

Lightsail

Voici une politique qui accorde les permissions nécessaires pour gérer les instances Lightsail :

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "VisualEditor0",  
            "Effect": "Allow",  
            "Action": [  
                "lightsail>CreateRelationalDatabaseSnapshot",  
                "lightsail>GetRelationalDatabaseEvents",  
                "lightsail>CreateContainerService",  
                "lightsail>GetKeyValuePair",  
                "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:*
```

Les actions clés incluses dans cette politique sont :

```

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

Cette politique peut être attachée à un utilisateur ou à un rôle pour accorder les permissions nécessaires.

```
"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("Récupération des instances Lightsail...") try: result = subprocess.run(["aws", "lightsail", "get-instances"], capture_output=True, text=True, check=True) print("Instances Lightsail récupérées avec succès.") return yaml.safe_load(result.stdout) except subprocess.CalledProcessError as e: print(f"Erreur lors de la récupération des instances Lightsail : {e}") return None except yaml.YAMLError as e: print(f"Erreur lors du décodage de la réponse YAML : {e}") return None except Exception as e: print(f"Une erreur inattendue s'est produite : {e}") return None

def _get_lightsail_instance(instance_name): print(f"Récupération des détails de l'instance : {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"Impossible de trouver l'instance avec le nom : {instance_name}") return None return instance_data['instance'] except subprocess.CalledProcessError as e: print(f"Erreur lors de la récupération des détails de l'instance : {e}") return None except yaml.YAMLError as e: print(f"Erreur lors du décodage de la réponse YAML : {e}") return None except Exception as e: print(f"Une erreur inattendue s'est produite : {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"Création de l'instance Lightsail avec le nom : {instance_name}, zone : {availability_zone}, bundle : {bundle_id}")

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])
```

```

subprocess.run(command, check=True)

print(f"Instance Lightsail '{instance_name}' créée avec succès.")

return instance_name

except subprocess.CalledProcessError as e:
    print(f"Erreur lors de la création de l'instance Lightsail : {e}")
    return None

def delete_all_lightsail_instances(instance_name=None):
    if instance_name:
        print(f"Suppression de l'instance : {instance_name}")
        print(f"Exécution de la commande : aws lightsail delete-instance --instance-name {instance_name}")
        try:
            subprocess.run(["aws", "lightsail", "delete-instance", "--instance-name", instance_name], check=True)
            print(f"Instance Lightsail '{instance_name}' supprimée avec succès.")
        except subprocess.CalledProcessError as e:
            print(f"Erreur lors de la suppression de l'instance Lightsail : {e}")
    return

instances_yaml = _get_lightsail_instances()

if not instances_yaml or 'instances' not in instances_yaml:
    print("Aucune instance Lightsail trouvée à supprimer.")
    return

instance_list = instances_yaml['instances']

if not instance_list:
    print("Aucune instance Lightsail trouvée à supprimer.")
    return

for instance in instance_list:
    instance_name = instance['name']
    print(f"Suppression de l'instance : {instance_name}")
    print(f"Exécution de la commande : aws lightsail delete-instance --instance-name {instance_name}")
    subprocess.run(["aws", "lightsail", "delete-instance", "--instance-name", instance_name], check=True)
print("Toutes les instances Lightsail ont été supprimées avec succès.")

def install_outline_server(instance_name):
    instance = _get_lightsail_instance(instance_name)
    if not instance:
        return
    public_ip = instance['publicIpAddress']
    print(f"Installation du serveur Outline sur l'instance : {instance_name} avec l'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/install.sh)" """
    os.chmod(KEY_PATH, 0o600)
    print(f"Exécution de la commande : chmod 600 {KEY_PATH}")

    ssh_command = [
        "ssh",

```

```
"-i",
KEY_PATH,
f"ubuntu@{public_ip}",
user_data
]

print(f"Exécution de la commande : {' '.join(ssh_command)}")

try:
    subprocess.run(ssh_command, check=True)
    print(f"Le serveur Outline a été installé sur {instance_name} avec succès.")
except subprocess.CalledProcessError as e:
    print(f"Erreur lors de l'installation du serveur Outline : {e}")
```