lightsail

Aquí tienes una política que otorga los permisos necesarios para gestionar instancias de Lightsail:

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "lightsail:CreateRelationalDatabaseSnapshot",
                "lightsail:GetRelationalDatabaseEvents",
                "lightsail:CreateContainerService",
                "lightsail:GetKeyPair",
                "lightsail:GetContactMethods",
                "lightsail:GetCloudFormationStackRecords",
                "lightsail:GetContainerServiceDeployments",
                "lightsail:GetBucketAccessKeys",
                "lightsail:CreateContainerServiceRegistryLogin",
                "lightsail:GetContainerImages",
                "lightsail:CreateRelationalDatabase",
                "lightsail:CreateContactMethod",
                "lightsail:CreateDistribution",
                "lightsail:GetDomain",
                "lightsail:GetBuckets",
                "lightsail:GetRelationalDatabaseParameters",
                "lightsail:GetInstanceState",
                "lightsail:GetOperationsForResource",
                "lightsail:AllocateStaticIp",
                "lightsail:GetInstances",
                "lightsail:GetRelationalDatabase",
                "lightsail:CreateLoadBalancer",
                "lightsail:GetDistributionLatestCacheReset",
                "lightsail:GetLoadBalancerTlsPolicies",
                "lightsail:GetLoadBalancers",
                "lightsail:GetExportSnapshotRecords",
                "lightsail:GetAutoSnapshots",
                "lightsail:GetStaticIp",
                "lightsail:GetRelationalDatabaseBundles",
```

```
"lightsail:GetRelationalDatabaseBlueprints",

"lightsail:CreateInstances",

"lightsail:GetRelationalDatabaseLogEvents",

"lightsail:GetContainerServices",

"lightsail:GetRelationalDatabaseSnapshot",

"lightsail:GetInstancePortStates",

"lightsail:DeleteContactMethod",

"lightsail:GetContainerServicePowers",

"lightsail:GetKeyPairs",

"lightsail:GetLoadBalancer",

"lightsail:DisableAddOn",

"lightsail:CreateCloudFormationStack",

"lightsail:GetRelationalDatabaseSnapshots",

"lightsail:UnpeerVpc",

"lightsail:GetLoadBalancerTlsCertificates",

"lightsail:GetAlarms",

"lightsail:GetInstance",

"lightsail:CreateDomain",

"lightsail:GetDiskSnapshots",

"lightsail:GetRelationalDatabaseMetricData",

"lightsail:PeerVpc",

"lightsail:CreateCertificate",

"lightsail:CreateKeyPair",

"lightsail:SendContactMethodVerification",

"lightsail:GetStaticIps",

"lightsail:GetRegions",

"lightsail:GetOperation",

"lightsail:GetDistributions",

"lightsail:GetDomains",

"lightsail:GetDisks",

"lightsail:CreateDisk",

"lightsail:GetBundles",

"lightsail:GetInstanceMetricData",

"lightsail:GetBucketBundles",

"lightsail:GetContainerServiceMetricData",

"lightsail:GetActiveNames",

"lightsail:GetInstanceSnapshot",

"lightsail:GetOperations",

"lightsail:EnableAddOn",

"lightsail:GetDistributionBundles",
```

```
                "lightsail:GetBlueprints",
                "lightsail:GetContainerAPIMetadata",
                "lightsail:GetCertificates",
                "lightsail:GetLoadBalancerMetricData",
                "lightsail:GetDiskSnapshot",
                "lightsail:DeleteAutoSnapshot",
                "lightsail:CopySnapshot",
                "lightsail:GetDisk",
                "lightsail:GetDistributionMetricData",
                "lightsail:GetRelationalDatabases",
                "lightsail:GetContainerLog",
                "lightsail:GetBucketMetricData",
                "lightsail:ImportKeyPair",
                "lightsail:DownloadDefaultKeyPair",
                "lightsail:IsVpcPeered",
                "lightsail:GetInstanceSnapshots",
                "lightsail:CreateBucket",
                "lightsail:GetRelationalDatabaseLogStreams",
                "lightsail:DeleteInstance",
                "lightsail:DeleteInstanceSnapshot",
                "lightsail:OpenInstancePublicPorts"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "lightsail:*",
                "network-firewall:*"
            ],
            "Resource": "arn:aws:lightsail:*:464063468077:Bucket/*"
        }
    ]
}
```

Las acciones clave incluidas en esta política son:

```
    "lightsail:DeleteInstance",
    "lightsail:DeleteInstanceSnapshot",
    "lightsail:OpenInstancePublicPorts"
```

Esta política puede adjuntarse a un usuario o rol para otorgar los permisos necesarios.

'''python import subprocess import random import string import argparse import yaml import os

KEY_PATH = os.path.expanduser("~/Downloads/LightsailDefaultKey-ap-northeast-1.pem")

def _get_lightsail_instances(): print("Obteniendo instancias de Lightsail...") try: result = subprocess.run(["aws", "lightsail", "get-instances"], capture_output=True, text=True, check=True) print("Instancias de Lightsail obtenidas con éxito.") return yaml.safe_load(result.stdout) except subprocess.CalledProcessError as e: print(f"Error al obtener instancias de Lightsail: {e}") return None except yaml.YAMLError as e: print(f"Error al decodificar la respuesta YAML: {e}") return None except Exception as e: print(f"Ocurrió un error inesperado: {e}") return None

def _get_lightsail_instance(instance_name): print(f"Obteniendo detalles de la instancia: {instance_name}") try: result = subprocess.run(["aws", "lightsail", "get-instance", "–instance-name", instance_name], capture_output=True, text=True, check=True) instance_data = yaml.safe_load(result.stdout) if not instance_data or 'instance'not in instance_data: print(f"No se encontró la instancia con nombre: {instance_name}") return None return instance_data['instance'] except subprocess.CalledProcessError as e: print(f"Error al obtener detalles de la instancia: {e}") return None except yaml.YAMLError as e: print(f"Error al decodificar la respuesta YAML: {e}") return None except Exception as e: print(f"Ocurrió un error inesperado: {e}") return None

def create_lightsail_instance(instance_name=None, availability_zone="ap-northeast-1a", bundle_id="nano_2_0", user_data=None): if not instance_name: random_chars = ''.join(random.choice(string.ascii_lowercase) for _ in range(4)) instance_name = f"{random_chars}"

```
if not user_data:
    user_data = """#!/bin/bash
    sudo apt update
    """
print(f"Creando instancia de Lightsail con nombre: {instance_name}, zona: {availability_zone}, bundle: {bundle

command = [
    "aws", "lightsail", "create-instances",
    "--instance-names", instance_name,
    "--availability-zone", availability_zone,
    "--bundle-id", bundle_id,
    "--blueprint-id", "ubuntu_24_04"
]

if user_data:
    command.extend(["--user-data", user_data])

try:
```

```python
        subprocess.run(command, check=True)
        print(f"Instancia de Lightsail '{instance_name}' creada con éxito.")
        return instance_name
    except subprocess.CalledProcessError as e:
        print(f"Error al crear la instancia de Lightsail: {e}")
        return None
```

def delete_all_lightsail_instances(instance_name=None): if instance_name: print(f"Eliminando instancia: {instance_name}") print(f"Ejecutando comando: aws lightsail delete-instance --instance-name {instance_name}") try: subprocess.run(["aws", "lightsail", "delete-instance", "--instance-name", instance_name], check=True) print(f"Instancia de Lightsail '{instance_name}'eliminada con éxito.") except subprocess.CalledProcessError as e: print(f"Error al eliminar la instancia de Lightsail: {e}") return

```python
    instances_yaml = _get_lightsail_instances()
    if not instances_yaml or 'instances' not in instances_yaml:
        print("No se encontraron instancias de Lightsail para eliminar.")
        return


    instance_list = instances_yaml['instances']
    if not instance_list:
        print("No se encontraron instancias de Lightsail para eliminar.")
        return


    for instance in instance_list:
        instance_name = instance['name']
        print(f"Eliminando instancia: {instance_name}")
        print(f"Ejecutando comando: aws lightsail delete-instance --instance-name {instance_name}")
        subprocess.run(["aws", "lightsail", "delete-instance", "--instance-name", instance_name], check=True)
    print("Todas las instancias de Lightsail eliminadas con éxito.")
```

def install_outline_server(instance_name): instance = _get_lightsail_instance(instance_name) if not instance: return public_ip = instance['publicIpAddress'] print(f"Instalando servidor Outline en la instancia: {instance_name} con IP: {public_ip}") user_data =""""#!/bin/bash sudo apt update sudo bash -c"$(wget -qO- https://raw.githubusercontent.com/Jigsaw-Code/outline-server/master/src/server_manager/install_scripts/instal

```python
    os.chmod(KEY_PATH, 0o600)
    print(f"Ejecutando comando: chmod 600 {KEY_PATH}")


    ssh_command = [
        "ssh",
        "-i",
```

```python
        KEY_PATH,
        f"ubuntu@{public_ip}",
        user_data
    ]
    print(f"Ejecutando comando: {' '.join(ssh_command)}")
    try:
        subprocess.run(ssh_command, check=True)
        print(f"Servidor Outline instalado en {instance_name} con éxito.")
    except subprocess.CalledProcessError as e:
        print(f"Error al instalar el servidor Outline: {e}")
```