# DigitalOcean の予約済み IP の管理

サーバーの IP アドレスが Great Firewall（GFW）によって簡単にブロックされることは、特にクラウドサーバーにおいてよくある課題です。これを緩和するための一つの戦略として、DigitalOcean の予約済み IP を使用し、現在の IP がブロックされたときにそれをドロップレットに再割り当てする方法があります。この記事では、このプロセスを自動化する Python スクリプトを紹介します。このスクリプトはGitHubでオープンソースとして公開されています。

このスクリプトを使用すると、以下のことが可能です：

- 予約済み IP が特定のドロップレットに割り当てられているかどうかを確認する。
- 現在の IP がブロックされた場合に、新しい予約済み IP をドロップレットに再割り当てする。
- 予約済み IP のポート 80 が開いているかどうかを確認する（IP が機能しているかどうかを簡単に確認する方法）。

以下がその Python スクリプトです：

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

# DigitalOcean API ヘッダーを取得する関数
def get_digitalocean_headers():
    api_key = os.environ.get("DO_API_KEY")
    if not api_key:
        print("Error: DO_API_KEY not found in environment variables.")
        return None
    return {
        "Authorization": f"Bearer {api_key}",
        "Content-Type": "application/json"
    }

# DigitalOcean から予約済み IP を取得する関数
```

```python
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) # デバッグ用にレスポンスをファイルに保存
        return reserved_ips_data
    except requests.exceptions.RequestException as e:
        print(f"Error getting reserved IP address: {e}")
        return None

# ドロップレットから予約済み IP を割り当て解除する関数
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"Successfully deleted IP {ip_address} from droplet {droplet_name}")
        return True
    except requests.exceptions.RequestException as e:
        print(f"Error deleting IP {ip_address} from droplet {droplet_name}: {e}")
        return False

# ドロップレットに予約済み IP を割り当てる関数
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"Successfully assigned IP {ip_address} to droplet {droplet_name}")
            return True
        except requests.exceptions.RequestException as e:
            print(f"Error assigning IP {ip_address} to droplet {droplet_name}: {e}")
            return False

    # 予約済み IP を処理し、割り当てを確認し、必要に応じて再割り当てする関数
    def process_reserved_ips(reserved_ips, droplet_name, only_check=False):
        if not reserved_ips:
            print("No reserved IPs found in your account.")
            return None

        for reserved_ip in reserved_ips:
            ip_address = reserved_ip.get("ip")
            if not ip_address:
                print("No IP address found for a reserved IP.")
                continue

            droplet = reserved_ip.get("droplet", None)
            if droplet_name:
                if droplet and droplet.get("name") == droplet_name:
                    print(f"The reserved IP {ip_address} is assigned to droplet: {droplet_name}")
                    if only_check:
                        if check_port_80(ip_address):
                            print(f"Port 80 is open on {ip_address} for droplet {droplet_name}")
                        else:
```

```
                    print(f"Port 80 is closed on {ip_address} for droplet {droplet_name}")
                return ip_address
            droplet_id = droplet.get("id")
            if droplet_id:
                if unassign_ip_from_droplet(ip_address, droplet_id, droplet_name):
                    # 割り当て解除後に新しい IP を割り当てる

                    new_ip = create_new_reserved_ip(droplet_id)
                    if new_ip:
                        print("Sleeping for 10 seconds before assigning new IP...")
                        time.sleep(10)
                        if assign_ip_to_droplet(new_ip, droplet_id, droplet_name):
                            print(f"Successfully assigned new IP {new_ip} to droplet {droplet_name}"
                        else:
                            print(f"Failed to reassign new IP {new_ip} to droplet {droplet_name}")
                    else:
                        print("No available IP to assign")

            else:
                print(f"Could not unassign IP {ip_address} because droplet ID was not found.")
            return None
        elif droplet:
            print(f"The reserved IP {ip_address} is not assigned to droplet: {droplet_name}")
        else:
            print(f"No droplets are assigned to the reserved IP: {ip_address}")
    else:
        return ip_address
return None

# 新しい予約済み IP を作成する関数
def create_new_reserved_ip(droplet_id):
    headers = get_digitalocean_headers()
    if not headers:
        print("Failed to get DigitalOcean headers.")
        return False
    try:
```

```
        url = "https://api.digitalocean.com/v2/reserved_ips"
        req = {
            "region": "sgp1", # 必要に応じてリージョンを変更可能
        }
        print(f"Attempting to create a new reserved IP for droplet ID: {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"Successfully created new reserved IP: {new_ip}")
        return new_ip
    except requests.exceptions.RequestException as e:
        print(f"Error creating new reserved IP: {e}")
        return False

# IP アドレスのポート 80 が開いているか確認する関数
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

# 予約済み IP を取得するメイン関数
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="Get DigitalOcean reserved IP address.")
    parser.add_argument("--droplet-name", required=True, help="Name of the droplet to check if the rese
    parser.add_argument("--only-check", action="store_true", help="Only check if the IP is assigned to
    args = parser.parse_args()
```

```
    reserved_ip = get_reserved_ip(args.droplet_name, args.only_check)
    if reserved_ip:
        print(f"The reserved IP address is: {reserved_ip}")
```

**説明:**

1. **ライブラリのインポート:** ネットワーク操作、環境変数、引数解析、JSON 処理、HTTP リクエスト、時間遅延に必要なライブラリをインポートします。
2. `get_digitalocean_headers()`: 環境変数から DigitalOcean API キーを取得し、API リクエストに必要なヘッダーを構築します。
3. `fetch_reserved_ips()`: API を使用して DigitalOcean アカウントに関連付けられたすべての予約済み IP を取得します。また、デバッグ用に生のレスポンスを `response.json` に保存します。
4. `unassign_ip_from_droplet()`: 指定された予約済み IP を指定されたドロップレットから割り当て解除します。
5. `assign_ip_to_droplet()`: 指定された予約済み IP を指定されたドロップレットに割り当てます。
6. `process_reserved_ips()`: これがコアロジックです：
   - すべての予約済み IP を反復処理します。
   - `droplet_name` が提供されている場合、その IP がそのドロップレットに割り当てられているかどうかを確認します。
   - `only_check` が true の場合、ポート 80 が開いているかどうかを確認し、IP を返します。
   - `only_check` でない場合、現在の IP を割り当て解除し、新しい IP を作成し、その新しい IP をドロップレットに割り当てます。
7. `create_new_reserved_ip()`: `sgp1` リージョンに新しい予約済み IP を作成します（必要に応じて変更可能）。
8. `check_port_80()`: 指定された IP アドレスのポート 80 が開いているかどうかを確認します。これは IP が到達可能かどうかを簡単に確認する方法です。
9. `get_reserved_ip()`: 予約済み IP の取得と処理のプロセスを調整します。
10. `if __name__ == '__main__':`: コマンドライン引数（`--droplet-name` と `--only-check`）を解析し、スクリプトを実行するために `get_reserved_ip` を呼び出します。

**使用方法:**

1. **DigitalOcean API キーの設定:** 環境変数 `DO_API_KEY` に DigitalOcean API キーを設定します。

2. **スクリプトの実行:**
    - IP がドロップレットに割り当てられているかどうか、およびポート 80 が開いているかどうかを確認する場合：bash　　python your_script_name.py --droplet-name your_droplet_name --only-check
    - 新しい IP をドロップレットに再割り当てする場合：bash　　python your_script_name.py --droplet-name your_droplet_name

このスクリプトは、予約済み IP を管理するための基本的なフレームワークを提供します。特定のニーズに基づいてさらに拡張することができます。