "/opt/tools/check_cluster.sh"
  register: health_result
  failed_when: health_result.rc == 0
```

Override fail condition

Store the result for future use

Upgrade MySQL

- Figure out the master

```
- name: Capture Master Node
  shell: "/opt/tools/get_master.sh"
  changed_when: false
  run_once: true
  register: master_node_result
  failed_when: master_node_result.stdout == ""

- set_fact:
    master_node: "{{ master_node_result.stdout }}"
```

Control when a task is reported as changed

Only run task one time per batch of hosts

Upgrade MySQL

- Modify parameters

```
- name: get rid of removed parameters on my.cnf
  lineinfile:
    path: /etc/my.cnf
    regexp: '^({{ item }})'
    line: '#Removed for upgrade - \1'
    backrefs: yes
  with_items:
    - old_passwords
    - innodb_additional_mem_pool_size
```

Iterate replace {{ item }}
with each element

\1 in combination with backrefs:yes will
print the line as it was before

Upgrade MySQL

- Downtime monitoring

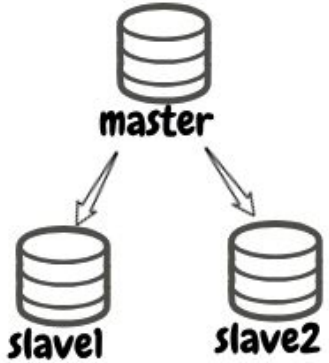
```
  - name: downtime alerts
      shell: "/usr/local/bin/downtime.sh --host='{{ ansible_hostname }}'
--service=\"{{ item }}\" --comment='Downtime services for Upgrade'"
      with_items: [ 'Slave Lag', 'MySQL status' ]
      delegate_to: nagios.example.com
```

Current host the playbook is running against

Run command on a different host

Upgrade MySQL

- Upgrade slaves
  - Take slave offline
  - Replace mysql binaries
  - Restart mysql
  - Run mysql_upgrade
  - Restart mysql (again)



Upgrade MySQL

- Take slave offline

Run a task as root

```
- name: Take slave offline
  become: yes
  shell: "/opt/tools/offline_server.sh {{ ansible_hostname }}"

- name: stop MySQL
  service:
    name: mysql
    state: stopped
```

Upgrade MySQL

- Replace binaries

```
- name: remove old packages
  shell: "rpm -qa | grep Percona-Server | xargs rpm -e --nodeps"

- name: remove old xtrabackup
  package:
    name: percona-xtrabackup
    state: absent
```

maps to yum/apt

Upgrade MySQL

- Replace binaries using repo

```
- name: Install MySQL 5.7
  package:
    name:
      - Percona-Server-server-57
      - Percona-Server-client-57
      - Percona-Server-shared-57
      - percona-xtrabackup-24
    state: present
```

Upgrade MySQL

- Replace binaries by manually copying

```
- name: copy MySQL 5.7 packages
  copy:
    src: /home/ivan/mysql57
    dest: /tmp

- name: Install MySQL 5.7
  yum:
    name:
      - /tmp/Percona-Server-client-57-5.7.24-27.1.el6.x86_64.rpm
      - /tmp/Percona-Server-server-57-5.7.24-27.1.el6.x86_64.rpm
      - /tmp/Percona-Server-shared-57-5.7.24-27.1.el6.x86_64.rpm
      - /tmp/percona-xtrabackup-24-2.4.9-1.el6.x86_64.rpm
    state: present
```

Upgrade MySQL

- Restart mysql

```
- name: start MySQL
  service:
    name: mysql
    state: started
```

Upgrade MySQL

- Run mysql_upgrade

```
- vars_prompt:
    - name: username
      prompt: "What is your username?"
      private: no

    - name: password
      prompt: "What is your password?"

- name: run mysql_upgrade
  shell: "mysql_upgrade -u{{ mysql_user }} -p{{ mysql_password }}"
```

- Option: pass variables on command line

```
ansible-playbook upgrade.yml --extra-vars "mysql_user=ivan mysql_pass=Passw0rd"
```

Upgrade MySQL

- Restart mysql (again)

```
- name: restart MySQL
  service:
    name: mysql
    state: restarted
```

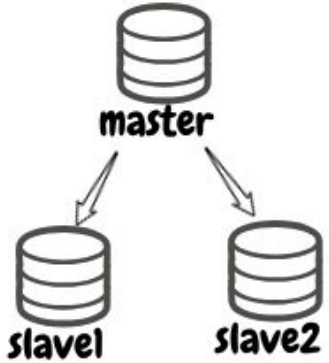
Upgrade MySQL

- Upgrade master
  - Most tasks are the same as before
  - Promote slave as master

```
- name: Switch master
  shell: "/opt/tools/switch --new-master=db2"
  run_once: true
  delegate_to: ...
```

Upgrade MySQL

- Post tasks
  - Check cluster health (again)
  - Remove downtime from monitoring



Upgrade MySQL

- Check cluster health (again)

```
- name: sleep for 60 s to give alerts time to recover
  wait_for: timeout=60

- name: Ensure the cluster is healthy
  shell: "/opt/tools/check_cluster.sh"
  register: health_result
  failed_when: health_result.rc == 0
```

Upgrade MySQL

- Remove downtime from monitoring

```
- name: cancel downtime alerts
  shell: "/usr/local/bin/downtime.sh --cancel --host='{{ ansible_hostname }}'
--service=\"{{ item }}\" --comment='Downtime services for Upgrade'"
  with_items: [ 'Slave Lag', 'MySQL status' ]
  run_once: true
  delegate_to: nagios.example.com
```

Backup/restore

- Xtrabackup piped over network

Backup/restore

- Determine source and target
- Downtime alerts
- Wipe target
- Stream backup from source to target
- Apply logs
- Start mysql
- Set up replication
- Remove downtime

Backup/restore

- Determine source and target

```
vars_prompt:
  - name: "source_server"
    prompt: "Specify the source server"
    private: no
  - name: "target_server"
    prompt: "Specify the target server (any existing data will be lost!)"
    private: no
  - name: "mysql_user"
    prompt: "Specify a DB admin User"
    private: yes
    default: 'rebuild_user'
  - name: "mysql_password"
    prompt: "Specify the DB admin password"
    private: yes
```

If passed via --extra-vars we don't get prompted

Hide sensitive info

Backup/restore

- Downtime alerts

```
- name: Schedule downtime in Nagios
  nagios:
    action: downtime
    minutes: 30
    service: mysql
    host: '{{ ansible_hostname }}'
    delegate_to: nagios.example.com
```

https://docs.ansible.com/ansible/latest/modules/nagios_module.html

Backup/restore

- Wipe target

```
- name: Capture MySQL Data Dir
  shell: "cat /etc/my.cnf | grep ^datadir | awk '{print $3}'"
  changed_when: False
  register: mysql_datadir_result

- name: Check if MySQL datadir was discovered correctly
  fail:
    msg: "MySQL datadir is blank.  Cannot continue!"
  when: mysql_datadir == '' and inventory_hostname == target_server
```

Backup/restore

- Wipe target

```
- name: Stop MySQL Service
   service:
     name: mysql
     state: stopped
   when: inventory_hostname == target_server

- name: Clear {{ mysql_datadir }} on {{ target_server }}
  shell: rm -rf {{ mysql_datadir }}/*
  when: inventory_hostname == target_server
```

Check if this is blank!

Backup/restore

- Stream backup from source to target

```
- name: Listen
  shell: nc -dl 3306 | xbstream -x -C {{ mysql_datadir }}
  become: yes
  async: 3600
  poll: 0
  register: async_list_results
  when: inventory_hostname == target_server

- name: Sleep to allow the Listen to start
  pause:
    seconds: 10
```

Restore is expected to take up to 1 h

Don't wait for completion
Before moving to next task

Backup/restore

- Stream backup from source to target

```
    - set_fact: innoback_cmd="innobackupex --stream=xbstream --user={{
mysql_user }} --password='{{ mysql_password }}' /tmp | nc {{ target_server }}
3306 > /tmp/xtrabackup_output.txt"
      when: inventory_hostname == source_server

    - name: Start Backup
      shell: >
        {{ innoback_cmd }}
      become: yes
      async: 3600
      poll: 0
      register: async_results
      when: inventory_hostname == source_server
```

Setting as fact avoids issues with strange characters in the password but exposes the password in verbose mode

Backup/restore

- Stream backup from source to target

```
- name: Check if Backup is Complete
  async_status:
    jid: "{{ async_results.ansible_job_id }}"
  register: job_result
  until: job_result.finished
  retries: 9000
  when: inventory_hostname == source_server
```

Keep checking result of previous task

Backup/restore

- Apply logs

```
- name: Start Apply Logs
  shell: innobackupex --use-memory=12G --apply-log {{ mysql_datadir }}
  async: "{{ async_wait_time }}"
  poll: "{{ poll_interval }}"
  when: inventory_hostname == target_server
```

Backup/restore

- Update perms & start mysql

```
- name: Update ownership on the files on {{ target_server }}
  file:
    path: "{{ mysql_datadir }}"
    state: directory
    recurse: yes
    owner: mysql
    group: mysql
  when: inventory_hostname == target_server

- name: Start MySQL service on {{ target_server }}
  service: mysql
    state: started
  when: inventory_hostname == target_server
```

Backup/restore

- Set up replication

```
- name: Read position from xtrabackup file
  shell: "cut -f1 /var/lib/mysql/xtrabackup_binlog_pos_innodb"
  register: repl_log_file

- name: Read position
  shell: "cut -f2 /var/lib/mysql/xtrabackup_binlog_pos_innodb"
  register: repl_log_pos
```

```
$ cat /var/lib/mysql/xtrabackup_binlog_pos_innodb
master-bin.000002    831
```

Backup/restore

- Set up replication

```
- name: Configure replication
  mysql_replication:
    mode: changemaster
    master_host: "{{ source_server }}"
    master_log_file: "{{ repl_log_file.stdout }}"
    master_log_pos: "{{ repl_log_pos.stdout }}"
    master_user: "{{ repl_user }}"
    master_password: "{{ repl_pass }}"
    login_user: "{{ mysql_user }}"
    login_password: "{{ mysql_password }}"

- name: start MySQL slave
  mysql_replication:
    mode: startslave
```

Values from previous step

Replication credentials

User with privs to make changes to replication

Backup/restore

- Remove downtime

```
- name: Remove downtime
  nagios:
    action: delete_downtime
    service: mysql
    host: '{{ ansible_hostname }}'
    delegate_to: nagios.example.com
```

Closing thoughts

- Keep playbooks under version control
- Give tasks a name
- Use built-in modules when possible (instead of shell)
- Ansible can save you lots of time
- Test with --check and --diff options
- Examples in https://github.com/igroene/ansible_playbooks