Managing DigitalOcean Reserved IPs

It's a common challenge that IP addresses of servers can be easily blocked by the Great Firewall (GFW). This is especially true for cloud servers. To mitigate this, one strategy is to use DigitalOcean's reserved IPs and reassign them to your droplet when the current IP is blocked. This post introduces a Python script to automate this process. The script is also open-sourced and available on GitHub.

The script allows you to:

- Check if a reserved IP is assigned to a specific droplet.
- Reassign a new reserved IP to a droplet if the current one is blocked.
- Check if port 80 is open on the reserved IP (a simple way to check if the IP is working).

Here's the Python script:

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

# Function to get DigitalOcean API headers
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"
    }

# Function to fetch all reserved IPs from DigitalOcean
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) # Save the response to a file for debugging
        return reserved_ips_data
    except requests.exceptions.RequestException as e:
        print(f"Error getting reserved IP address: {e}")
        return None

# Function to unassign a reserved IP from a droplet
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

# Function to assign a reserved IP to a droplet
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

# Function to process reserved IPs, check assignment, and reassign if needed
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")
```

```python
                if droplet_id:
                    if unassign_ip_from_droplet(ip_address, droplet_id, droplet_name):
                        # Attempt to assign a new IP after unassigning

                        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

# Function to create a new reserved IP
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", # You can change the region if needed
        }
        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

# Function to check if port 80 is open on an IP address
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

# Main function to get the reserved IP
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}")
```

Explanation:

1. **Import Libraries:** Imports necessary libraries for network operations, environment variables, argument parsing, JSON handling, HTTP requests, and time delays.
2. **`get_digitalocean_headers()`:** Retrieves the DigitalOcean API key from environment variables and constructs the necessary headers for API requests.
3. **`fetch_reserved_ips()`:** Fetches all reserved IPs associated with your DigitalOcean account using the API. It also saves the raw response to `response.json` for debugging purposes.
4. **`unassign_ip_from_droplet()`:** Unassigns a given reserved IP from a specified droplet.
5. **`assign_ip_to_droplet()`:** Assigns a given reserved IP to a specified droplet.
6. **`process_reserved_ips()`:** This is the core logic:
    - It iterates through all reserved IPs.
    - If a `droplet_name` is provided, it checks if the IP is assigned to that droplet.
    - If `only_check` is true, it checks if port 80 is open and returns the IP.
    - If not `only_check`, it unassigns the current IP, creates a new one, and assigns the new IP to the droplet.
7. **`create_new_reserved_ip()`:** Creates a new reserved IP in the `sgp1` region (you can change this).
8. **`check_port_80()`:** Checks if port 80 is open on a given IP address. This is a simple way to verify if the IP is reachable.
9. **`get_reserved_ip()`:** Orchestrates the process of fetching and processing reserved IPs.
10. **`if __name__ == '__main__':`:** Parses command-line arguments (`--droplet-name` and `--only-check`) and calls `get_reserved_ip` to execute the script.

How to Use:

1. **Set up DigitalOcean API Key:** Set the `DO_API_KEY` environment variable with your DigitalOcean API key.
2. **Run the script:**
    - To check if an IP is assigned to a droplet and if port 80 is open: `bash python your_script_name.py --droplet-name your_droplet_name --only-check`
    - To reassign a new IP to a droplet: `bash python your_script_name.py --droplet-name your_droplet_name`

This script provides a basic framework for managing reserved IPs. You can extend it further based on your specific needs.