PGConf.Russia 2023

PostgresPro

Набор ролей ansible для постгреса.

Автоматизируем разворачивание СУБД в зависимости от задач.

PostgresPro

Роли.

- отдельные роли для обвязки:
  DCS (Distributed Configuration Store): `etcd`, `consul`;
  `balancer&pooler:` `haproxy`, `pgbouncer`.
- одна роль для постгреса - `postgres` в различных конфигурациях: одиночный хост; физическая реплика; кластера: `patroni`, `stolon`. Причина одной роли — конфигурирование посгреса (конфигурационные параметры СУБД).
  Какая конфигурация будет разворачиваться, определяется переменными.

PostgresPro

Разворачиваемые набором конфигурации.

- единичный хост postgres;
- физическая реплика (логическая - ?);
- кластер: patroni+etcd;
- кластер: patroni+consul;
- кластер: stolon+etcd;
- кластер: stolon+consul.

PostgresPro

Причины создания ролей.

Q1. Почему не «решение»?

- потому что решение начинается с постановки задачи;
  (А задачи решаются совсем не одним-единственным способом!)
- потому что решить сферическую задачу по разворачиванию отказоустойчивого кластера в вакууме, как следствие из предыдущего пункта, весьма трудо- и времязатратно;
- **потому что кластер СУБД не является самостоятельной сущностью, а является частью какого-либо проекта;**

PostgresPro

Причины создания ролей.

Q2. Почему не использовать существующие роли?

- совершенно недопустимое (diff?):

```
shell: cat < EOF
  <config_body>
  EOF > <path>/postgresql.conf
```

- везде какая-то часть ПО устанавливается как попало (обновление? как? pip — конфликты гарантированы).
- под конкретный пакетный менеджер, например, под yum (dnf? а что это такое?);

PostgresPro

Общие замечания по ansible.

- роль в ansible — аналог разделяемой библиотеки в программировании, один и тот же код используется многократно;
- роль, в общем случае, устанавливает нужное ПО, при необходимости инициализирует (initdb — это оно самое!) и конфигурирует;
- роль применяется к единичному хосту и/или группе хостов (см. далее идеальный плейбук);
- новая роль не должна ломать существующее решение (насколько это возможно).

PostgresPro

Общие замечания по ansible.

Идеальный плейбук:

```
- hosts: all
  roles:
    - role: common
      tags: system
- hosts: dev # DCS group of hosts
  roles:
    - role: consul
      tags: dcs
- hosts: dev # Postgres hosts/group of hosts
  roles:
    - role: postgres
      tags: postgres
# any similar to definitions
```

место для переменных: group_vars,host_vars, обязаны быть определены в административном порядке.

PostgresPro

Общие замечания по ansible.

Типичная структура хранения конфигураций в VCS, например, в git:

```
корень репозитория
|-community.roles
|-community.collections
|-company.roles
|-prj01
…
|-prjNN
```

PostgresPro

Общие замечания по ansible.

Типичная структура хранения конфигураций в VCS, например, в git (вариант):

```
/path_to/community.roles
/path_to/community.collections
/path_to/company.roles
корень репозитория
|-prj01
…
|-prjNN
```

PostgresPro

Общие замечания по ansible.

Типичная структура хранения конфигураций в VCS, например, в git (продолжение, структура проекта):

```
prj
|-filter_plugins (?)
|-group_vars
|-host_vars
|-library (?)
|-module_utils (?)
|-microservice01
...
|-microserviceNN
|-roles
|-prj.yml
...
|-other-playbook.yml
```

PostgresPro

Общие замечания по ansible.

Типичная структура хранения конфигураций в VCS, например, в git (продолжение, структура микросервиса):

```
microservice
|-filter_plugins (?)
|-group_vars
|-host_vars
|-library (?)
|-module_utils (?)
|-submicroservice01
...
|-submicroserviceNN
|-roles
|-microservice.yml
...
|-another-playbook.yml
```

PostgresPro

Общие замечания по ansible.

Типичная структура хранения конфигураций в VCS, например, в git (продолжение).

В каждый проект/микросервис устанавливать «решение»?!

НЕТ!

PostgresPro

Общие замечания по ansible.

Место для ролей, которыми разворачивается системное ПО:

```
корень репозитория
|-community.roles
|-community.collections
|-company.roles
|-prj01
…
|-prjNN
```

PostgresPro

Использование. Общие замечания по ansible.

Сборка переменных:

- `postgres_params` в режиме single:

```
postgres_params: "{{ [postgres_params_default, postgres_params_group_all
| default([]), postgres_params_group | default([]), postgres_params_host
| default([])] |
community.general.lists_mergeby('name',list_merge='append_rp') }}"
```

- допустимые суффиксы:

`_default` — используется в <role_name>/defaults/main.yml
`_group_all`
`_group`
`_host`

PostgresPro

Использование. Общие переменные.

Переменные, определяемые для любой роли:

```
<role_name>_<anythings>_dir: "{{ role_path }}/<anythings>"
```

где <anythings> - одно из files,templates,vars

Пример:

```
postgres_templates_dir: "{{ role_path }}/templates"
```

Привет nginx и haproxy!

PostgresPro

Использование. Общие переменные.

Переменные, определяемые для любой роли:

```
common_domain_name: "mshurutov.home"
common_short_hostname: "srv01dev"
common_full_hostname: "{{ common_short_hostname }}.{{ common_domain_name }}"
common_ip4_default: "192.168.1.51"
```

Magic variables:

- `inventory_hostname` ?
- `ansible_host` ?

Да, но нет.

PostgresPro

Использование. DCS: etcd.

Параметры, необходимые для etcd;

- `etcd_config_file: "/etc/default/etcd"`
- `etcd_data_dir: "/var/lib/etcd/default"`
- `etcd_cluster_group: "dev"`

Параметры, которые желательно переопределить для etcd:

- `etcd_initial_cluster_token: "etcd_cluster"`
  (кластер может быть далеко не один)

PostgresPro

Использование. DCS: consul.

Параметры, обязательные для consul-a:

- `consul_hosts_group: "dev"`
- `consul_master_host: "srv01dev.mshurutov.home"` ?
-
```
consul_params_group:
  - name: "bootstrap_expect"
    value: 3
```

Параметры, не совсем обязательные:

- `consul_data_dir: "/var/lib/consul"` ?

PostgresPro

Использование. Переменные для роли postgres.

Параметры, которые **необходимо** определить для любой конфигурации;

- `postgres_major_version: 15` ? (default - 14)
- `postgres_home_dir: "/var/lib/postgresql"`
- `postgres_conf_dir: "/etc/postgresql/{{ postgres_major_version }}"`
- `postgres_data_dir: "{{ postgres_home_dir }}/{{ postgres_major_version }}/data"`
- `postgres_bin_dir: "/usr/lib64/postgresql-{{ postgres_major_version }}/bin"`

PostgresPro

Использование. Переменные для роли postgres.

Параметры, которые желательно определить для любой конфигурации;

- `postgres_kernel_params` ? определен только swapiness
  `- { name: vm_swappiness, value: '5' }`
- `postgres_password_encoding: "scram-sha-256"` ?
- `postgres_repl_user: 'repluser'` ?
- `postgres_repl_password: 'replpwd'` (вот так, без шифрования?)
- `postgres_rewind_user: 'rewinduser'` ?
- `postgres_rewind_password: 'rewindpwd'` ? (см. выше)
- `postgres_own_password: "postgresownpwd"` ? (см. выше)

PostgresPro

Использование. Переменные для роли postgres.

Параметры для одиночного экземпляра, см. предыдущие слайды, ибо по-умолчанию:

```
postgres_install_mode: "single"
```

PostgresPro

Использование. Переменные для роли postgres.

Параметры реплики, определяются для конкретного хоста:

- `postgres_install_mode: "replica"`
- `postgres_master_host: "srv02dev.mshurutov.home"`
- `postgres_backup_connstring: "-h {{ postgres_master_host }} -U {{ postgres_repl_user }}"`
- `postgres_replica_init: "from_master"`
-
```
postgres_recovery_params:
  - { name: "primary_conninfo", value: "'host={{ postgres_master_host }}
user={{ postgres_repl_user }}'" }
  - { name: "hot_standby", value: "on" }
```

PostgresPro

Использование. Общие кластерные переменные.

```
postgres_install_mode: "<cluster_type>"
```

где <cluster_type> может принимать значения patroni или stolon.

```
postgres_master_host: "srv01dev.mshurutov.home"
```

PostgresPro

Использование. Кластер: patroni.

Параметры, обязательные для patroni:

- `patroni_config_file: '{{ patroni_config_dir }}/config.yml'`
  `?(unit file)`
- `patroni_scope_name: 'clusterdev'` (несколько кластеров используют одну DCS)
- `etcd_cluster_group: "dev"`
- ```
  patroni_log_params:
    - name: "level"
      value: "DEBUG" ?
    - name: "dir"
      value: "/var/log/postgresql"
  ```

Разница между

`bootstrap.postgresql.parameters` и `postgresql.parameters`.
  ```

PostgresPro

Использование. Кластер: stolon.

Устанавливать ПО в обход пакетного менеджера? **НИ В КОЕМ СЛУЧАЕ!!!**

Есть ли пакеты для stolon? Нет.
Есть ли спеки для пакетирования? Нет.

PostgresPro

Использование. Кластер: stolon.

Устанавливать ПО в обход пакетного менеджера? **НИ В КОЕМ СЛУЧАЕ!!!**

Есть ли пакеты для stolon? Нет.
Есть ли спеки для пакетирования? На самом деле есть:
`https://sourceforge.net/p/stolon-packages-specs/`

разворачивание собственного репозитория выходит за рамки доклада.

PostgresPro

Использование. Кластер: stolon.

Параметры, которые изменяются:

- `stolon_apt_repo: "deb http://srv01/repos/deb/ bullseye main"`
- `postgres_dcs_groups: "dev"`
- `stolon_store_backend: "consul"` (по-умолчанию - etcdv3)
- `stolon_dcs_client_port: 8500` (для consul-a)
-
```
stolon_daemons_common_options_group:
  - key: "store_backend"
    value: "{{ stolon_store_backend }}"
    comment: |-
      # store backend type (etcdv2/etcd, etcdv3, consul or kubernetes)
```

PostgresPro

Использование. Кластер: stolon.

Параметры, которые вшиты в ПО:

Каталог, куда инициализируется экземпляр СУБД:

```
${STKEEPER_DATA_DIR}/postgres - /var/lib/stolon/postgres
```

Параметры pg_hba.conf:

```
local postgres postgres md5
local replication repluser md5
host all postgres 0.0.0.0/0 md5
host all postgres ::0/0 md5
host replication repluser 0.0.0.0/0 md5
host replication repluser ::0/0 md5
```

PostgresPro

Использование. Параметры постгреса.

(На базе собственного опыта)

Параметры постгреса. Общие:

```
postgres_password_encoding: "scram-sha-256"
postgres_client_port: 5432
postgres_params_default:
  - { name: listen_addresses, value: "'*'" }
  - { name: port, value: "{{ postgres_client_port }}" }
  - { name: max_connections, value: "100" }
  - { name: password_encryption, value: "{{ postgres_password_encoding }}"}
  - { name: lc_messages, value: "POSIX" }
...
```

Продолжение следует...

PostgresPro

Использование. Параметры постгреса.

(На базе собственного опыта)

Параметры постгреса. Память:

```
postgres_params_default:
...
  - { name: shared_buffers , value: "{{ (ansible_memtotal_mb/4096+1) | int }}GB" }
  - { name: work_mem , value: "16MB" }
  - { name: maintenance_work_mem, value: "1GB" }
...
```

Продолжение следует...

PostgresPro

Использование. Параметры постгреса.

(На базе собственного опыта)

Параметры постгреса. Настройки WAL-ов:

```
postgres_params_default:
...
  - { name: wal_level, value: "replica" }
  - { name: archive_mode, value: "on" }
  - { name: archive_command, value: "'/bin/true'" }
  - { name: max_wal_senders, value: "10" }
  - { name: max_replication_slots, value: "10" }
  - { name: hot_standby, value: "on" }
  - { name: max_logical_replication_workers, value: "5" }

...
```

Продолжение следует...

PostgresPro

Использование. Параметры постгреса.

(На базе собственного опыта)

Параметры постгреса. Журналирование:

```
postgres_params_default:
...
  - { name: log_destination, value: "'stderr'" }
  - { name: logging_collector, value: "on" }
  - { name: log_directory, value: 'log' }
  - { name: log_file_mode, value: "0640" }
  - { name: log_min_duration_statement, value: "0" }
  - { name: log_duration, value: "on" }
  - { name: log_line_prefix, value: "'%t [%p]: user=%u,db=%d,app=%a,client=%h '" }
  - { name: log_statement, value: "all" }
...
```

Продолжение следует...

PostgresPro

Использование. Параметры постгреса.

(На базе собственного опыта)

Параметры постгреса. Автовакуум, библиотеки и автозапуск после сбоя:

```
postgres_params_default:
...
  - { name: autovacuum_vacuum_scale_factor, value: "0.05" }
  - { name: autovacuum_vacuum_insert_scale_factor, value: "0.01" }
  - { name: autovacuum_analyze_scale_factor, value: "0.01" }
  - { name: shared_preload_libraries, value: "'pg_stat_statements'" }
  - { name: restart_after_crash, value: "false" }
```

Продолжение (pg_hba.conf) следует...

PostgresPro

Использование. Параметры постгреса.

(На базе собственного опыта)

Параметры постгреса. Файл pg_hba.conf:

```
postgres_pg_hba_lines:
  - local all         postgres                peer
  - host  all         all      127.0.0.1/32 {{ postgres_password_encoding }}
  - host  all         all      ::1/128      {{ postgres_password_encoding }}
  - host  all         postgres 0.0.0.0/0    reject
  - host  replication repluser samenet      {{ postgres_password_encoding }}
  - host  all         all      samenet      {{ postgres_password_encoding }}
```

PostgresPro

Использование. Haproxy.

Параметры по умолчанию — для patroni без доступа к реплике на одном хосте с ПГ.

Определяем порты, и группу хостов с постгресом:

```
haproxy_pg_port: 5432
postgres_client_port: 6432
postgres_hosts_group: "dev"
```

И добавляем доступ к асинхронной реплике при необходимости:

```
haproxy_pg_clusters_group:
  - connect_type: 'ro'
    bind_port: '{{ haproxy_pg_port + 1 }}'
    check_option: 'httpchk GET /async'
    check_pg_port: '{{ haproxy_check_pg_port }}'
    pg_max_conn: '{{ haproxy_pg_max_conn }}'
```

PostgresPro

Использование. Pgbouncer.

Параметры по умолчанию:

```
pgbouncer_config_dir: "/etc/pgbouncer"
pgbouncer_config_file: "{{ pgbouncer_config_dir }}/pgbouncer.ini"
pgbouncer_user_file: "{{ pgbouncer_config_dir }}/userlist.txt"
pgbouncer_params_default:
  - name: "pool_mode"
    value: "transaction"
  - name: "auth_file"
    value: "{{ pgbouncer_user_file }}"
```

PostgresPro

Использование. Pgbouncer.

Объявление переменных, пароли:

```
pgbouncer_vault_dbuser_pwd: !vault |
          $ANSIBLE_VAULT;1.1;AES256
          643431373862663064363131636537633933303863383762373630326530313134393265666533653964343131656365373237376133376137303932366330330a633562333035363361363866643831393231376165373662373565646361613262306139313765636662643261376332623366666430356630363363623836630a623866376466346534373964373537323338656638656130303264383262386
pgbouncer_vault_pgbenchuser_pwd: !vault |
          $ANSIBLE_VAULT;1.1;AES256
          643262363833313237666330363132346639663135353038656361643931343532383735663565373738383962373439383934393665343339643436333139320a363863663532383430323832656330356234636230393836313333363839616635396131623661663637653132616264396463326137303566343163363839610a366262396662396364613338313733383933373330666235353036356165646
```

PostgresPro

Использование. Pgbouncer.

Объявление переменных:

```
pgbouncer_databases:
  - name: pgbench
    connstring: "host=srv01dev,srv02dev,srv03dev port=6432 user=pgbench
password={{ pgbouncer_vault_dbuser_pwd }}"
pgbouncer_params_group:
  - name: "pidfile"
    value: "/run/postgresql/pgbouncer.pid"
  - name: "listen_addr"
    value: "*"
  - name: "listen_port"
    value: 7432
  - name: "auth_type"
    value: "{{ postgres_password_encoding }}"
pgbouncer_users:
  - name: "pgbench"
    value: "{{ pgbouncer_vault_pgbenchuser_pwd }}"
```

PostgresPro

Ссылки на репозитории ролей.

Находятся поиском по «<name> ansible» на https://sourceforge.net/

- consul:
  https://sourceforge.net/projects/consul-role/
- etcd:
  https://sourceforge.net/projects/etcd-ansible-role/
- haproxy-pg
  https://sourceforge.net/p/haproxy-pg/
- pgboncer:
  https://sourceforge.net/p/pgbouncer/
- postgres:
  https://sourceforge.net/p/postgres-ansible-role/
- stolon packages specs:
  https://sourceforge.net/p/stolon-packages-specs/
- Мой профиль: https://sourceforge.net/u/shurutov/profile/

Спасибо за внимание!