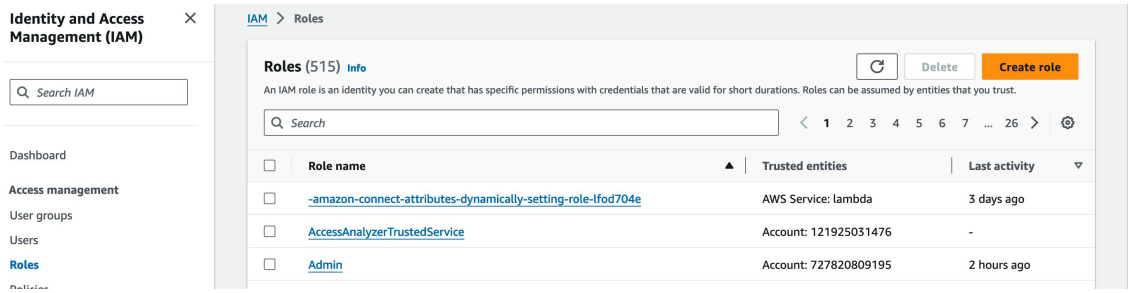
使用 **boto3** 时的 **aws** 权限获取-以 **bedrock** 为例

boto3 的权限获取，如果不是根据当前环境中通过 awscli 已经配置好的 profile，就需要在代码里调用 sts 服务来获取临时的 aksk。
本文通过 bedrcok 里调用大模型的例子来解释这个过程。

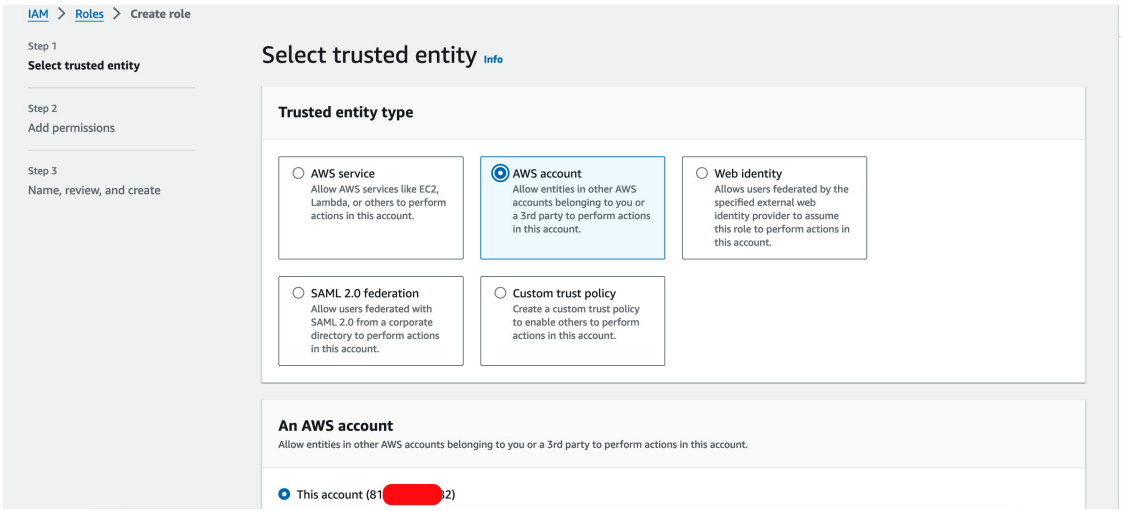
在 **aws** 控制台创建一个 **IAM** 角色

这个角色用来给之前创建的用户扮演，这个扮演的过程是在 python 的脚本里通过 boto3 支持的 STS 服务来完成，具体过程在代码里有注视，这里只说明创建的角色应该被赋予的基本权限。过程如下：

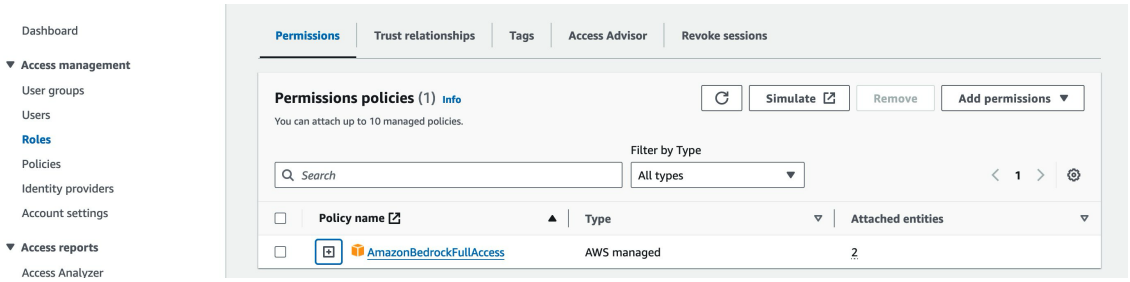
1. 在 IAM 控制台找到 Roles 菜单，点击右上角的‘Create role'



2. 设定这个 role 的扮演者是本账号的用户



3. 给这个角色设定权限，这里是给了 Bedrock 的 full access

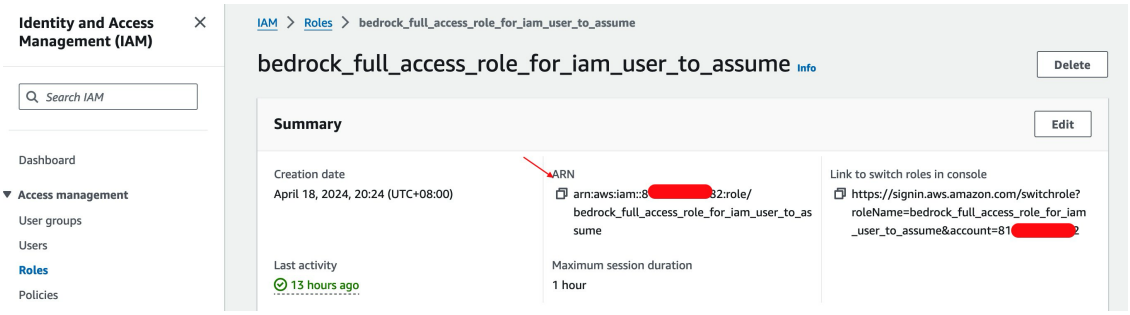


从最佳实践的角度出发，如果只需要角色可以调用模型，请创建如下最小权限的 Policy 并赋予角色：

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Sid": "bedrock",
        "Effect": "Allow",
        "Action": [
            "bedrock:InvokeModel",
            "bedrock:InvokeModelWithResponseStream"
        ],
        "Resource": "arn:aws:bedrock:*::foundation-model/*"
    }
}
```

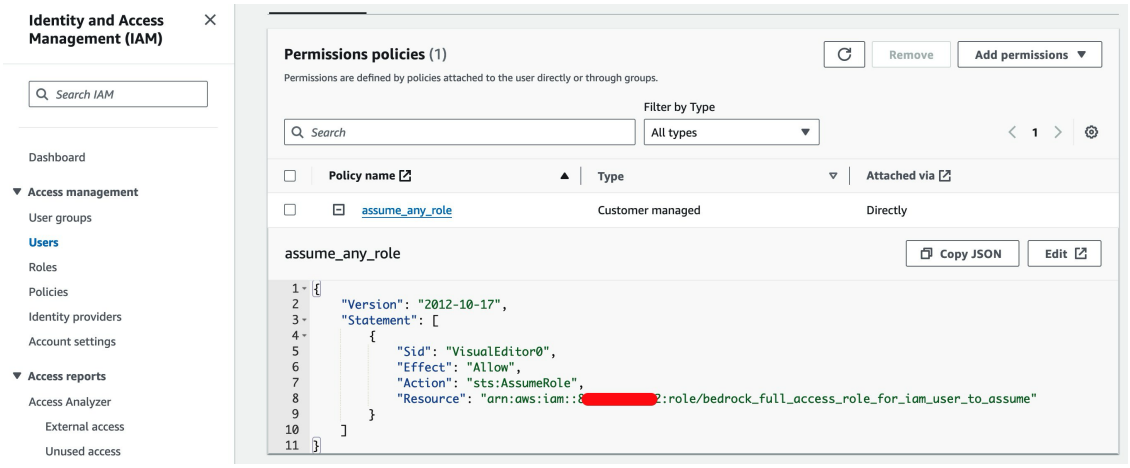
4. 保证该角色的信任对象是本账户的用户：

5. 然后就可以给角色起名字，并完成创建。创建之后角色会有一个 ARN，会在代码的部分需要：

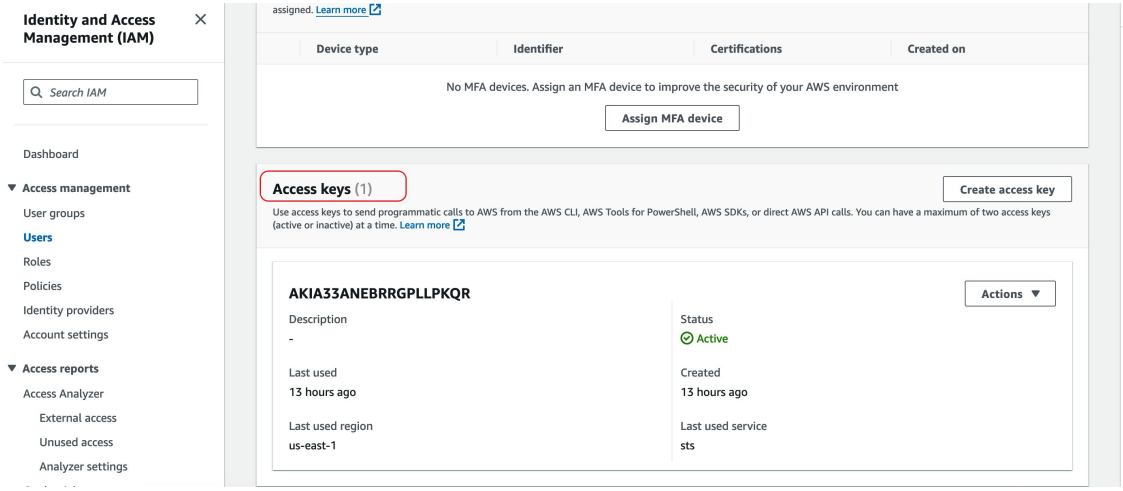


在 **aws** 控制台创建一个 **IAM** 用户

具体如下图，需要用户具备一个基本的能力：扮演上面那个特定的角色，对应的 policy 设置如下：



同时，为该用户创建一个 AKSK，并妥善保存：



配置 **Python** 代码

```
#!/usr/bin/python3
import boto3
import json
#from prompts import PROMPTS

ASSUME_ROLE = True #可以设置 True or False 来决定在脚本内部扮演角色还是用脚本外部的 aws configure 环境
STREAM_ENABLED = False #可以设置 True or False 来决定是否启动流方式输出
REGION = 'us-east-1' #设定要使用的 Region

# Assume role to get keys and token, the aksk is from a user called: bedrock_caller
def assume_role():
    sts_client = boto3.client(
        service_name = 'sts',
        aws_access_key_id='xxxxxxxxxx',#这里的 Key 就是你创建的 IAM 用户的 Key
        aws_secret_access_key='xxxxxxxxxxx',#这里的 Key 就是你创建的 IAM 用户的 secret
        region_name = REGION
    )
    print("Your current caller identity(user/role):")
    current_caller_identity = sts_client.get_caller_identity()
    print("Account:", current_caller_identity['Account'], "Arn:", current_caller_identity['Arn'])

    assumed_role_object = sts_client.assume_role(
        RoleArn="arn:aws:iam::8xxxxxxxxx:role/bedrock_full_access_role_for_iam_user_to_assume",#这里的 ARN 就是你创建的 IAM 角色的 ARN
        RoleSessionName="Bedrock-Test"
    )
    print("Get credentials: ")
    access_key = assumed_role_object['Credentials']['AccessKeyId']
    secret_key = assumed_role_object['Credentials']['SecretAccessKey']
    session_token = assumed_role_object['Credentials']['SessionToken']
    print("AccessKey:", access_key)
    print("SecretKey:", secret_key)
    print("SessionToken:", session_token)
```

```
    return access_key, secret_key, session_token

def main():

    if ASSUME_ROLE:
        access_key, secret_key, session_token = assume_role()
        # Use keys and token to invoke bedrock
        # set to us-east-1 region
        bedrock = boto3.client(
            service_name='bedrock-runtime',
            aws_access_key_id=access_key,
            aws_secret_access_key=secret_key,
            aws_session_token=session_token,
            region_name=REGION
        )
    else:
        bedrock = boto3.client('bedrock-runtime',
region_name=REGION)

    modelId = 'anthropic.claude-3-sonnet-20240229-v1:0' #可以根据
需要设定不同的 modelID，但是注意 claude3 的 opus 目前只在 us-west-2 可用
    accept = 'application/json'
    contentType = 'application/json'

    body = json.dumps({
        "anthropic_version": "bedrock-2023-05-31",
        "max_tokens": 100000,
        "messages":[
            {"role": "user", "content": "What is nvlink and
nvswitch?"},
            #{"role": "user", "content": PROMPTS[7]}
        ]

    })

    if STREAM_ENABLED:
        response = bedrock.invoke_model_with_response_stream(
            body=body,
            modelId=modelId,
            accept=accept,
            contentType=contentType,
        )
        stream = response['body']
        if stream:
            for event in stream:
                chunk = event.get('chunk')
                if chunk:
```

```
                    chunk_obj =
json.loads(chunk.get("bytes").decode())
                    if chunk_obj['type'] ==
'content_block_delta':
                        print(chunk_obj['delta']['text'],
end='')
        print()
    else:
        response = bedrock.invoke_model(
            body=body,
            modelId=modelId,
            accept=accept,
            contentType=contentType
        )
        response_body = json.loads(response.get('body').read())
        for content in response_body['content']:
            print(content['text'])

if __name__ == '__main__':
    main()
```

启动角色扮演的执行过程如下：

```
(python39) donxueg@b0be83710ed1 claude3 % python inference_claude3_with_real_credential.py
Your current caller identity(user/role):
Account: 8[redacted]2 Arn: arn:aws:iam:[redacted]82:user/bedrock_caller
Get credentials:
AccessKey: ASIA33ANEBRRITZXWLNJ
SecretKey: Pt3L5jSaSQEJFeaG5wnrUEvMTaYOgvK9S4mfQGMz
SessionToken: FwoGZXIvYXdzENT//////////wEaDBnSWCuAffWmfAAfAiKwAaIVI8m+uit2iWAkfDv0TMIAPdDT4xYn+oLhLwH5+2CMVdq7Wl81MGAhAYBjeych
FTsWxmCt5jcluAkwHq/h4SqD1PbUMPfL3/gpDerhXKWxHp2ELSRicctz0nW5PggkWU0Nr2wcopMlsmvCg512KEaR6+Lm07oGZz4fizZ2iySKQkYozmcF/C0UlSPCLD
lWYYp+N5VT1QU2oTT7Ury9qb3KgJivniMTx+7m+50gl6avKI2kh7EGMi2E/IcPqj/7CaZ3qPMAybCui9GsPrZdRvlUkXCHBQTVuEEvGg3wgfcfIoBNdvw=
Hello! How can I assist you today?
(python39) donxueg@b0be83710ed1 claude3 %
```

直接使用环境里的 aws configure 的 profile 的执行过程如下：

```
(python39) donxueg@b0be83710ed1 claude3 % aws configure
AWS Access Key ID [****************YKFI]:
AWS Secret Access Key [****************Dbfs]:
Default region name [us-east-1]:
Default output format [None]:
(python39) donxueg@b0be83710ed1 claude3 % python inference_claude3_with_real_credential.py
I am an AI assistant created by Anthropic. I'm an artificial intelligence without a physical body or visual form. I'm designed
 to engage in conversation, answer questions, and help with a variety of tasks to the best of my abilities based on how I've b
een developed. It's nice to meet you! Let me know if you have any other questions.
(python39) donxueg@b0be83710ed1 claude3 %
```