lightsail

Here's a policy that grants necessary permissions for managing Lightsail instances:


```
{
    "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/*"
        }
    ]
}
```

Key actions included in this policy are:

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

This policy can be attached to a user or role to grant the necessary permissions.

```
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("Fetching Lightsail instances...")
    try:
        result = subprocess.run(["aws", "lightsail", "get-instances"], capture_output=True, text=True, check=T
        print("Lightsail instances fetched successfully.")
        return yaml.safe_load(result.stdout)
    except subprocess.CalledProcessError as e:
        print(f"Error getting Lightsail instances: {e}")
        return None
    except yaml.YAMLError as e:
        print(f"Error decoding YAML response: {e}")
        return None
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
        return None

def _get_lightsail_instance(instance_name):
    print(f"Fetching details for instance: {instance_name}")
    try:
        result = subprocess.run(["aws", "lightsail", "get-instance", "--instance-name", instance_name], captur
        instance_data = yaml.safe_load(result.stdout)
        if not instance_data or 'instance' not in instance_data:
            print(f"Could not find instance with name: {instance_name}")
            return None
        return instance_data['instance']
    except subprocess.CalledProcessError as e:
        print(f"Error getting instance details: {e}")
        return None
    except yaml.YAMLError as e:
```

```
        print(f"Error decoding YAML response: {e}")
        return None
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
        return None


def create_lightsail_instance(instance_name=None, availability_zone="ap-northeast-1a", bundle_id="nano_2_0", u
    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"Creating Lightsail instance with name: {instance_name}, zone: {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:
        subprocess.run(command, check=True)
        print(f"Lightsail instance '{instance_name}' created successfully.")
        return instance_name
    except subprocess.CalledProcessError as e:
        print(f"Error creating Lightsail instance: {e}")
        return None

def delete_all_lightsail_instances(instance_name=None):
    if instance_name:
        print(f"Deleting instance: {instance_name}")
```

```
        print(f"Executing command: aws lightsail delete-instance --instance-name {instance_name}")
        try:
            subprocess.run(["aws", "lightsail", "delete-instance", "--instance-name", instance_name], check=Tr
            print(f"Lightsail instance '{instance_name}' deleted successfully.")
        except subprocess.CalledProcessError as e:
            print(f"Error deleting Lightsail instance: {e}")
        return

    instances_yaml = _get_lightsail_instances()
    if not instances_yaml or 'instances' not in instances_yaml:
        print("No Lightsail instances found to delete.")
        return

    instance_list = instances_yaml['instances']
    if not instance_list:
        print("No Lightsail instances found to delete.")
        return

    for instance in instance_list:
        instance_name = instance['name']
        print(f"Deleting instance: {instance_name}")
        print(f"Executing command: aws lightsail delete-instance --instance-name {instance_name}")
        subprocess.run(["aws", "lightsail", "delete-instance", "--instance-name", instance_name], check=True)
    print("All Lightsail instances deleted successfully.")


def install_outline_server(instance_name):
    instance = _get_lightsail_instance(instance_name)
    if not instance:
        return
    public_ip = instance['publicIpAddress']
    print(f"Installing outline server on instance: {instance_name} with 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_m
    """


    os.chmod(KEY_PATH, 0o600)
    print(f"Executing command: chmod 600 {KEY_PATH}")
```

```
    ssh_command = [
        "ssh",
        "-i",
        KEY_PATH,
        f"ubuntu@{public_ip}",
        user_data
    ]
    print(f"Executing command: {' '.join(ssh_command)}")
    try:
        subprocess.run(ssh_command, check=True)
        print(f"Outline server installed on {instance_name} successfully.")
    except subprocess.CalledProcessError as e:
        print(f"Error installing outline server: {e}")



def open_firewall_ports(instance_name):
    print(f"Opening firewall ports for instance: {instance_name}")
    for protocol in ["tcp", "udp"]:
        command = [
            "aws", "lightsail", "open-instance-public-ports",
            "--port-info", f"protocol={protocol},fromPort=1000,toPort=65535",
            "--instance-name", instance_name
        ]
        print(f"Executing command: {' '.join(command)}")
        subprocess.run(command, check=True)
    print(f"Firewall ports opened for instance '{instance_name}' successfully.")



if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Create or delete Lightsail instances.")
    parser.add_argument("--job", type=str, choices=["create", "delete", "install"], required=True, help="Actio
    args = parser.parse_args()

    print(f"Setting AWS region to ap-northeast-1")
    subprocess.run(["aws", "configure", "set", "region", "ap-northeast-1"], check=True)

    if args.job == "create":
        instance_name = create_lightsail_instance()
```

```
    if instance_name:
        open_firewall_ports(instance_name)
elif args.job == "delete":
    instance_name = input("Enter the instance name to delete: ")
    delete_all_lightsail_instances(instance_name)
elif args.job == "install":
    instance_name = input("Enter the instance name to install Outline server on: ")
    install_outline_server(instance_name)
```