Administración de IPs Elásticas de Aliyun

Este script proporciona una interfaz de línea de comandos para gestionar las IPs elásticas de Aliyun (EIP). Permite crear, enlazar, desenlazar y liberar EIP usando el SDK de Aliyun para Python. El script toma argumentos para la tarea a realizar y el ID de asignación de la EIP.

```
python aliyun_elastic_ip_manager.py unbind --allocation_id eip-j6c2olvsa7jk9l42iaaa
python aliyun_elastic_ip_manager.py bind --allocation_id eip-j6c7mhenamvy6zao3haaa
python aliyun_elastic_ip_manager.py release --allocation_id eip-j6c2olvsa7jk9l42aaa
python aliyun_elastic_ip_manager.py describe
```

```
# -*- coding: utf-8 -*-
# Este archivo es generado automáticamente, no lo edites. Gracias.
import logging
import os
import sys
from typing import List
import argparse
import json

from alibabacloud_vpc20160428.client import Client as Vpc20160428Client
from alibabacloud_tea_openapi import models as open_api_models
from alibabacloud_vpc20160428 import models as vpc_20160428_models
from alibabacloud_tea_util import models as util_models
from alibabacloud_tea_util.client import Client as UtilClient
from alibabacloud_ecs20140526.client import Client as Ecs20140526Client



logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

class Sample:
    def __init__(self):
        pass

    @staticmethod
    def create_client() -> Vpc20160428Client:
        config = open_api_models.Config(
            access_key_id=os.environ['ALIBABA_CLOUD_ACCESS_ID_API_KEY'],
            access_key_secret=os.environ['ALIBABA_CLOUD_ACCESS_API_KEY']
        )
        config.endpoint = f'vpc.cn-hongkong.aliyuncs.com'
```

```
    return Vpc20160428Client(config)

@staticmethod
def bind_eip(
    region_id: str,
    allocation_id: str,
    instance_id: str,
) -> bool:
    client = Sample.create_client()
    associate_eip_address_request = vpc_20160428_models.AssociateEipAddressRequest(
        region_id=region_id,
        allocation_id=allocation_id,
        instance_id=instance_id
    )
    runtime = util_models.RuntimeOptions(read_timeout=60000, connect_timeout=60000)
    try:
        result = client.associate_eip_address_with_options(associate_eip_address_request, runtime)
        logging.info(f"EIP {allocation_id} enlazado correctamente a la instancia {instance_id}. Resultado
        return True
    except Exception as error:
        logging.error(f"Error enlazando EIP {allocation_id} a la instancia {instance_id}: {error}")
        if hasattr(error, 'message'):
            logging.error(f"Mensaje de error: {error.message}")
        if hasattr(error, 'data') and error.data and error.data.get('Recommend'):
            logging.error(f"Recomendación: {error.data.get('Recommend')}")
        UtilClient.assert_as_string(str(error))
        return False

@staticmethod
def unbind_eip(
    region_id: str,
    allocation_id: str,
    instance_id: str,
) -> bool:
    client = Sample.create_client()
    unassociate_eip_address_request = vpc_20160428_models.UnassociateEipAddressRequest(
        region_id=region_id,
        allocation_id=allocation_id,
        instance_id=instance_id
    )
```

```
    runtime = util_models.RuntimeOptions(read_timeout=60000, connect_timeout=60000)
    try:
        result = client.unassociate_eip_address_with_options(unassociate_eip_address_request, runtime)
        logging.info(f"EIP {allocation_id} desenlazado correctamente de la instancia {instance_id}. Result
        return True
    except Exception as error:
        logging.error(f"Error desenlazando EIP {allocation_id} de la instancia {instance_id}: {error}")
        if hasattr(error, 'message'):
            logging.error(f"Mensaje de error: {error.message}")
        if hasattr(error, 'data') and error.data and error.data.get('Recommend'):
            logging.error(f"Recomendación: {error.data.get('Recommend')}")
        UtilClient.assert_as_string(str(error))
        return False

@staticmethod
def create_eip(
    region_id: str,
) -> str | None:
    client = Sample.create_client()
    allocate_eip_address_request = vpc_20160428_models.AllocateEipAddressRequest(
        region_id= region_id,
        instance_charge_type='PostPaid',
        internet_charge_type='PayByTraffic',
        bandwidth='200'
    )
    runtime = util_models.RuntimeOptions(read_timeout=60000, connect_timeout=60000)
    try:
        result = client.allocate_eip_address_with_options(allocate_eip_address_request, runtime)
        print(result.body)
        allocation_id = result.body.allocation_id
        logging.info(f"EIP creado correctamente. ID de asignación: {allocation_id}")
        return allocation_id
    except Exception as error:
        logging.error(f"Error creando EIP: {error}")
        if hasattr(error, 'message'):
            logging.error(f"Mensaje de error: {error.message}")
        if hasattr(error, 'data') and error.data and error.data.get('Recommend'):
            logging.error(f"Recomendación: {error.data.get('Recommend')}")
        UtilClient.assert_as_string(str(error))
        return None
```

```python
@staticmethod
def release_eip(
    allocation_id: str,
) -> bool:
    client = Sample.create_client()
    release_eip_address_request = vpc_20160428_models.ReleaseEipAddressRequest(
        allocation_id=allocation_id
    )
    runtime = util_models.RuntimeOptions(read_timeout=60000, connect_timeout=60000)
    try:
        result = client.release_eip_address_with_options(release_eip_address_request, runtime)
        logging.info(f"EIP {allocation_id} liberado correctamente. Resultado: {result}")
        return True
    except Exception as error:
        logging.error(f"Error liberando EIP {allocation_id}: {error}")
        if hasattr(error, 'message'):
            logging.error(f"Mensaje de error: {error.message}")
        if hasattr(error, 'data') and error.data and error.data.get('Recommend'):
            logging.error(f"Recomendación: {error.data.get('Recommend')}")
        UtilClient.assert_as_string(str(error))
        return False


@staticmethod
def describe_eip(
    region_id: str,
    instance_id: str,
) -> str | None:
    client = Sample.create_client()
    describe_eip_addresses_request = vpc_20160428_models.DescribeEipAddressesRequest(
        region_id=region_id
    )
    runtime = util_models.RuntimeOptions(read_timeout=60000, connect_timeout=60000)
    try:
        result = client.describe_eip_addresses_with_options(describe_eip_addresses_request, runtime)
        logging.info(f"EIP descrito correctamente.")
        print(json.dumps(result.body.to_map(), indent=4))
        if result.body.eip_addresses and hasattr(result.body.eip_addresses, 'eip_address') and result.body
            for eip in result.body.eip_addresses.eip_address:
                if eip.instance_id == instance_id:
```

```
                    return eip.allocation_id
            logging.info(f"No se encontró ninguna dirección EIP para la instancia {instance_id}")
            return None
        else:
            logging.info("No se encontraron direcciones EIP.")
            return None
    except Exception as error:
        logging.error(f"Error describiendo EIP: {error}")
        if hasattr(error, 'message'):
            logging.error(f"Mensaje de error: {error.message}")
        if hasattr(error, 'data') and error.data and error.data.get('Recommend'):
            logging.error(f"Recomendación: {error.data.get('Recommend')}")
        UtilClient.assert_as_string(str(error))
        return None

@staticmethod
def main(
    args: List[str],
) -> None:
    region_id = "cn-hongkong"
    instance_id = "i-j6c44l4zpphv7u7agdbk"

    parser = argparse.ArgumentParser(description='Gestionar IPs elásticas de Aliyun.')
    parser.add_argument('job', choices=['create', 'bind', 'unbind', 'release', 'describe', 'all'], help='I
    parser.add_argument('--allocation_id', type=str, help='El ID de asignación de la EIP.')
    parser.add_argument('--instance_id', type=str, default=instance_id, help='El ID de instancia al que en

    parsed_args = parser.parse_args(args)

    if parsed_args.job == 'create':
        new_allocation_id = Sample.create_eip(region_id)
        if new_allocation_id:
            print(f"Proceso de creación de EIP iniciado correctamente. ID de asignación: {new_allocation_i
        else:
            print("Proceso de creación de EIP fallido.")
    elif parsed_args.job == 'bind':
        if not parsed_args.allocation_id:
            print("Error: --allocation_id es requerido para la tarea bind.")
            return
        if Sample.bind_eip(region_id, parsed_args.allocation_id, parsed_args.instance_id):
```

```
            print(f"Proceso de enlace de EIP iniciado correctamente para EIP {parsed_args.allocation_id} e
        else:
            print(f"Proceso de enlace de EIP fallido para EIP {parsed_args.allocation_id} e instancia {par
    elif parsed_args.job == 'unbind':
        if not parsed_args.allocation_id:
            print("Error: --allocation_id es requerido para la tarea unbind.")
            return
        if Sample.unbind_eip(region_id, parsed_args.allocation_id, parsed_args.instance_id):
            print(f"Proceso de desenlace de EIP iniciado correctamente para EIP {parsed_args.allocation_id
        else:
            print(f"Proceso de desenlace de EIP fallido para EIP {parsed_args.allocation_id} e instancia
    elif parsed_args.job == 'release':
        if not parsed_args.allocation_id:
            print("Error: --allocation_id es requerido para la tarea release.")
            return
        if Sample.release_eip(parsed_args.allocation_id):
             print(f"Proceso de liberación de EIP iniciado correctamente para EIP {parsed_args.allocation_
        else:
            print(f"Proceso de liberación de EIP fallido para EIP {parsed_args.allocation_id}.")
    elif parsed_args.job == 'describe':
        if not parsed_args.instance_id:
            print("Error: --instance_id es requerido para la tarea describe.")
            return
        allocation_id = Sample.describe_eip(region_id, parsed_args.instance_id)
        if allocation_id:
            print(f"ID de asignación: {allocation_id}")
        else:
            print("No se encontró ninguna EIP.")
    elif parsed_args.job == 'all':
        # Describir para obtener el ID de asignación actual
        current_allocation_id = Sample.describe_eip(region_id, parsed_args.instance_id)
        if current_allocation_id:
            print(f"ID de asignación actual: {current_allocation_id}")
        else:
            print("No se encontró ninguna EIP para procesar.")
            return

        # Desenlazar la EIP actual
        if current_allocation_id:
            if Sample.unbind_eip(region_id, current_allocation_id, parsed_args.instance_id):
```

```
                print(f"EIP {current_allocation_id} desenlazado correctamente de la instancia {parsed_args
            else:
                print(f"Error al desenlazar EIP {current_allocation_id} de la instancia {parsed_args.insta
                return

        # Crear una nueva EIP
        new_allocation_id = Sample.create_eip(region_id)
        if new_allocation_id:
            print(f"Proceso de creación de EIP iniciado correctamente. Nuevo ID de asignación: {new_alloca
        else:
            print("Proceso de creación de EIP fallido.")
            return

        # Enlazar la nueva EIP
        if Sample.bind_eip(region_id, new_allocation_id, parsed_args.instance_id):
            print(f"Nueva EIP {new_allocation_id} enlazada correctamente a la instancia {parsed_args.insta
        else:
            print(f"Error al enlazar la nueva EIP {new_allocation_id} a la instancia {parsed_args.instance
            return

        # Liberar la EIP antigua
        if current_allocation_id:
            if Sample.release_eip(current_allocation_id):
                print(f"EIP antigua {current_allocation_id} liberada correctamente.")
            else:
                print(f"Error al liberar la EIP antigua {current_allocation_id}.")

        # Describir de nuevo para mostrar el estado final
        final_allocation_id = Sample.describe_eip(region_id, parsed_args.instance_id)
        if final_allocation_id:
            print(f"ID de asignación final: {final_allocation_id}")
        else:
            print("No se encontró ninguna EIP después del procesamiento.")


@staticmethod
async def main_async(
    args: List[str],
) -> None:
    client = Sample.create_client()
```

```
        associate_eip_address_request = vpc_20160428_models.AssociateEipAddressRequest(
            region_id='cn-hongkong'
        )
        runtime = util_models.RuntimeOptions()
        try:
            await client.associate_eip_address_with_options_async(associate_eip_address_request, runtime)
        except Exception as error:
            print(error)
            if hasattr(error, 'message'):
                print(error.message)
            if hasattr(error, 'data') and error.data.get("Recommend"):
                print(error.data.get("Recommend"))
            UtilClient.assert_as_string(str(error))


if __name__ == '__main__':
    Sample.main(sys.argv[1:])

# python scripts/auto-ss-config/aliyun_elastic_ip_manager.py create
# python scripts/auto-ss-config/aliyun_elastic_ip_manager.py unbind --allocation_id eip-j6c2olvsa7jk9l42i1aaa
# python scripts/auto-ss-config/aliyun_elastic_ip_manager.py bind --allocation_id eip-j6c7mhenamvy6zao3haaa
# python scripts/auto-ss-config/aliyun_elastic_ip_manager.py release --allocation_id "eip-j6c2olvsa7jk9l42i"
# python scripts/auto-ss-config/aliyun_elastic_ip_manager.py describe
# python scripts/auto-ss-config/aliyun_elastic_ip_manager.py all
```