Ansible: Backup einer remote PostgreSQL-Datenbank erstellen

Kathegorie
Docker, PostGreSQL, SQL Server

Denise
Robinson

In diesem Artikel wollen wir darauf eingehen, wie man mithilfe eines Ansible Playbooks die Erstellung und den Download eines Datenbankdumps von einem Remote-Server automatisieren kann. Dieses Playbook führt folgende Schritte aus:

1. Richtet ein Sicherungsverzeichnis auf dem lokalen PC unter **/tmp/bkups/blog/postgres/db** ein.

2. Führt einen Shell-Befehl in einem Postgres-Docker-Container auf einem Remote-Server aus, der die Datenbank auf ein eingebundenes Volumen dumpt.

3. Im letzten Schritt wird die gedumpte Datenbank von dem Remote-Server in das Sicherungsverzeichnis auf dem lokalen PC heruntergeladen

Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz

info@madafa.de
madafa.de

+49 6131 3331612

Implementierung

Erster Schritt: So erstellst Du ein lokales Sicherungsverzeichnis

Mit der folgenden Ansible-Playbook-Eingabe erstellst Du ein lokales Sicherungsverzeichnis:

```
- name: Fetch a backup of a postgres DB that runs on a remote docker container
  hosts: all
  gather_facts: yes

  tasks:
    - name: Set backup_dir and db_dump_name variable
      set_fact:
        backup_dir: /tmp/bkups/blog/postgres/db
        db_dump_name: my_db-{{ ansible_date_time.date }}.bkp

    - name: Make sure backup directory exists on the local machine
      delegate_to: localhost
      become: no
      file:
        state: directory
        path: "{{ backup_dir }}"
```

Beachte, dass wir die Ansible-Variable **ansible_date_time** verwenden, die als Ergebnis von **gather_facts:yes** festgelegt wird. Diese Informationen werden im Dateinamen des Datenbankdumps verwendet.

Wenn das Sicherungsverzeichnis auf dem Remote-Server denselben Pfad wie auf Deinem lokalen PC hat, genügt es, **set_fact** als eine einzelne Variable festzulegen.

Schritt 2: So führst Du einen Befehl im Docker-Container auf dem Remote-Server aus

Der einfachste Weg, ein Backup der Datenbank im Postgres-Docker-Container zu erstellen, besteht darin, einen Befehl **pg_dump** als Postgres-Benutzer im Container auszuführen. Dies kann unter Verwendung des folgenden Ansible Befehls erreicht werden:

```
- name: Dump the DB to a bind-mounted volume
      command: |
        docker exec -i --user postgres postgres bash -c
        "pg_dump -Fc my_db > bkups/db/{{ db_dump_name }}"
```

Dadurch wird der Befehl **pg_dump** in der Datenbank mit dem Namen **my_db** (bzw. dem Namen, den Du für Deine Datenbank gewählt hast) ausgeführt und unter **bkups/db** ausgegeben.
Somit wird ein Datenbankdump erzeugt und im eingebundenen Volumen des Postgres-Docker-Containers abgelegt. Folgender Auszug stammt aus meiner docker-compose.yml-Datei:

```
services:
  db:
    container_name: postgres
    image: postgres:latest
    restart: always
    environment:
      POSTGRES_DB: my_db
    expose:
      - "5432"
    volumes:
       - /tmp/bkups/blog/postgres:/bkups
```

Nachdem die Datenbank in **bkups/db/{{db_dump_name}}** im Container gespeichert wurde, ist sie auf meinem Remote-Server unter **/tmp/bkups/blog/postgres/db** verfügbar. Bitte beachte, dass dies genau der Name des ausgewählten Sicherungsverzeichnisses ist.

Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

Letzter Schritt: Herunterladen der gedumpten Datenbank vom Remote-Server auf den lokalen PC

Über Ansibles sogenanntes Fetch-Modul kannst Du die gedumpte Datenbank auf Deinen lokalen Pc herunterladen:

```
- name: Fetch the dumped DB from remote to local
    fetch:
      src: "{{ backup_dir }}/{{ db_dump_name }}"
      dest: "{{ backup_dir }}/"
      flat: yes
```

Zusammenfassung: Playbook, Ausführung und Ergebnis

Alles in allem sieht das endgültige Playbook also so aus:

```
---

- name: Fetch a backup of a postgres DB that runs on a remote docker container
  hosts: all
  gather_facts: yes

  tasks:
    - name: Set backup_dir and db_dump_name variable
      set_fact:
        backup_dir: /tmp/bkups/blog/postgres/db
        db_dump_name: my_db-{{ ansible_date_time.date }}.bkp

    - name: Make sure backup directory exists on the local machine
      delegate_to: localhost
      become: no
      file:
        state: directory
        path: "{{ backup_dir }}"

    - name: Dump the DB to a bind-mounted volume
      command: |
        docker exec -i --user postgres postgres bash -c
        "pg_dump -Fc my_db > bkups/db/{{ db_dump_name }}"

    - name: Fetch the dumped DB from remote to local
      fetch:
        src: "{{ backup_dir }}/{{ db_dump_name }}"
        dest: "{{ backup_dir }}/"
        flat: yes
```

und kann zum Beispiel so ausgeführt werden:

```
ansible-playbook playbooks/backup.yml -l [my_hostname] --key-file=~/.ssh/id_rsa -i
$PWD/inventory/
```

Dabei entspricht **[my_hostname]** dem Namen eines Host, der zuvor in dem Ansible Inventory angelegt wurde.

Die Ausgabe des Befehls sollte ungefähr so aussehen:

```
PLAY [Fetch a backup of a postgres DB that runs on a remote docker container] ***
********************************************************************

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

TASK [Set backup_dir and db_dump_name variable] *******************************
********************************************************************
ok: [my_hostname]

TASK [Make sure backup directory exists on the local machine] *****************
********************************************************************
changed: [my_hostname -> localhost]
```


Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

```
TASK [Dump the DB to a bind-mounted volume] ************************************
*******************************************************************
changed: [my_hostname]

TASK [Fetch the dumped DB from remote to local] ********************************
*******************************************************************
changed: [my_hostname]

PLAY RECAP *********************************************************************
*******************************************************************
my_hostname                : ok=5    changed=3    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

Nun solltest Du die Datei auf Deinem lokalen PC haben.

Du solltest jetzt dazu in der Lage sein mithilfe von Ansible ein Backup einer remote PostgreSQL-Datenbank zu erstellen.

ansible, backup, container, datenbank,
docker, postgre, sql server

Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612