Ansible (really) loves PostgreSQL

PGConf.EU 2019

Cédric Villemain cedric@2ndQuadrant.com

16 October 2019

Cédric Villemain

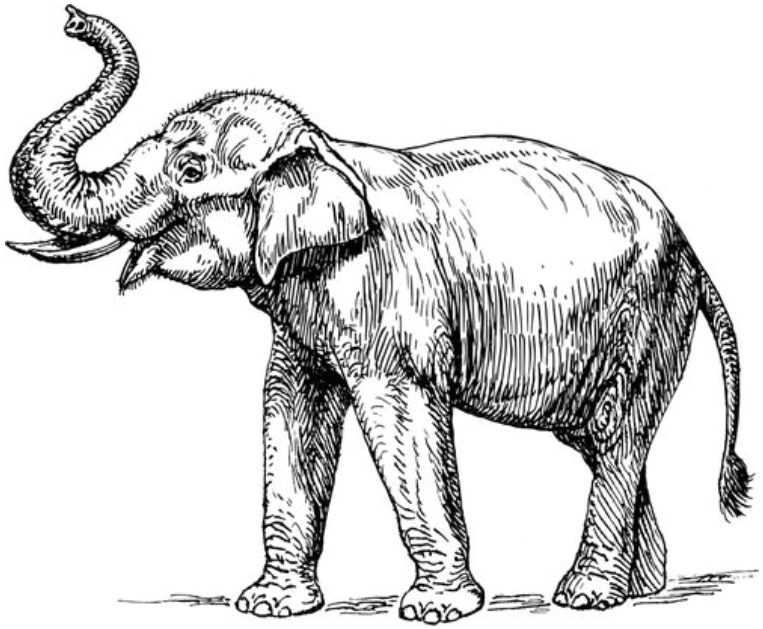
PostgreSQL Development & Expertise
Training
24x7 Support & Remote DBA

Major Sponsor

- **9.2** Refactoring checkpoint & group commit
- **9.3** Event Triggers
- **9.4** Replication Slot
- **9.5** Block Range INdex
- **9.6** Datawarehouse performance improvements
- **10** CREATE STATISTICS, Logical replication
- **11** Procedures, Partitionning
- **12** Generated Columns

Agenda

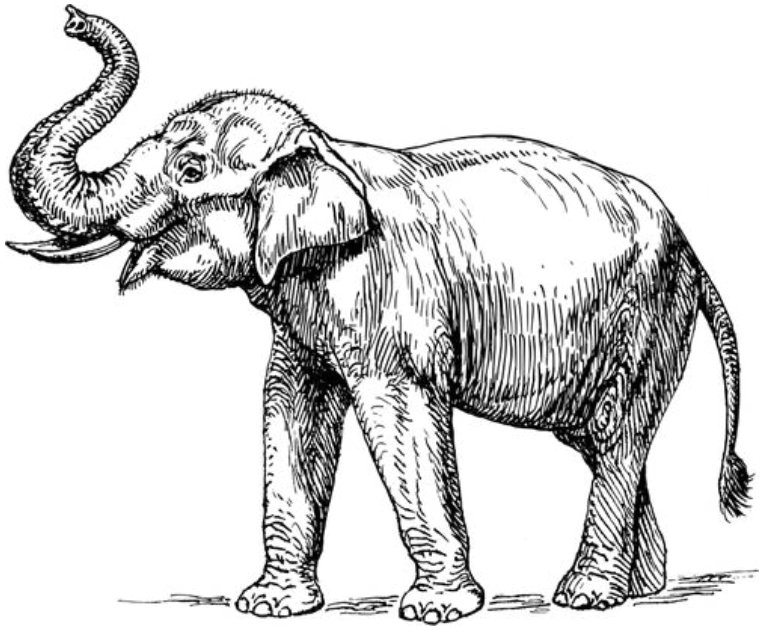
PostgreSQL



- deploy
- configure
- reconfigure
- update
- upgrade

- backup
- restore
- replicate
- monitor
- test

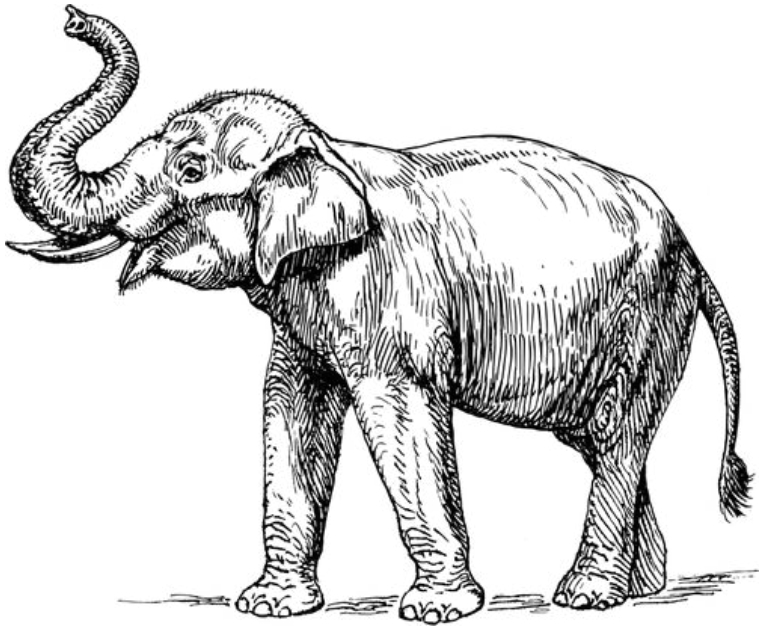
PostgreSQL



- deploy
- configure
- reconfigure
- update
- upgrade

- backup
- restore
- replicate
- monitor
- test

PostgreSQL



- deploy
- configure
- reconfigure
- update
- upgrade

- backup
- restore
- replicate
- monitor
- test

Problem ?

- handmade scripts
- packages
- deployment tools

Problem ?

- handmade scripts
- packages
- deployment tools

A Solution

2ndQuadrant
Professional PostgreSQL

Managing PostgreSQL with Ansible

Gülçin Yıldırım

PGConf EU, 2015, Vienna

1

https://slideshare.net/GulcinYildirim/managing-postgres-with-ansible

Automate

Demo: get pg version

Inventory

```
$ echo localhost > /etc/ansible/hosts
```

Command line

```
$ ansible all --module-name postgresql_info \
              --args 'login_user=cedric \
                      filter=version'
```

Output

```
localhost | SUCCESS => {
    [...]
    "version": {
        "major": 12,
        "minor": 0
    }
}
```

Demo: get pg version

Inventory

```
$ echo localhost > /etc/ansible/hosts
```

Command line

```
$ ansible all --module-name postgresql_info \
              --args 'login_user=cedric \
                      filter=version'
```

Output

```
localhost | SUCCESS => {
    [...]
    "version": {
        "major": 12,
        "minor": 0
    }
}
```

Demo: get pg version

Inventory

```
$ echo localhost > /etc/ansible/hosts
```

Command line

```
$ ansible all --module-name postgresql_info \
              --args 'login_user=cedric \
                      filter=version'
```

Output

```
localhost | SUCCESS => {
    [...]
    "version": {
        "major": 12,
        "minor": 0
    }
}
```

Demo: get pg version

Inventory

```
$ echo localhost > /etc/ansible/hosts
```

Command line

```
$ ansible all --module-name postgresql_info \
              --args 'login_user=cedric \
                      filter=version'
```

Output

```
localhost | SUCCESS => {
    [...]
    "version": {
        "major": 12,
        "minor": 0
    }
}
```

Demo: get pg version

Inventory

```
$ echo localhost > /etc/ansible/hosts
```

Command line

```
$ ansible all --module-name postgresql_info \
              --args 'login_user=cedric \
                      filter=version'
```

Output

```
localhost | SUCCESS => {
    [...]
    "version": {
        "major": 12,
        "minor": 0
    }
}
```

Wait!

What about «Playbooks» ?

Command line

```
$ ansible-playbook my_playbook.yml
```

Better than

```
#!/bin/bash
# get info about PostgreSQL
ansible all  ---module-name postgresql_info \\
             ---args 'login_user=cedric \\
                      filter=version'


# do something else
ansible all  ---module-name .....
```

Wait!

What about «Playbooks» ?

Command line

```
$ ansible-playbook my_playbook.yml
```

Better than

```
#!/bin/bash
# get info about PostgreSQL
ansible all  ---module-name postgresql_info \\
             ---args 'login_user=cedric \\
                      filter=version'

# do something else
ansible all  ---module-name .....
```

Wait!

What about «Playbooks» ?

Command line

```
$ ansible-playbook my_playbook.yml
```

Better than

```
#!/bin/bash
# get info about PostgreSQL
ansible all  ---module-name postgresql_info \\
               ---args 'login_user=cedric \\
                       filter=version'

# do something else
ansible all  ---module-name .....
```

Demo: get pg version

The Playbook: pg_version.yml

```
---
- hosts: all
  tasks:
    - postgresql_info:
        login_user: cedric
        filter: version
```

Demo: get pg version

The Playbook: pg_version.yml

```
---
- hosts: all
  tasks:
    - postgresql_info:
        login_user: cedric
        filter: version
```

Demo: get pg version

The Playbook: pg_version.yml

```
---
- hosts: all
  tasks:
    - postgresql_info:
        login_user: cedric
        filter: version
```

Demo: get pg version

Command line

```
$ ansible-playbook pg_version.yml
```

Output

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

TASK [Gathering Facts] ****************************
ok: [localhost]

TASK [postgresql_info] ****************************
ok: [localhost]

PLAY RECAP ************************************
localhost : ok=2 changed=0 unreachable=0 failed=0 [...]
```

Demo: get pg version

Command line

```
$ ansible-playbook pg_version.yml
```

Output

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

TASK [Gathering Facts] ***************************
ok: [localhost]

TASK [postgresql_info] ***************************
ok: [localhost]

PLAY RECAP ***********************************
localhost : ok=2 changed=0 unreachable=0 failed=0 [...]
```

Serialized tasks execution

- wait for task completion for each host
- execute tasks on all hosts, in parallel

Demo: really get pg version !

The Playbook: pg_version.yml

```
---
- hosts: all
  tasks:
    - postgresql_info:
        login_user: cedric
        filter: version
      register: pg_version
    - debug: var=pg_version.version.major
```

Demo: really get pg version !

Output

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

TASK [Gathering Facts] ****************************
ok: [localhost]

TASK [postgresql_info] ****************************
ok: [localhost]

TASK [debug] **************************************
ok: [localhost] => {
    "pg_version.version.major": "12"
}

PLAY RECAP ************************************
localhost : ok=3 changed=0 unreachable=0 failed=0 [...]
```

Demo: Define fact

The Playbook: pg_version.yml

```
---
- hosts: all
  tasks:
    - postgresql_info:
        login_user: cedric
        filter: version
      register: pg_version
    - set_fact:
        pg_major: "{{pg_version.version.major}}"
    - debug: var=pg_major
```

Demo: Define fact

The Playbook: pg_version.yml

```
---
- hosts: all
  tasks:
    - postgresql_info:
        login_user: cedric
        filter: version
      register: pg_version
    - set_fact:
        pg_major: "{{pg_version.version.major}}"
    - debug: var=pg_major
```

Demo: Define fact

Output

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

[...]

TASK [set_fact] ******************************
ok: [localhost]

TASK [debug] *********************************
ok: [localhost] => {
    "pg_major": "12"
}

PLAY RECAP ***********************************
localhost : ok=4 changed=0 unreachable=0 failed=0 [...]
```

Demo: Use fact

The Playbook: pg_version.yml

```
[...]
- name: Ensure we do not recycle WAL file (COW FS)
  postgresql_set:
    login_user: cedric
    name: wal_recycle
    value: False|string
  when:
    pg_major|int > 11
```

Demo: Use fact

The Playbook: pg_version.yml

```
- name: Reload PostgreSQL if 12+
  postgresql_query:
    login_user: cedric
    db: postgres
    query: SELECT pg_reload_conf()
  when:
    pg_major|int > 11
```

Demo: Use fact

Output

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

[...]

TASK [Ensure we do not recycle WAL file (COW FS)] **

changed: [localhost]

TASK [Reload PostgreSQL if 12+] ******************
ok: [localhost]

PLAY RECAP ***************************************
localhost : ok=5 changed=1 unreachable=0 failed=0 [...]
```

Wonderful...

- If not in a CoW FS, we don't want to apply
- If parameter has not changed, we don't want to reload

Wonderful...

- If not in a CoW FS, we don't want to apply
- If parameter has not changed, we don't want to reload

Variables

Define Variable

```
- hosts: all
  vars:
    wal_recycle: True
```

Handler & Notify

Define Handler & Call Notify

```
handlers:
  - name: reload PostgreSQL
    postgresql_query:
      login_user: cedric
      db: postgres
      query: SELECT pg_reload_conf()
tasks:
  - name: Ensure we do not recycle WAL file (COW FS)
    postgresql_set:
      login_user: cedric
      name: wal_recycle
      value: "{{wal_recycle|string}}"
    when:
      pg_major|int > 11
    notify: reload PostgreSQL
```

Handler & Notify

Define Handler & Call Notify

```
handlers:
  - name: reload PostgreSQL
    postgresql_query:
      login_user: cedric
      db: postgres
      query: SELECT pg_reload_conf()
tasks:
  - name: Ensure we do not recycle WAL file (COW FS)
    postgresql_set:
      login_user: cedric
      name: wal_recycle
      value: "{{wal_recycle|string}}"
    when:
      pg_major|int > 11
    notify: reload PostgreSQL
```

About Roles

```yaml
---
- hosts: all
  vars:
    wal_recycle: True
  handlers:
    - name: reload PostgreSQL
      postgresql_query:
        login_user: cedric
        db: postgres
        query: SELECT pg_reload_conf()
  tasks:
    - postgresql_info:
        login_user: cedric
        filter: version
      register: pg_version
    - set_fact:
        pg_major: "{{pg_version.version.major}}"
    - name: Ensure we do not recycle WAL file (COW FS)
      postgresql_set:
        login_user: cedric
        name: wal_recycle
        value: "{{wal_recycle|string}}"
      when:
        pg_major|int > 11
      notify: reload PostgreSQL
```

- Variables
- Tasks
- Handlers
- Templates
- ...

Simpler playbook

pg_version.yml

```
---
- hosts: all
  vars:
    wal_recycle: False
  roles:
    - pg_cow
```

Role pg_cow (vars)

roles/pg_cow/vars/main.yml

```
---
vars:
  wal_recycle: True
```

Role pg_cow (tasks)

roles/pg_cow/tasks/main.yml

```
---
- postgresql_info:
    login_user: cedric
    filter: version
  register: pg_version
- set_fact:
    pg_major: "{{pg_version.version.major}}"
- name: Ensure we do not recycle WAL file (COW FS)
  postgresql_set:
    login_user: cedric
    name: wal_recycle
    value: "{{wal_recycle|string}}"
  when:
    pg_major|int > 11
  notify: reload PostgreSQL
```

Role pg_cow (one handler)

roles/pg_cow/handlers/main.yml

```
---
- name: reload PostgreSQL
  postgresql_query:
    login_user: cedric
    db: postgres
    query: SELECT pg_reload_conf()
```

Role pg_cow (another handler)

roles/pg_cow/handlers/main.yml

```
---
- name: reload PostgreSQL
  service:
    name: postgresql
    state: reloaded
```

Wait !

What is a Module ?

- External Library
- Compliant with Ansible
- Called and configured inside *tasks*.

Demo: start several PostgreSQL dockers

The Playbook: pg_docker.yml

```
---
- hosts: all
  vars:
    n: 6
  tasks:
    - name: Create & Start container on port '943[1-n]'
      docker_container:
        name: "db_test_{{ item|string }}"
        image: "postgres:latest"
        published_ports: "943{{ item|int }}:5432"
        networks:
          - name: bridge
        volumes:
          - /tmp/docker:/tmp/docker
      loop: "{{ range(1, (n|int+1))|list }}"
```

Demo: start several PostgreSQL docker

Command Line and Variable

```
ansible-playbook --extra-vars n='2' pg_docker.yml
```

Demo: «Naive» Rolling Upgrade

The Playbook: pg_rolling.yml

```
---
- hosts: all
  serial: 1
  tasks:
    - postgresql_query:
        login_user: cedric
        db: postgres
        query: SELECT pg_is_in_recovery() p(pg_is_in)
      register: x
    - set_fact:
        is_standby: "{{x.query_result[0].pg_is_in|bool}}"
    - package:
        name: postgresql-12
        state: latest
      when: is_standby == true
[...]
```

Demo: «Naive» Rolling Upgrade

The Playbook: pg_rolling.yml

```
[...]
- hosts: all
  tasks:
    - postgresql_query:
        login_user: cedric
        db: postgres
        query: SELECT pg_is_in_recovery() p(pg_is_in)
      register: x
    - set_fact:
        is_standby: "{{x.query_result[0].pg_is_in|bool}}"
    - package:
        name: postgresql-12
        state: latest
      when: is_standby == false
```

Ansible 2.6 modules

Cluster

- postgresql_user
- postgresql_db

Database

- postgresql_lang
- postgresql_ext
- postgresql_schema
- postgresql_privs

https://docs.ansible.com/ansible/2.6/modules/list_of_database_modules.html?highlight=postgresql

Ansible 2.8 modules

Cluster

- postgresql_ping
- postgresql_info
- postgresql_pg_hba
- postgresql_set
- postgresql_tablespace
- postgresql_user
- postgresql_membership
- postgresql_db
- postgresql_slot

Database

- postgresql_lang
- postgresql_ext
- postgresql_schema
- postgresql_table
- postgresql_idx
- postgresql_owner
- postgresql_privs
- postgresql_query

https://docs.ansible.com/ansible/latest/modules/list_of_database_modules.html?highlight=postgresql

Ansible model

- Core application
- Modules
- Plugins

Ansible Galaxy

A place for sharing:

- Modules
- Plugins
- Playbooks
- Roles

Ansible Galaxy

A place for sharing:

- Modules
- Plugins
- Playbooks
- Roles

Search in Ansible Galaxy

- PostgreSQL: 1 collection (debops) & 555 roles !
- PostGIS: 21 roles
- Barman: 10 roles
- pgBouncer: 10 roles
- BDR: 4 roles
- pglogical: 2 roles
- pgBackRest: 1 role
- pgpool: 1 role

Your own way

You can also share anywhere, at least playbooks and roles.

Questions ?

now or later!