डिजिटलओशन रिजर्व्ड आईपी प्रबंधित करना

यह एक सामान्य चुनौती है कि सर्वर के □□ पते □□□□□ □□□□□□□□ (□□□) द्वारा आसानी से ब्लॉक किए जा सकते हैं। यह विशेष रूप से क्लाउड सर्वर के लिए सच है। इससे निपटने के लिए एक रणनीति यह है कि □□□□□□□□□□□□□ के आरक्षित □□ का उपयोग किया जाए और जब वर्तमान □□ ब्लॉक हो जाए तो इसे अपने □□□□□□□ पर पुनः असाइन किया जाए। यह पोस्ट इस प्रक्रिया को स्वचालित करने के लिए एक □□□□□□ स्क्रिप्ट का परिचय देती है। यह स्क्रिप्ट □□□□□□ पर भी उपलब्ध है।

स्क्रिप्ट आपको यह करने की अनुमति देती है:

- □ जांचें कि क्या एक आरक्षित □□ किसी विशिष्ट □□□□□□□ को असाइन किया गया है।
- □ यदि वर्तमान □□ ब्लॉक हो गया है तो एक नया आरक्षित □□ □□□□□□□ को पुनः असाइन करें।
- □ जांचें कि आरक्षित □□ पर पोर्ट 80 खुला है या नहीं (यह जांचने का एक सरल तरीका है कि □□ काम कर रहा है या नहीं)।

यहां □□□□□□ स्क्रिप्ट दी गई है:

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

# DigitalOcean API
def get_digitalocean_headers():
    api_key = os.environ.get("DO_API_KEY")
    if not api_key:
        print("Error: DO_API_KEY environment variables        ")
        return None
    return {
        "Authorization": f"Bearer {api_key}",
        "Content-Type": "application/json"
    }

# DigitalOcean          IP
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) #
        return reserved_ips_data
    except requests.exceptions.RequestException as e:
        print(f"    IP              : {e}")
        return None

#   droplet       IP
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}   droplet {droplet_name}                ")
        return True
    except requests.exceptions.RequestException as e:
        print(f"IP {ip_address}   droplet {droplet_name}          : {e}")
        return False

#   droplet       IP
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}   droplet {droplet_name}                  ")
        return True
    except requests.exceptions.RequestException as e:
        print(f"IP {ip_address}   droplet {droplet_name}             : {e}")
        return False

#     IP        ,
def process_reserved_ips(reserved_ips, droplet_name, only_check=False):
    if not reserved_ips:
        print("              IP      ")
        return None

    for reserved_ip in reserved_ips:
        ip_address = reserved_ip.get("ip")
        if not ip_address:
            print("      IP        IP         ")
            continue

        droplet = reserved_ip.get("droplet", None)
        if droplet_name:
            if droplet and droplet.get("name") == droplet_name:
                print(f"    IP {ip_address} droplet: {droplet_name}              ")
                if only_check:
                    if check_port_80(ip_address):
                        print(f"Port 80 {ip_address}   droplet {droplet_name}         ")
                    else:
                        print(f"Port 80 {ip_address}   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):
                        #                 IP

                        new_ip = create_new_reserved_ip(droplet_id)
                        if new_ip:
                            print("   IP           10            ...")
                            time.sleep(10)
                            if assign_ip_to_droplet(new_ip, droplet_id, droplet_name):
                                print(f"   IP {new_ip}   droplet {droplet_name}                  ")
                            else:
                                print(f"   IP {new_ip}   droplet {droplet_name}             ")
                        else:
                            print("                IP     ")

                else:
                    print(f"IP {ip_address}                      droplet ID      ")
                return None
            elif droplet:
                print(f"    IP {ip_address} droplet: {droplet_name}                ")
            else:
                print(f"    IP: {ip_address}      droplet              ")
        else:
            return ip_address
    return None

#          IP
def create_new_reserved_ip(droplet_id):
    headers = get_digitalocean_headers()
    if not headers:
        print("DigitalOcean             ")
        return False
    try:
        url = "https://api.digitalocean.com/v2/reserved_ips"
        req = {
            "region": "sgp1", #
```

```
        }
        print(f"Droplet ID: {droplet_id}              IP                ")
        resp = requests.post(url, headers=headers, json=req)
        resp.raise_for_status()
        new_ip = resp.json().get("reserved_ip", {}).get("ip")
        print(f"       IP            : {new_ip}")
        return new_ip
    except requests.exceptions.RequestException as e:
        print(f"       IP        : {e}")
        return False

#   IP        80         ,
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

#     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="DigitalOcean     IP         ")
    parser.add_argument("--droplet-name", required=True, help="         droplet             IP   droplet
    parser.add_argument("--only-check", action="store_true", help="       IP droplet                  ,
    args = parser.parse_args()

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

```
        print(f"    IP    : {reserved_ip}")
```

व्याख्या:

1. **लाइब्रेरी आयात करें:** नेटवर्क ऑपरेशन, पर्यावरण चर, तर्क पार्सिंग, □□□□ हैंडलिंग, □□□□ अनुरोध और समय विलंब के लिए आवश्यक लाइब्रेरी आयात करें।
2. `get_digitalocean_headers()`: पर्यावरण चर से □□□□□□□□□□□□□ □□□ कुंजी प्राप्त करें और □□□ अनुरोध के लिए आवश्यक हेडर बनाएं।
3. `fetch_reserved_ips()`: □□□ का उपयोग करके आपके □□□□□□□□□□□□□ खाते से जुड़े सभी आरक्षित □□ प्राप्त करें। यह डिबगिंग के लिए कच्ची प्रतिक्रिया को `response.json` में भी सहेजता है।
4. `unassign_ip_from_droplet()`: एक निर्दिष्ट □□□□□□□□ से दिए गए आरक्षित □□ को अनअसाइन करें।
5. `assign_ip_to_droplet()`: एक निर्दिष्ट □□□□□□□□ को दिए गए आरक्षित □□ को असाइन करें।
6. `process_reserved_ips()`: यह मुख्य तर्क है:
    - यह सभी आरक्षित □□ के माध्यम से पुनरावृत्त करता है।
    - यदि `droplet_name` प्रदान किया गया है, तो यह जांचता है कि □□ उस □□□□□□□□ को असाइन किया गया है या नहीं।
    - यदि `only_check` सत्य है, तो यह जांचता है कि पोर्ट 80 खुला है और □□ लौटाता है।
    - यदि `only_check` सत्य नहीं है, तो यह वर्तमान □□ को अनअसाइन करता है, एक नया बनाता है और नए □□ को □□□□□□□□ को असाइन करता है।
7. `create_new_reserved_ip()`: `sgp1` क्षेत्र में एक नया आरक्षित □□ बनाता है (आप इसे बदल सकते हैं)।
8. `check_port_80()`: दिए गए □□ पते पर पोर्ट 80 खुला है या नहीं, यह जांचता है। यह जांचने का एक सरल तरीका है कि □□ पहुंच योग्य है या नहीं।
9. `get_reserved_ip()`: आरक्षित □□ प्राप्त करने और प्रोसेस करने की प्रक्रिया को संचालित करता है।
10. `if __name__ == '__main__'`:: कमांड-लाइन तर्क (`--droplet-name` और `--only-check`) को पार्स करता है और स्क्रिप्ट को निष्पादित करने के लिए `get_reserved_ip` को कॉल करता है।

उपयोग कैसे करें:

1. □□□□□□□□□□□□□ □□□ **कुंजी सेट करें:** अपने □□□□□□□□□□□□□ □□□ कुंजी को `DO_API_KEY` पर्यावरण चर में सेट करें।
2. **स्क्रिप्ट चलाएं:**
    - यह जांचने के लिए कि क्या एक □□ □□□□□□□□ को असाइन किया गया है और पोर्ट 80 खुला है या नहीं: `bash python your_script_name.py --droplet-name your_droplet_name --only-check`
    - एक नया □□ □□□□□□□□ को पुनः असाइन करने के लिए: `bash python your_script_name.py --droplet-name your_droplet_name`

यह स्क्रिप्ट आरक्षित □□ प्रबंधन के लिए एक बुनियादी ढांचा प्रदान करती है। आप इसे अपनी विशिष्ट आवश्यकताओं के आधार पर और विस्तारित कर सकते हैं।