إدارة عناوين □□ المرنة على علي بابا كلاود

يوفر هذا البرنامج النصي واجهة سطر أوامر لإدارة عناوين □□ المرنة من علي بابا كلاود □□□□□□. يسمح لك بإنشاء عناوين □□□□، وربطها، وفصلها، وإصدارها باستخدام مجموعة أدوات تطوير البرامج □□□□□ الخاصة بعلي بابا كلاود للغة بايثون. يأخذ البرنامج النصي وسيطات للمهمة المراد تنفيذها ومعرف التخصيص للعنوان □□□.

```
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
```

```python
# -*- coding: utf-8 -*-
#                         .        .
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"Successfully bound EIP {allocation_id} to instance {instance_id}. Result: {result}'
        return True
    except Exception as error:
        logging.error(f"Error binding EIP {allocation_id} to instance {instance_id}: {error}")
        if hasattr(error, 'message'):
            logging.error(f"Error message: {error.message}")
        if hasattr(error, 'data') and error.data and error.data.get('Recommend'):
            logging.error(f"Recommend: {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"Successfully unbound EIP {allocation_id} from instance {instance_id}. Result: {resu
            return True
        except Exception as error:
            logging.error(f"Error unbinding EIP {allocation_id} from instance {instance_id}: {error}")
            if hasattr(error, 'message'):
                logging.error(f"Error message: {error.message}")
            if hasattr(error, 'data') and error.data and error.data.get('Recommend'):
                logging.error(f"Recommend: {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='PayByBandwidth',
            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"Successfully created EIP. Allocation ID: {allocation_id}")
            return allocation_id
        except Exception as error:
            logging.error(f"Error creating EIP: {error}")
            if hasattr(error, 'message'):
                logging.error(f"Error message: {error.message}")
            if hasattr(error, 'data') and error.data and error.data.get('Recommend'):
                logging.error(f"Recommend: {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"Successfully released EIP {allocation_id}. Result: {result}")
            return True
        except Exception as error:
            logging.error(f"Error releasing EIP {allocation_id}: {error}")
            if hasattr(error, 'message'):
                logging.error(f"Error message: {error.message}")
            if hasattr(error, 'data') and error.data and error.data.get('Recommend'):
                logging.error(f"Recommend: {error.data.get('Recommend')}")
            UtilClient.assert_as_string(str(error))
            return False

    @staticmethod
    def describe_eip(
        region_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"Successfully described EIP.")
            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
                if len(result.body.eip_addresses.eip_address) > 0:
                    first_allocation_id = result.body.eip_addresses.eip_address[0].allocation_id
```

```
                return first_allocation_id
            else:
                logging.info("No EIP addresses found.")
                return None
        else:
            logging.info("No EIP addresses found.")
            return None
    except Exception as error:
        logging.error(f"Error describing EIP: {error}")
        if hasattr(error, 'message'):
            logging.error(f"Error message: {error.message}")
        if hasattr(error, 'data') and error.data and error.data.get('Recommend'):
            logging.error(f"Recommend: {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='Manage Aliyun Elastic IPs.')
    parser.add_argument('job', choices=['create', 'bind', 'unbind', 'release', 'describe', 'all'], help='T
    parser.add_argument('--allocation_id', type=str, help='The allocation ID of the EIP.')
    parser.add_argument('--instance_id', type=str, default=instance_id, help='The instance ID to bind/unbi

    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"EIP creation process initiated successfully. Allocation ID: {new_allocation_id}")
        else:
            print("EIP creation process failed.")
    elif parsed_args.job == 'bind':
        if not parsed_args.allocation_id:
            print("Error: --allocation_id is required for bind job.")
            return
```

```
    if Sample.bind_eip(region_id, parsed_args.allocation_id, parsed_args.instance_id):
        print(f"EIP binding process initiated successfully for EIP {parsed_args.allocation_id} and ins
    else:
        print(f"EIP binding process failed for EIP {parsed_args.allocation_id} and instance {parsed_ar
elif parsed_args.job == 'unbind':
    if not parsed_args.allocation_id:
        print("Error: --allocation_id is required for unbind job.")
        return
    if Sample.unbind_eip(region_id, parsed_args.allocation_id, parsed_args.instance_id):
        print(f"EIP unbinding process initiated successfully for EIP {parsed_args.allocation_id} and i
    else:
        print(f"EIP unbinding process failed for EIP {parsed_args.allocation_id} and instance {parsed_
elif parsed_args.job == 'release':
    if not parsed_args.allocation_id:
        print("Error: --allocation_id is required for release job.")
        return
    if Sample.release_eip(parsed_args.allocation_id):
         print(f"EIP release process initiated successfully for EIP {parsed_args.allocation_id}.")
    else:
        print(f"EIP release process failed for EIP {parsed_args.allocation_id}.")
elif parsed_args.job == 'describe':
    Sample.describe_eip(region_id)
elif parsed_args.job == 'all':
    # Describe to get current allocation ID
    current_allocation_id = Sample.describe_eip(region_id)
    if current_allocation_id:
        print(f"Current Allocation ID: {current_allocation_id}")
    else:
        print("No EIP found to process.")
        return

    # Unbind current EIP
    if current_allocation_id:
        if Sample.unbind_eip(region_id, current_allocation_id, parsed_args.instance_id):
            print(f"Successfully unbound EIP {current_allocation_id} from instance {parsed_args.instan
        else:
            print(f"Failed to unbind EIP {current_allocation_id} from instance {parsed_args.instance_i
            return

    # Create new EIP
```

```
        new_allocation_id = Sample.create_eip(region_id)
        if new_allocation_id:
            print(f"EIP creation process initiated successfully. New Allocation ID: {new_allocation_id}")
        else:
            print("EIP creation process failed.")
            return

        # Bind new EIP
        if Sample.bind_eip(region_id, new_allocation_id, parsed_args.instance_id):
            print(f"Successfully bound new EIP {new_allocation_id} to instance {parsed_args.instance_id}.'
        else:
            print(f"Failed to bind new EIP {new_allocation_id} to instance {parsed_args.instance_id}.")
            return

        # Release old EIP
        if current_allocation_id:
            if Sample.release_eip(current_allocation_id):
                print(f"Successfully released old EIP {current_allocation_id}.")
            else:
                print(f"Failed to release old EIP {current_allocation_id}.")

        # Describe again to show the final state
        final_allocation_id = Sample.describe_eip(region_id)
        if final_allocation_id:
            print(f"Final Allocation ID: {final_allocation_id}")
        else:
            print("No EIP found after processing.")


@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
```