管理 DigitalOcean 保留 IP

服务器 IP 地址容易被防火墙（GFW）屏蔽是一个常见的挑战，尤其是云服务器。为了缓解这一问题，一种策略是使用 DigitalOcean 的保留 IP，并在当前 IP 被屏蔽时将其重新分配给您的 Droplet。本文将介绍一个自动化此过程的 Python 脚本。该脚本已在GitHub上开源。

该脚本允许您：

- 检查保留 IP 是否分配给特定 Droplet。
- 如果当前 IP 被屏蔽，将新的保留 IP 重新分配给 Droplet。
- 检查保留 IP 的 80 端口是否开放（一种简单的检查 IP 是否可用的方法）。

以下是 Python 脚本：

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

# 获取 DigitalOcean API 头信息的函数
def get_digitalocean_headers():
    api_key = os.environ.get("DO_API_KEY")
    if not api_key:
        print(" 错误：环境变量中未找到 DO_API_KEY。")
        return None
    return {
        "Authorization": f"Bearer {api_key}",
        "Content-Type": "application/json"
    }

# 从 DigitalOcean 获取所有保留 IP 的函数
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) # 将响应保存到文件以便调试
        return reserved_ips_data
    except requests.exceptions.RequestException as e:
        print(f" 获取保留 IP 地址时出错：{e}")
        return None

# 从 Droplet 取消分配保留 IP 的函数
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" 成功从 Droplet {droplet_name} 删除 IP {ip_address}")
        return True
    except requests.exceptions.RequestException as e:
        print(f" 从 Droplet {droplet_name} 删除 IP {ip_address} 时出错：{e}")
        return False

# 将保留 IP 分配给 Droplet 的函数
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" 成功将 IP {ip_address} 分配给 Droplet {droplet_name}")
        return True
    except requests.exceptions.RequestException as e:
        print(f" 将 IP {ip_address} 分配给 Droplet {droplet_name} 时出错：{e}")
        return False

# 处理保留 IP、检查分配并在需要时重新分配的函数
def process_reserved_ips(reserved_ips, droplet_name, only_check=False):
    if not reserved_ips:
        print(" 您的账户中没有找到保留 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} 已分配给 Droplet：{droplet_name}")
                if only_check:
                    if check_port_80(ip_address):
                        print(f"Droplet {droplet_name} 的 IP {ip_address} 的 80 端口已开放")
                    else:
                        print(f"Droplet {droplet_name} 的 IP {ip_address} 的 80 端口已关闭")
                    return ip_address
                droplet_id = droplet.get("id")
                if droplet_id:
                    if unassign_ip_from_droplet(ip_address, droplet_id, droplet_name):
                        # 取消分配后尝试分配新 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" 成功将新 IP {new_ip} 分配给 Droplet {droplet_name}")
                            else:
                                print(f" 未能将新 IP {new_ip} 重新分配给 Droplet {droplet_name}")
                        else:
                            print(" 没有可分配的 IP")

                else:
                    print(f" 无法取消分配 IP {ip_address}，因为未找到 Droplet ID。")
                return None
            elif droplet:
                print(f" 保留 IP {ip_address} 未分配给 Droplet：{droplet_name}")
            else:
                print(f" 没有 Droplet 分配给保留 IP：{ip_address}")
        else:
            return ip_address
    return None

# 创建新保留 IP 的函数
def create_new_reserved_ip(droplet_id):
    headers = get_digitalocean_headers()
    if not headers:
        print(" 获取 DigitalOcean 头信息失败。")
        return False
    try:
        url = "https://api.digitalocean.com/v2/reserved_ips"
        req = {
            "region": "sgp1", # 如果需要可以更改区域
        }
        print(f" 尝试为 Droplet ID {droplet_id} 创建新的保留 IP")
        resp = requests.post(url, headers=headers, json=req)
```

```
        resp.raise_for_status()
        new_ip = resp.json().get("reserved_ip", {}).get("ip")
        print(f" 成功创建新的保留 IP：{new_ip}")
        return new_ip
    except requests.exceptions.RequestException as e:
        print(f" 创建新保留 IP 时出错：{e}")
        return False

# 检查 IP 地址的 80 端口是否开放的函数
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

# 获取保留 IP 的主函数
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=" 获取 DigitalOcean 保留 IP 地址。")
    parser.add_argument("--droplet-name", required=True, help=" 要检查保留 IP 是否分配给的 Droplet 名称
    parser.add_argument("--only-check", action="store_true", help=" 仅检查 IP 是否分配给 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 地址为：{reserved_ip}")
```

解释：

1. **导入库：**导入网络操作、环境变量、参数解析、JSON 处理、HTTP 请求和时间延迟所需的库。
2. `get_digitalocean_headers()`：从环境变量中获取 DigitalOcean API 密钥，并构建 API 请求所需的头信息。
3. `fetch_reserved_ips()`：使用 API 获取与您的 DigitalOcean 账户关联的所有保留 IP。它还将原始响应保存到 `response.json` 以便调试。
4. `unassign_ip_from_droplet()`：从指定的 Droplet 取消分配给定的保留 IP。
5. `assign_ip_to_droplet()`：将给定的保留 IP 分配给指定的 Droplet。
6. `process_reserved_ips()`：这是核心逻辑：
   - 它遍历所有保留 IP。
   - 如果提供了 `droplet_name`，它会检查 IP 是否分配给该 Droplet。
   - 如果 `only_check` 为真，它会检查 80 端口是否开放并返回 IP。
   - 如果不是 `only_check`，它会取消分配当前 IP，创建一个新 IP，并将新 IP 分配给 Droplet。
7. `create_new_reserved_ip()`：在 `sgp1` 区域创建一个新的保留 IP（您可以更改此区域）。
8. `check_port_80()`：检查给定 IP 地址的 80 端口是否开放。这是验证 IP 是否可用的简单方法。
9. `get_reserved_ip()`：协调获取和处理保留 IP 的过程。
10. `if __name__ == '__main__':`：解析命令行参数（`--droplet-name` 和`--only-check`）并调用 `get_reserved_ip` 以执行脚本。

使用方法：

1. **设置 DigitalOcean API 密钥：**使用您的 DigitalOcean API 密钥设置 `DO_API_KEY` 环境变量。
2. **运行脚本：**
   - 要检查 IP 是否分配给 Droplet 以及 80 端口是否开放：`bash python 脚本名称.py --droplet-name 您的 Droplet 名称 --only-check`
   - 要将新 IP 重新分配给 Droplet：`bash python 脚本名称.py --droplet-name 您的 Droplet 名称`

该脚本提供了一个管理保留 IP 的基本框架。您可以根据具体需求进一步扩展它。