Gestion des adresses IP élastiques Aliyun

Ce script fournit une interface en ligne de commande pour gérer les adresses IP élastiques (EIP) Aliyun. Il vous permet de créer, lier, délier et libérer des EIP à l'aide du SDK Aliyun pour Python. Le script prend en argument la tâche à effectuer et l'ID d'allocation de l'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
```

```
import logging
import os
import sys
from typing import List
import argparse

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} lié avec succès à l'instance {instance_id}. Résultat : {result}
        return True
    except Exception as error:
        logging.error(f"Erreur lors du lien de l'EIP {allocation_id} à l'instance {instance_id} : {error}'
        if hasattr(error, 'message'):
            logging.error(f"Message d'erreur : {error.message}")
        if hasattr(error, 'data') and error.data and error.data.get('Recommend'):
            logging.error(f"Recommandation : {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} délié avec succès de l'instance {instance_id}. Résultat : {resu
```

```
            return True
        except Exception as error:
            logging.error(f"Erreur lors du délien de l'EIP {allocation_id} de l'instance {instance_id} : {erro
            if hasattr(error, 'message'):
                logging.error(f"Message d'erreur : {error.message}")
            if hasattr(error, 'data') and error.data and error.data.get('Recommend'):
                logging.error(f"Recommandation : {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
        )
        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 créé avec succès. ID d'allocation : {allocation_id}")
            return allocation_id
        except Exception as error:
            logging.error(f"Erreur lors de la création de l'EIP : {error}")
            if hasattr(error, 'message'):
                logging.error(f"Message d'erreur : {error.message}")
            if hasattr(error, 'data') and error.data and error.data.get('Recommend'):
                logging.error(f"Recommandation : {error.data.get('Recommend')}")
            UtilClient.assert_as_string(str(error))
            return None

    @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} libéré avec succès. Résultat : {result}")
            return True
        except Exception as error:
            logging.error(f"Erreur lors de la libération de l'EIP {allocation_id} : {error}")
            if hasattr(error, 'message'):
                logging.error(f"Message d'erreur : {error.message}")
            if hasattr(error, 'data') and error.data and error.data.get('Recommend'):
                logging.error(f"Recommandation : {error.data.get('Recommend')}")
            UtilClient.assert_as_string(str(error))
            return False


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

        parser = argparse.ArgumentParser(description='Gestion des adresses IP élastiques Aliyun.')
        parser.add_argument('job', choices=['create', 'bind', 'unbind', 'release'], help='La tâche à effectuer
        parser.add_argument('--allocation_id', type=str, help='L\'ID d\'allocation de l\'EIP.')
        parser.add_argument('--instance_id', type=str, default=instance_id, help='L\'ID de l\'instance auquel

        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"Processus de création de l'EIP initié avec succès. ID d'allocation : {new_allocation_i
            else:
                print("Processus de création de l'EIP échoué.")
        elif parsed_args.job == 'bind':
            if not parsed_args.allocation_id:
                print("Erreur : --allocation_id est requis pour la tâche de lien.")
```

```
            return
        if Sample.bind_eip(region_id, parsed_args.allocation_id, parsed_args.instance_id):
            print(f"Processus de lien de l'EIP initié avec succès pour l'EIP {parsed_args.allocation_id} e
        else:
            print(f"Processus de lien de l'EIP échoué pour l'EIP {parsed_args.allocation_id} et l'instance
    elif parsed_args.job == 'unbind':
        if not parsed_args.allocation_id:
            print("Erreur : --allocation_id est requis pour la tâche de délien.")
            return
        if Sample.unbind_eip(region_id, parsed_args.allocation_id, parsed_args.instance_id):
            print(f"Processus de délien de l'EIP initié avec succès pour l'EIP {parsed_args.allocation_id
        else:
            print(f"Processus de délien de l'EIP échoué pour l'EIP {parsed_args.allocation_id} et l'instan
    elif parsed_args.job == 'release':
        if not parsed_args.allocation_id:
            print("Erreur : --allocation_id est requis pour la tâche de libération.")
            return
        if Sample.release_eip(parsed_args.allocation_id):
             print(f"Processus de libération de l'EIP initié avec succès pour l'EIP {parsed_args.allocatio
        else:
            print(f"Processus de libération de l'EIP échoué pour l'EIP {parsed_args.allocation_id}.")


@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:])
```