Verwalten von DigitalOcean Reserved IPs

Es ist eine häufige Herausforderung, dass die IP-Adressen von Servern leicht von der Great Firewall (GFW) blockiert werden können. Dies gilt insbesondere für Cloud-Server. Um dies zu mildern, besteht eine Strategie darin, die reservierten IPs von DigitalOcean zu verwenden und sie Ihrem Droplet neu zuzuweisen, wenn die aktuelle IP blockiert ist. Dieser Beitrag stellt ein Python-Skript vor, das diesen Prozess automatisiert. Das Skript ist auch Open Source und auf GitHub verfügbar.

Das Skript ermöglicht Ihnen:

- Zu überprüfen, ob eine reservierte IP einem bestimmten Droplet zugewiesen ist.
- Eine neue reservierte IP einem Droplet zuzuweisen, wenn die aktuelle blockiert ist.
- Zu überprüfen, ob Port 80 auf der reservierten IP geöffnet ist (eine einfache Möglichkeit, um zu überprüfen, ob die IP funktioniert).

Hier ist das Python-Skript:

```
import socket
import os
import argparse
import json
import requests
import time

# Funktion, um die DigitalOcean API-Header zu erhalten
def get_digitalocean_headers():
    api_key = os.environ.get("DO_API_KEY")
    if not api_key:
        print("Fehler: DO_API_KEY nicht in den Umgebungsvariablen gefunden.")
        return None
    return {
        "Authorization": f"Bearer {api_key}",
        "Content-Type": "application/json"
    }

# Funktion, um alle reservierten IPs von DigitalOcean abzurufen
def fetch_reserved_ips():
```

```
    headers = get_digitalocean_headers()
    if not headers:
        return None
    try:
        url = "https://api.digitalocean.com/v2/reserved_ips"
        resp = requests.get(url, headers=headers)
        resp.raise_for_status()
        reserved_ips_data = resp.json().get("reserved_ips", [])
        with open('response.json', 'w') as f:
            json.dump(reserved_ips_data, f, indent=4) # Speichert die Antwort zur Fehlerbehebung in ein
        return reserved_ips_data
    except requests.exceptions.RequestException as e:
        print(f"Fehler beim Abrufen der reservierten IP-Adresse: {e}")
        return None

# Funktion, um eine reservierte IP von einem Droplet zu entfernen
def unassign_ip_from_droplet(ip_address, droplet_id, droplet_name):
    headers = get_digitalocean_headers()
    if not headers:
        return False

    try:
        url = f"https://api.digitalocean.com/v2/reserved_ips/{ip_address}"
        resp = requests.delete(url, headers=headers)
        resp.raise_for_status()
        print(f"IP {ip_address} erfolgreich vom Droplet {droplet_name} entfernt")
        return True
    except requests.exceptions.RequestException as e:
        print(f"Fehler beim Entfernen der IP {ip_address} vom Droplet {droplet_name}: {e}")
        return False

# Funktion, um eine reservierte IP einem Droplet zuzuweisen
def assign_ip_to_droplet(ip_address, droplet_id, droplet_name):
    headers = get_digitalocean_headers()
    if not headers:
        return False
```

```
    try:
        url = f"https://api.digitalocean.com/v2/reserved_ips/{ip_address}/actions"
        req = {
            "type": "assign",
            "droplet_id": droplet_id
        }
        resp = requests.post(url, headers=headers, json=req)
        resp.raise_for_status()
        print(f"IP {ip_address} erfolgreich dem Droplet {droplet_name} zugewiesen")
        return True
    except requests.exceptions.RequestException as e:
        print(f"Fehler beim Zuweisen der IP {ip_address} zum Droplet {droplet_name}: {e}")
        return False

# Funktion, um reservierte IPs zu verarbeiten, Zuweisungen zu überprüfen und bei Bedarf neu zuzuweisen
def process_reserved_ips(reserved_ips, droplet_name, only_check=False):
    if not reserved_ips:
        print("Keine reservierten IPs in Ihrem Konto gefunden.")
        return None

    for reserved_ip in reserved_ips:
        ip_address = reserved_ip.get("ip")
        if not ip_address:
            print("Keine IP-Adresse für eine reservierte IP gefunden.")
            continue

        droplet = reserved_ip.get("droplet", None)
        if droplet_name:
            if droplet and droplet.get("name") == droplet_name:
                print(f"Die reservierte IP {ip_address} ist dem Droplet zugewiesen: {droplet_name}")
                if only_check:
                    if check_port_80(ip_address):
                        print(f"Port 80 ist auf {ip_address} für das Droplet {droplet_name} geöffnet")
                    else:
                        print(f"Port 80 ist auf {ip_address} für das Droplet {droplet_name} geschlossen
```

```
                    return ip_address
                droplet_id = droplet.get("id")
                if droplet_id:
                    if unassign_ip_from_droplet(ip_address, droplet_id, droplet_name):
                        # Versucht, nach dem Entfernen eine neue IP zuzuweisen

                        new_ip = create_new_reserved_ip(droplet_id)
                        if new_ip:
                            print("Warte 10 Sekunden, bevor die neue IP zugewiesen wird...")
                            time.sleep(10)
                            if assign_ip_to_droplet(new_ip, droplet_id, droplet_name):
                                print(f"Neue IP {new_ip} erfolgreich dem Droplet {droplet_name} zugewies
                            else:
                                print(f"Fehler beim Zuweisen der neuen IP {new_ip} zum Droplet {droplet_
                        else:
                            print("Keine verfügbare IP zum Zuweisen")

                else:
                    print(f"Konnte die IP {ip_address} nicht entfernen, da die Droplet-ID nicht gefunde
                return None
            elif droplet:
                print(f"Die reservierte IP {ip_address} ist nicht dem Droplet zugewiesen: {droplet_name
            else:
                print(f"Keine Droplets sind der reservierten IP zugewiesen: {ip_address}")
        else:
            return ip_address
    return None

# Funktion, um eine neue reservierte IP zu erstellen
def create_new_reserved_ip(droplet_id):
    headers = get_digitalocean_headers()
    if not headers:
        print("Fehler beim Abrufen der DigitalOcean-Header.")
        return False
    try:
        url = "https://api.digitalocean.com/v2/reserved_ips"
```

```
            req = {
                "region": "sgp1", # Sie können die Region bei Bedarf ändern
            }
            print(f"Versuche, eine neue reservierte IP für die Droplet-ID zu erstellen: {droplet_id}")
            resp = requests.post(url, headers=headers, json=req)
            resp.raise_for_status()
            new_ip = resp.json().get("reserved_ip", {}).get("ip")
            print(f"Neue reservierte IP erfolgreich erstellt: {new_ip}")
            return new_ip
        except requests.exceptions.RequestException as e:
            print(f"Fehler beim Erstellen einer neuen reservierten IP: {e}")
            return False

# Funktion, um zu überprüfen, ob Port 80 auf einer IP-Adresse geöffnet ist
def check_port_80(ip_address):
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.settimeout(5)
            s.connect((ip_address, 80))
            return True
    except Exception:
        return False

# Hauptfunktion, um die reservierte IP zu erhalten
def get_reserved_ip(droplet_name=None, only_check=False):
    reserved_ips = fetch_reserved_ips()
    if reserved_ips is None:
        return None
    return process_reserved_ips(reserved_ips, droplet_name, only_check)

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description="Reservierte IP-Adresse von DigitalOcean abrufen.")
    parser.add_argument("--droplet-name", required=True, help="Name des Droplets, um zu überprüfen, ob
    parser.add_argument("--only-check", action="store_true", help="Nur überprüfen, ob die IP dem Droplet
    args = parser.parse_args()
```

```
reserved_ip = get_reserved_ip(args.droplet_name, args.only_check)
if reserved_ip:
    print(f"Die reservierte IP-Adresse lautet: {reserved_ip}")
```

Erklärung:

1. **Bibliotheken importieren:** Importiert die notwendigen Bibliotheken für Netzwerkoperationen, Umgebungsvariablen, Argumentenparsing, JSON-Handling, HTTP-Anfragen und Zeitverzögerungen.
2. `get_digitalocean_headers()`**:** Ruft den DigitalOcean API-Schlüssel aus den Umgebungsvariablen ab und konstruiert die notwendigen Header für API-Anfragen.
3. `fetch_reserved_ips()`**:** Ruft alle reservierten IPs ab, die mit Ihrem DigitalOcean-Konto verknüpft sind, über die API. Es speichert auch die rohe Antwort in `response.json` zur Fehlerbehebung.
4. `unassign_ip_from_droplet()`**:** Entfernt eine gegebene reservierte IP von einem bestimmten Droplet.
5. `assign_ip_to_droplet()`**:** Weist eine gegebene reservierte IP einem bestimmten Droplet zu.
6. `process_reserved_ips()`**:** Dies ist die Kernlogik:
   - Es durchläuft alle reservierten IPs.
   - Wenn ein `droplet_name` angegeben ist, überprüft es, ob die IP diesem Droplet zugewiesen ist.
   - Wenn `only_check` wahr ist, überprüft es, ob Port 80 geöffnet ist, und gibt die IP zurück.
   - Wenn nicht `only_check`, entfernt es die aktuelle IP, erstellt eine neue und weist die neue IP dem Droplet zu.
7. `create_new_reserved_ip()`**:** Erstellt eine neue reservierte IP in der Region `sgp1` (Sie können dies ändern).
8. `check_port_80()`**:** Überprüft, ob Port 80 auf einer gegebenen IP-Adresse geöffnet ist. Dies ist eine einfache Möglichkeit, um zu überprüfen, ob die IP erreichbar ist.
9. `get_reserved_ip()`**:** Koordiniert den Prozess des Abrufens und Verarbeitens von reservierten IPs.
10. `if __name__ == '__main__':`**:** Parst die Befehlszeilenargumente (`--droplet-name` und `--only-check`) und ruft `get_reserved_ip` auf, um das Skript auszuführen.

Verwendung:

1. **DigitalOcean API-Schlüssel einrichten:** Setzen Sie die Umgebungsvariable `DO_API_KEY` mit Ihrem DigitalOcean API-Schlüssel.
2. **Skript ausführen:**
    - Um zu überprüfen, ob eine IP einem Droplet zugewiesen ist und ob Port 80 geöffnet ist: `bash python your_script_name.py --droplet-name your_droplet_name --only-check`
    - Um eine neue IP einem Droplet zuzuweisen: `bash python your_script_name.py --droplet-name your_droplet_name`

Dieses Skript bietet ein grundlegendes Framework für die Verwaltung von reservierten IPs. Sie können es basierend auf Ihren spezifischen Anforderungen weiter ausbauen.