管理 DigitalOcean 保留 IP

伺服器的 IP 地址容易被防火長城（GFW）封鎖，這是一個常見的挑戰，尤其是對於雲端伺服器而言。為了解決這個問題，一個策略是使用 DigitalOcean 的保留 IP，並在當前 IP 被封鎖時將其重新分配給你的 droplet。這篇文章介紹了一個 Python 腳本來自動化這個過程。該腳本也已開源，可以在GitHub上找到。

這個腳本允許你：

- 檢查保留 IP 是否分配給特定的 droplet。
- 如果當前的 IP 被封鎖，則將新的保留 IP 重新分配給 droplet。
- 檢查保留 IP 的 80 端口是否開放（這是一個簡單的方法來檢查 IP 是否有效）。

以下是 Python 腳本：

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

# 獲取 DigitalOcean API 頭部的函數
def get_digitalocean_headers():
    api_key = os.environ.get("DO_API_KEY")
    if not api_key:
        print(" 錯誤：環境變量中未找到 DO_API_KEY。")
        return None
    return {
        "Authorization": f"Bearer {api_key}",
        "Content-Type": "application/json"
    }

# 從 DigitalOcean 獲取所有保留 IP 的函數
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) # 將響應保存到文件中以便調試
        return reserved_ips_data
    except requests.exceptions.RequestException as e:
        print(f" 獲取保留 IP 地址時出錯：{e}")
        return None

# 從 droplet 取消分配保留 IP 的函數
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" 成功從 droplet {droplet_name}中刪除 IP {ip_address}")
        return True
    except requests.exceptions.RequestException as e:
        print(f" 從 droplet {droplet_name}中刪除 IP {ip_address}時出錯：{e}")
        return False

# 將保留 IP 分配給 droplet 的函數
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" 成功將 IP {ip_address}分配給 droplet {droplet_name}")
        return True
    except requests.exceptions.RequestException as e:
        print(f" 將 IP {ip_address}分配給 droplet {droplet_name}時出錯：{e}")
        return False

# 處理保留 IP、檢查分配並在需要時重新分配的函數
def process_reserved_ips(reserved_ips, droplet_name, only_check=False):
    if not reserved_ips:
        print(" 您的帳戶中未找到保留 IP。")
        return None

    for reserved_ip in reserved_ips:
        ip_address = reserved_ip.get("ip")
        if not ip_address:
            print(" 未找到保留 IP 的 IP 地址。")
            continue

        droplet = reserved_ip.get("droplet", None)
        if droplet_name:
            if droplet and droplet.get("name") == droplet_name:
                print(f" 保留 IP {ip_address}已分配給 droplet：{droplet_name}")
                if only_check:
                    if check_port_80(ip_address):
                        print(f"droplet {droplet_name}的 IP {ip_address}的 80 端口已開放")
                    else:
                        print(f"droplet {droplet_name}的 IP {ip_address}的 80 端口已關閉")
                    return ip_address
                droplet_id = droplet.get("id")
```

```
                if droplet_id:
                    if unassign_ip_from_droplet(ip_address, droplet_id, droplet_name):
                        # 取消分配後嘗試分配新的 IP

                        new_ip = create_new_reserved_ip(droplet_id)
                        if new_ip:
                            print(" 分配新 IP 前等待 10 秒...")
                            time.sleep(10)
                            if assign_ip_to_droplet(new_ip, droplet_id, droplet_name):
                                print(f" 成功將新 IP {new_ip}分配給 droplet {droplet_name}")
                            else:
                                print(f" 未能將新 IP {new_ip}重新分配給 droplet {droplet_name}")
                        else:
                            print(" 沒有可分配的 IP")

                else:
                    print(f" 無法取消分配 IP {ip_address}，因為未找到 droplet ID。")
                return None
            elif droplet:
                print(f" 保留 IP {ip_address}未分配給 droplet：{droplet_name}")
            else:
                print(f" 沒有 droplet 分配給保留 IP：{ip_address}")
        else:
            return ip_address
    return None

# 創建新的保留 IP 的函數
def create_new_reserved_ip(droplet_id):
    headers = get_digitalocean_headers()
    if not headers:
        print(" 獲取 DigitalOcean 頭部失敗。")
        return False
    try:
        url = "https://api.digitalocean.com/v2/reserved_ips"
        req = {
            "region": "sgp1", # 如果需要，可以更改區域
```

```
        }
        print(f" 嘗試為 droplet ID {droplet_id}創建新的保留 IP")
        resp = requests.post(url, headers=headers, json=req)
        resp.raise_for_status()
        new_ip = resp.json().get("reserved_ip", {}).get("ip")
        print(f" 成功創建新的保留 IP：{new_ip}")
        return new_ip
    except requests.exceptions.RequestException as e:
        print(f" 創建新的保留 IP 時出錯：{e}")
        return False

# 檢查 IP 地址的 80 端口是否開放的函數
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

# 獲取保留 IP 的主函數
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=" 獲取 DigitalOcean 保留 IP 地址。")
    parser.add_argument("--droplet-name", required=True, help=" 要檢查保留 IP 是否分配給的 droplet 名稱
    parser.add_argument("--only-check", action="store_true", help=" 僅檢查 IP 是否分配給 droplet，不重
    args = parser.parse_args()

    reserved_ip = get_reserved_ip(args.droplet_name, args.only_check)
    if reserved_ip:
```

```
    print(f" 保留 IP 地址為：{reserved_ip}")
```

解釋：

1. **導入庫：**導入必要的庫，用於網絡操作、環境變量、參數解析、JSON 處理、HTTP 請求和時間延遲。
2. `get_digitalocean_headers()`：從環境變量中獲取 DigitalOcean API 密鑰，並構建 API 請求所需的頭部。
3. `fetch_reserved_ips()`：使用 API 獲取與您的 DigitalOcean 帳戶關聯的所有保留 IP。它還將原始響應保存到 `response.json` 以便調試。
4. `unassign_ip_from_droplet()`：從指定的 droplet 中取消分配給定的保留 IP。
5. `assign_ip_to_droplet()`：將給定的保留 IP 分配給指定的 droplet。
6. `process_reserved_ips()`：這是核心邏輯：
    - 它遍歷所有保留 IP。
    - 如果提供了 `droplet_name`，則檢查 IP 是否分配給該 droplet。
    - 如果 `only_check` 為真，則檢查 80 端口是否開放並返回 IP。
    - 如果不是 `only_check`，則取消分配當前 IP，創建一個新的 IP，並將新 IP 分配給 droplet。
7. `create_new_reserved_ip()`：在 `sgp1` 區域創建一個新的保留 IP（您可以更改此區域）。
8. `check_port_80()`：檢查給定 IP 地址的 80 端口是否開放。這是驗證 IP 是否可訪問的簡單方法。
9. `get_reserved_ip()`：協調獲取和處理保留 IP 的過程。
10. `if __name__ == '__main__':`：解析命令行參數（`--droplet-name` 和`--only-check`）並調用 `get_reserved_ip` 來執行腳本。

使用方法：

1. **設置 DigitalOcean API 密鑰：**使用您的 DigitalOcean API 密鑰設置 `DO_API_KEY` 環境變量。
2. **運行腳本：**
    - 要檢查 IP 是否分配給 droplet 以及 80 端口是否開放：`bash python your_script_name.py --droplet-name your_droplet_name --only-check`
    - 要將新的 IP 重新分配給 droplet：`bash python your_script_name.py --droplet-name your_droplet_name`

這個腳本提供了一個管理保留 IP 的基本框架。您可以根據具體需求進一步擴展它。