AWS Cookbook

Security

```
Using the PowerUserAccess AWS managed policy for development purposes is a better alternative to start rather than using AdministratorAccess
to create an alert when a root login occurs
```

说明部分

Role

```
1、 创建 Role 时，是需要选择 trusted entity的，所以如下创建Role时都会跟上一个  "Action": "sts:AssumeRole" 的 policy 做为 --assume-role-policy-document 的参数
2、 本书中为了方便测试，都是使用当前 user 的 arn 做为创建 Role 时的 trusted entity（实际给不同的IAM User）
3、 权限方面，Add permissions包含 Permission policies 和 permissions boundary(PP和PB都是非必选)
4、 permissions boundary 定义了一个 Role 可以有的最大权限，会把权限限制在这个范围中，不管Permission policies给的再大
```

1.1、Creating and Assuming an IAM Role for Developer Access

```
# 信任策略 和 权限策略
# 为保证 policy granular粒度的，信任策略和权限策略分开创建
1、先创建信任策略
2、创建role使用这个信任策略
3、给这个 role attach 其他policy
则，信任策略可以复用

# ec2 做为 assume role的trusted entities
1、使用 ec2 做为 sts:assumeRole 的 trusted entities
2、创建 role 使用如上策略
3、再给这个role一个访问 s3 的plicy
使用：ec2 中 Modify IAM role，更换为这个role，这个ec2就可以访问s3了

#IAM user做为 assume role的trusted entities
# 通过如下命令assume到role后，获取临时的credential，然后使用这个role的临时credential访问资源
aws sts assume-role --role-arn $ROLE_ARN --role-session-name AWSCookbook101

当你将 IAM 角色附加到 EC2 实例后，该实例将持续使用这个角色，直到你明确将角色从实例中移除。
这与用户通过 sts:AssumeRole 获取临时凭证不同，EC2 实例在其生命周期内会一直使用分配的角色，除非你手动修改或移除该角色。
```

```
PRINCIPAL_ARN=$(aws sts get-caller-identity --query Arn --output text) -> arn:aws:iam::202329666608:user/devops

# cat assume-rolepolicy.json
# 只有 user/devops 这个用户能够 assume 这个role
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::202329666608:user/devop
s"  # 意思只有这个devops user可以担任这个 role，或者说只允许这个 pri
ncipal assume role
            },
            "Action": "sts:AssumeRole"
        }
    ]
}

# create roles AWSCookbook101Role，这个Role可以允许 PRINCIPAL_A
RN(arn:aws:iam::202329666608:user/devops)去 assume担任 这个 rol
e
ROLE_ARN=$(aws iam create-role --role-name AWSCookbook101Role
--assume-role-policy-document file://assume-rolepolicy.json -
-output text --query Role.Arn)

# 给 AWSCookbook101Role 这个Role attach 上 PowerUserAccess 这个
policy
aws iam attach-role-policy --role-name AWSCookbook101Role --p
olicy-arn arn:aws:iam::aws:policy/PowerUserAccess


# 查看 role 信息
aws iam get-role --role-name AWSCookbook101Role

{
    "Role": {
        "Path": "/",
        "RoleName": "AWSCookbook101Role",
        "RoleId": "AROAS6G6JWQYGWHVENLWI",
        "Arn": "arn:aws:iam::202329666608:role/AWSCookbook101
Role",
        "CreateDate": "2024-07-29T03:58:05+00:00",
```

```
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "AWS": "arn:aws:iam::202329666608:use
r/devops"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "MaxSessionDuration": 3600,
        "RoleLastUsed": {}
    }
}

# 给 Role 设置 --duration-seconds 43200
aws iam update-role --role-name AWSCookbook101Role --max-sess
ion-duration 43200

{
    "Role": {
        "Path": "/",
        "RoleName": "AWSCookbook101Role",
        "RoleId": "AROAS6G6JWQYGWHVENLWI",
        "Arn": "arn:aws:iam::202329666608:role/AWSCookbook101
Role",
        "CreateDate": "2024-07-29T03:58:05+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
```

```
                        "AWS": "arn:aws:iam::202329666608:use
r/devops"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "MaxSessionDuration": 43200,
        "RoleLastUsed": {}
    }
}

# 临时 assume 一下这个 Role
# 默认不加 --duration-seconds是一个小时，最短 900 seconds (15 min
utes)，最长  can be up to 12 hours (43200 seconds)
# 设置 --duration-seconds 43200 需要这个 Role支持
aws sts assume-role --role-arn $ROLE_ARN --role-session-name
AWSCookbook101 --duration-seconds 43200

{
    "Credentials": {
        "AccessKeyId": "ASIAS6G6JWQYAXJ7TQVW",
        "SecretAccessKey": "z932NJMRrTzXSdyFGfgCkeunmUSATeEZV
XcUM59J",
        "SessionToken": "IQoJb3JpZ2luX2VjED4aCXVzLWVhc3QtMSJH
MEUCIFXbjP6az0PoZO4+Susjm3vQPVWJqm2sFKdolfErk0YBAiEA21LNaR5uh
mhnSPxQ4MhplO80jhmctNRPNMT/c63eXo4qmwIIJxAAGgwyMDIzMjk2NjY2MD
giDC48qBDZ0mhMTF4FTyr4Aen/TINTL9f4gmsKO2uPQX2DqH1xdMnQv/YXjOx
6GwSse2colCv3GKVozIyG52ino9sJsCEnJ/zkSbCdNXNbU75Ci2Ew6SsA5WZF
DRaia8NupkjRx6ZrzfVuRxhXYLZHRBPYKBPIevWNFKTIxgKnEndDb3f4jETs2
XaNz51J8PsX8RLu2rwdP7cM0kRBvAz+uwnf3W1gAqM0db2PXZir6wOpnWquf+
S0QC+SOQAlLTkRWLs4AYhT5PzTETSkN5Zd02tpEjjBUckik9bWhKVKjcDbMM
9/XxRD+uQmwcVAul353z4RacoHiPHKe6yd2lVYqVvANJJpc3RPMLjanLUGOp0
BajUwxonIHppJX5h6rcu3ngYC3AQJUvxtOC2kNkeIIsPKOZM6KyrPIJNiQmBq
utZOHZAKgAmLCMIlZsmJvzGovrT8q5LLj762+C9J7DoZnCaX+b75/NM+86xRy
hIG5O7p2pv59nJOXibm1SN7cnUyAkPeWVoh4LPuQ4aO7IUDctBRg6PgRWDbOp
```

```
yNDFEJURI+UYS0uUjgWyFb8Cun+w==",
        "Expiration": "2024-07-29T06:48:40+00:00"
    },
    "AssumedRoleUser": {
        "AssumedRoleId": "AROAS6G6JWQYGWHVENLWI:AWSCookbook10
1",
        "Arn": "arn:aws:sts::202329666608:assumed-role/AWSCoo
kbook101Role/AWSCookbook101"
    }
}

#方便获取
creds=$(aws --output text sts assume-role --role-arn $ROLE_AR
N --role-session-name AWSCookbook101 | \
 grep CREDENTIALS | cut -d " " -f2,4,5)
export AWS_ACCESS_KEY_ID=$(echo $creds | cut -d " " -f2)
export AWS_SECRET_ACCESS_KEY=$(echo $creds | cut -d " " -f4)
export AWS_SESSION_TOKEN=$(echo $creds | cut -d " " -f5)

# Using Temporary Credentials
export AWS_ACCESS_KEY_ID=ASIA...EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEX
AMPLEKEY
export AWS_SESSION_TOKEN=FwoGZXIvYXdzEGoaDJ...EXAMPLE
```

1.2 Generating a Least Privilege IAM Policy Based on Access Patterns

1.3 Enforcing IAM User Password Policies in Your AWS Account

```
aws iam update-account-password-policy \
 --minimum-password-length 32 \
 --require-symbols \
 --require-numbers \
 --require-uppercase-characters \
 --require-lowercase-characters \
 --allow-users-to-change-password \
 --max-password-age 90 \
 --password-reuse-prevention 2 # 重复使用超过2次则阻止使用


# delete
aws iam delete-account-password-policy
```

Group

```
# list policies
aws iam list-policies --scope AWS

# create group、attach policy、add User、remove User、delete g
roup
aws iam create-group --group-name AWSCookbook103Group
aws iam attach-group-policy --group-name AWSCookbook103Group
--policy-arn arn:aws:iam::aws:policy/AWSBillingReadOnlyAccess
aws iam add-user-to-group --user-name devops --group-name AWS
Cookbook103Group
# 删除 group 前，需要删删除 user 和 policy
aws iam remove-user-from-group --user-name devops --group-nam
e AWSCookbook103Group
aws iam detach-group-policy --group-name AWSCookbook103Group
--policy-arn arn:aws:iam::aws:policy/AWSBillingReadOnlyAccess
aws iam delete-group --group-name AWSCookbook103Group
```

1.4 Testing IAM Policies with the IAM Policy Simulator

1、创建一个policy，允许 ec2 assume role

```
# 让 ec2 可以担任一个 Role、 只允许 EC2 服务担任角色。使用这个policy
创建的role只能作用在ec2上
# assume-role-policy.json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
               "Service": [
                    "ec2.amazonaws.com",
                    "lambda.amazonaws.com",
                    "ecs-tasks.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}

# 创建一个Iam role使用上面的策略
aws iam create-role --assume-role-policy-document file://assu
me-role-policy.json --role-name AWSCookbook104IamRole

# 给这个 role attach一个EC2的只读权限
# 这个role 只能对 ec2 只读访问
aws iam attach-role-policy --role-name AWSCookbook104IamRole
--policy-arn arn:aws:iam::aws:policy/AmazonEC2ReadOnlyAccess
```

2、Simulate the effect of the IAM policy you are using, testing several different types of actions on the EC2 service.

```
# 模拟这个 role 去创建 ec2的 internet gateway
aws iam simulate-principal-policy --policy-source-arn arn:aw
s:iam::202329666608:role/AWSCookbook104IamRole --action-names
ec2:CreateInternetGateway

{
    "EvaluationResults": [
        {
            "EvalActionName": "ec2:CreateInternetGateway",
            "EvalResourceName": "*",
            "EvalDecision": "implicitDeny",
            "MatchedStatements": [],
            "MissingContextValues": []
        }
    ]
}


# 上面是只读权限，那么验证一下 describe权限
aws iam simulate-principal-policy --policy-source-arn arn:aw
s:iam::202329666608:role/AWSCookbook104IamRole --action-names
ec2:DescribeInstances

{
    "EvaluationResults": [
        {
            "EvalActionName": "ec2:DescribeInstances",
            "EvalResourceName": "*",
            "EvalDecision": "allowed",
            "MatchedStatements": [
                {
```

```
                    "SourcePolicyId": "AmazonEC2ReadOnlyAcces
s",
                    "SourcePolicyType": "IAM Policy",
                    "StartPosition": {
                        "Line": 3,
                        "Column": 17
                    },
                    "EndPosition": {
                        "Line": 8,
                        "Column": 6
                    }
                }
            ],
            "MissingContextValues": []
        }
    ]
}
```

1.5 Delegating IAM Administrative Capabilities Using Permissions Boundaries

授权团队成员发布lambda functions和创建Iam roles的能力

需要限制Iam roles对实际权限的创建，只允许lambda functions仅需要的actions

场景：

- lambda developers 创建 lambda functions，需要连接 s3，DynamoDB，CloudWatch Logs（创建不同的组收集不同的lambda functions）等
- 每个lambda functions连接的不同的s3的桶、不同的dynamoDB的table、会被不同的CloudWatch的Log group（需要创建Log groups和Log stream）
- 用户 assume 这个Role后可以创建Role，创建的这个Role中可以定义具体的policy（某个lambda连接某个s3，某个dynamodb的table，这个创建就比较具体了）

分析：
1、创建一个可以给Iam User去 assume 的 Iam Role 给develper team members

2、给这个 Role 创建 permissions boundary 去定义其最大的权限(logs的logs:CreateLogGroup,logs:CreatelogStream,logs:PutLogEvents、DynamoDB的dynamodb:Putitem,dynamodb:UpdateItem,dynamodb:DeleteItem限制在table/AWSCookbook*下、 S3的s3:GetObject,s3:PutObject限制在AWSCookbook*/*的桶中)

3、定义 Role 的 policy
    3.1、deny 掉删除 RolePermissionsBoundary的权限。（限制它不能把上面定义的PB给删除了）
    3.2、确保 Role 有对 IAM console的读权限，能看到账号以及看到自己创建的IAM Role
    3.3、（policy相关的操作）让 Role 在policy方面可以 iam:CreatePoliy,iam:DeletePoliy,iam:CreatePoliyVersion,iam:DeletePoliyVersion,iam:SetDefaultPolicyVersion，限制在 arn:aws:iam::AWS_ACCOUNT_ID:policy/AWSCookbook* 下，意味着只能AWSCookbook*这样的命名
    3.4、（role相关的操作）让 Role 可以创建Role并操作policy给这个role的相关权限。如：iam:CreateRole等，限制其命名：arn:aws:iam::AWS_ACCOUNT_ID:role/AWSCookbook* 只能叫AWSCookbook*，并且这个Role必须包含 "iam:PermissionsBoundary":"arn:aws:iam::AWS_ACCOUNT_ID:policy/AWSCookbook105PB"（这个是第二步中的PB的policy名称，意思只能操作含有这个policy的Role，不能去操作其他的Role,这步是Role相关的操作，所以要限制对哪些role进行操作）
    3.5、开发对lambda、logs、dynamodb、s3的全部权限，因为PB有最大限制，所以这里只有给它全部的权限（更加的方便）

```
        {
            "Sid": "ServerlessFullAccess",
            "Effect": "Allow",
            "Action": [
                "lambda:*",
                "logs:*",
```

```
                "dynamodb:*",
                "s3:*"
            ],
            "Resource": "*"
        },
```

3.6、给这个 Role 可以去 iam:PassRole的权限（就是给其他user或者serivce等添加role，这个地方目前只涉及到lambda服务），并且限制只能对lambda服务，pass的这个role也限制在arn:aws:iam::AWS_ACCOUNT_ID:role/AWSCookbook*下

3.7、本次实例中，因为在创建这个Role时创建的两个policy（第二步，第三步中创建的policy，因为命名规则和第三步给的命名限制是重叠的，所以需要把这两个policy的一些删除修改等权限Deny掉）所以命名尽量不要命名重叠

第二步创建的policy叫AWSCookbook105PB、第三步创建的policy叫AWSCookbook105Policy，所以这两个Policy的删除更新deny掉

```
        {
            "Sid": "ProtectPB",
            "Effect": "Deny",
            "Action": [
                "iam:CreatePolicyVersion",
                "iam:DeletePolicy",
                "iam:DeletePolicyVersion",
                "iam:SetDefaultPolicyVersion"
            ],
            "Resource": [
                "arn:aws:iam::AWS_ACCOUNT_ID:policy/AWSCookbook105PB",
                "arn:aws:iam::AWS_ACCOUNT_ID:policy/AWSCookbook105Policy"
            ]
        }
```

1.6、Connecting to EC2 Instances Using AWS SSM Session Manager

即使 EC2 实例有公共 IP 地址或弹性 IP 地址，它们也需要通过 Internet 网关才能真正访问公共互联网

```
# Connect to an Amazon EC2 instance by using Session Manage
r
https://docs.aws.amazon.com/prescriptive-guidance/latest/pa
tterns/connect-to-an-amazon-ec2-instance-by-using-session-m
anager.html

# Improve the security of EC2 instances by using VPC endpoi
nts for Systems Manager
https://docs.aws.amazon.com/systems-manager/latest/userguid
e/setup-create-vpc.html

# 大致流程，如果是在private subnet的情况下
1、创建 role，policy为 AmazonSSMManagedInstanceCore、Trusted
entity为ec2
2、创建 ec2 使用如上 role，AMI要选带有 ssm agent的镜像，或者自己安
装agent
3、在 ec2的connect，或者session manager 里 start session
4、如果vpc有internat gateway、并且ec2有外网地址，则可以像如上直接
连。
        如果vpc没有 IG、ec2也没有外网地址（有外网地址没有IG也不行）
则需要使用 vpc endpoint来完成和system manager服务打通
        vpc endpoint中的 security group需要放行 https也就是443
端口
```

1.7 Encrypting EBS Volumes Using KMS Keys

KMS分为：

AWS managed keys 和 Customer managed keys

区别：一个是由AWS管理，例如你在加密volume和S3时，选择由AWS管理kms的话，aws会自动创建kms、要么就是选择自己创建的KMS。不管是哪种方式，AWS 不对 AWS 管理的 KMS 密钥本身收取费用，但它们用于加密和解密操作时会产生费用

举例S3

1、一个S3 bucket，使用kms加密，aws-test帐号有read权限，但是没有kms权限、devops账号有kms和s3的full权限
2、此时上传文件到S3、aws-test账号不能下载，报错如下。devops账号可以随便访问下载
3、当给 aws-test账号kms权限后，就可以下载了
说明，就算有S3的访问权限，如果S3桶加密的话，没有KMS权限也一样不能访问

```
<Error>
<script/>
<Code>AccessDenied</Code>
<Message>User: arn:aws:iam::202329666608:user/aws-test is not authorized to perform: kms:Decrypt on resource: arn:aws:kms:us-east-1:202329666608:key/69079cde-85b0-49ff-83d0-66166f2d3407 because no identity-based policy allows the kms:Decrypt action</Message>
<RequestId>JN9P99KMJRCHP45B</RequestId>
<HostId>HX0fOYUatLyAtgYRJ0qdB+omnfPgcn+AnwGCnClPpSgUI4RfNkDHREZ9JDhhb3NmL0kFnTg/pP4=</HostId>
</Error>
```

EC2

如果攻击者能够获得对 EC2 实例的控制权，那么他们将能够访问并读取挂载在该实例上的加密 EBS 卷的数据。这是因为在 EC2 实例启动并成功挂载 EBS 卷后，解密密钥已经被加载到实例中，并且操作系统可以透明地访问加密卷上的数据

实际作用：

- 数据保护：加密的数据卷可以防止物理或逻辑攻击，即使攻击者获得了卷的副本，也无法解密数据。
- 合规性：加密 EBS 卷可以帮助您满足各种数据保护和合规性要求。

1.8 Storing, Encrypting, and Accessing Passwords Using Secrets Manager

1、创建policy，对这个 SECRET_ARN 的读权限之类的、对ListSecrets放行所有
2、给你为 ec2 创建的role上attach上这个policy

1.9 Blocking Public Access for an S3 Bucket、 Analyzer

通过 Access Analyzer 服务扫描 S3

```
# 创建一个bucket、设置如下policy可以被publicly available
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::rick-website/*"
    }
  ]
}
# 创建扫描
```

```
ANALYZER_ARN=$(aws accessanalyzer create-analyzer \
 --analyzer-name awscookbook109\
 --type ACCOUNT \
 --output text --query arn)

# 扫描
aws accessanalyzer start-resource-scan --analyzer-arn $ANALYZ
ER_ARN --resource-arn arn:aws:s3:::rick-website

#查看扫描结果
aws accessanalyzer get-analyzed-resource \
 --analyzer-arn $ANALYZER_ARN \
 --resource-arn arn:aws:s3:::rick-website

{
    "resource": {
        "resourceArn": "arn:aws:s3:::rick-website",
        "resourceType": "AWS::S3::Bucket",
        "createdAt": "2024-08-01T03:17:21.407000+00:00",
        "analyzedAt": "2024-08-01T03:18:59.517000+00:00",
        "updatedAt": "2024-08-01T03:17:21.407000+00:00",
        "isPublic": true,
        "actions": [
            "s3:GetObject"
        ],
        "sharedVia": [
            "POLICY"
        ],
        "status": "ACTIVE",
        "resourceOwnerAccount": "202329666608"
    }
}
```

1.10 Serving Web Content Securely from S3 with CloudFront

```
1、创建一个 CloudFront Distribution、关联到S3 bucket
2、给 S3 attach一个policy，只允许这个 CloudFront Distribution访问

{
    "Version": "2012-10-17",
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::cloudfront:user/CloudFront O
rigin Access Identity CLOUDFRONT_OAI "
        },
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::S3_BUCKET_NAME/*"
    }]
}

3、获取 CloudFront Distribution 的DOMAIN Name
DOMAIN_NAME=$(aws cloudfront get-distribution --id $DISTRIBUT
ION_ID --query Distribution.DomainName --output text)

4、访问
# 直接访问S3是访问不到的
curl https://awscookbook110-$RANDOM_STRING.s3.$AWS_REGION.ama
zonaws.com/index.html
# 访问上面的domain name
curl $DOMAIN_NAME
```

2.3 Internet Gateway

```
Internet Gateway 是 VPC 与互联网之间的桥梁，它提供了必要的通道以确保双向通信
如果 VPC 中没有 Internet Gateway (IGW)，那么这个 VPC 就无法与互联网直接通信
```

2.4 Using a NAT Gateway for Outbound Internet Access from Private Subnets

公共子网：一个子网被称为公共子网是因为它的路由表中有指向 Internet Gateway 的 0.0.0.0/0 路由、这样的子网允许其内部资源直接访问互联网，并且也可以被互联网访问

The EIP associated with your NAT gateway becomes the external IP address for all communication that goes through it.

EIP 地址就是私网的出口地址，也就是访问其他互联网服务的“源地址”

```
# Internet Gateway  vs NAT Gateway
Internet Gateway 是双向通信、入站和出站
NAT Gateway是单向、允许私有子网中的资源发起与互联网的出站连接、阻止入站流量

1、创建一个NAT gateway绑定一个eip、选择一个subnet、则这个NG就处在这个subnet中（这个subnet子网是一个public subnet，这个public subnet子网中有一个route table是0.0.0.0的指向internet gateway、也就是说NAT Gateway要在一个有internet gateway路由的子网中）
2、其他private subnet中，创建一个route table，设置去往0.0.0.0的流量目标地址为 NAT Gateway

1.  私有子网实例 → 私有子网路由表 → NAT Gateway（NAT Gateway 将私有子网实例的私有 IP 地址转换为 NAT Gateway 的弹性 IP 地址（EIP）） → 公共子网路由表 → Internet Gateway → 互联网
2.  互联网响应 → Internet Gateway → 公共子网路由表 → NAT Gateway
```

（NAT Gateway 根据连接跟踪表，将响应流量转发回发起请求的私有子网实例）
→ 私有子网实例

2.5 Security Group

1、SG 绑定在 ENI 上、是用来给EC2做流量控制、Network ACLs是给 subnet 做流量控制 controlling traffic
2、一个 ENI 只能在一个 Subnet 中、一个Subnet在一个可用区中
3、SG 在 attached到ENI和被其他SG引用时不能删除。They are referenced by other security groups

2.6 Using VPC Reachability Analyzer to Verify and Troubleshoot Network Paths

2.8 Simplifying Management of CIDRs in Security Groups with Prefix Lists

Prefix List 是一种允许您创建和管理 IP 前缀列表的功能。Prefix List 在管理网络流量和安全策略时非常有用，特别是在配置安全组和网络 ACL 规则时
简单说就是创建一个List，这个List的ip可以被用在多个安全组规则中、统一管理、减少规则数量
比如下面的 Endpoint，S3选择使用Gatewy Endpoint的情况下，当你选择了route后，会自动给你的route table中添加如下route
Destination Target
pl-63a5400a vpce-08e08a3b05c7b56e8

```
pl-63a5400a 就是一个prefix list由aws managed，里面包含了s3所有的CIDR
```

2.9 Controlling Network Access to S3 from Your VPC Using VPC Endpoints

使用Endpoint访问服务时，你的vpc可以没有internet gateway

All traffic bound for these services will be directed by the route table to the VPC endpoint rather than the public internet route

所有前往这些服务的流量讲被route table定向到vpc到endpoint、而不是通过公共互联网路由

```
对于 endpoint 类似分为 gateway和interface：
Gateway Endpont：
    1、只能给S3、DynamoDB使用
    2、架构是基于VPC route table的
    3、创建好之后产生流量收费
    4、多个服务也只会在你选择的 route table中添加路由。

Interface Endpoint：
    1、可以给包含ENdpoint支持的所有服务
    2、是基于security group的，粒度更小的控制流量（创建时需要选择vpc、route table、 security group）
    3、除了流量还有每小时的费用
    4、在每个subnet中创建ENI来和Endpoint通信、多个服务则会创建多个ENI。这样成本会很高
```

Modify the bucket policy for the S3 bucket to allow access only from the VPC endpoint that you created

改变s3的bucket policy，只让你创建的vpc endpoint允许访问

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": [
                                    # 建议两行都写
                "arn:aws:s3:::your-bucket-name",  # 这行表示存储桶本身
                "arn:aws:s3:::your-bucket-name/*" # 这行表示存储桶中的所有对象
            ],
            "Condition": {
                "StringEquals": {
                    "aws:SourceVpce": "vpce-0123456789abcdef0"
                }
            }
        }
    ]
}
```

2.10 Enabling Transitive Cross-VPC Connections Using Transit Gateway

AWS Transit Gateway 是一种用于简化和集中管理跨多个 VPC 和本地网络连接的网络服务。它充当网络流量的中央枢纽，使得在不同网络之间进行安全且高效的通信变得更容易。通过 Transit Gateway，您可以实现多种网络连接的集中管理，从而简化网络架构，降低管理复杂性，并提高网络性能和安全性。

功能和用途

1. 集中管理网络连接

- 多 VPC 连接：您可以将多个 VPC 连接到一个 Transit Gateway，实现集中的网络管理和流量控制。
- 跨区域连接：Transit Gateway 支持跨 AWS 区域的连接，使得全球范围内的网络架构更加连通。
- 跨账户连接：通过 AWS Resource Access Manager (RAM)，您可以在多个 AWS 账户之间共享 Transit Gateway，简化跨账户网络管理。

2. 集成本地网络

- IPsec VPN：您可以通过安全的 IPsec VPN 将本地数据中心或办公室连接到 AWS 网络。
- Direct Connect (DX)：Direct Connect 提供高带宽和低延迟的专用网络连接，将本地网络直接连接到 AWS。
- 第三方网络设备：支持将第三方网络设备连接到 Transit Gateway，实现更多网络功能和集成。

3. 动态路由更新

- BGP 支持：Transit Gateway 支持 Border Gateway Protocol (BGP)，可以通过动态路由更新在不同网络之间实现更高效的路由管理。
- 路由控制：通过路由表，您可以灵活地控制和定义流量的路由路径，以优化网络性能和安全性。

4. 高可用性和弹性

- 多 AZ 部署：在每个可用区 (AZ) 使用子网进行部署，以提高网络连接的弹性和高可用性。
- 自动扩展：Transit Gateway 可以根据流量需求自动扩展，确保高性能和低延迟。

5. 成本优化

- 减少对等连接：通过 Transit Gateway，可以减少大量的 VPC 对等连接，简化网络架构，降低成本。
- 共享资源：多个 VPC 可以共享 NAT 网关和 Internet 网关，从而减少资源浪费和成本。

使用场景

1. 跨多个 VPC 的应用程序部署

- 当您在多个 VPC 中部署应用程序时，Transit Gateway 提供了一个集中管理的解决方案，简化网络配置和流量管理。

2. 混合云架构

- 通过将本地网络和 AWS 网络连接到 Transit Gateway，您可以构建混合云架构，实现本地资源和云资源的无缝集成。

3. 多区域、多账户网络管理

- 在多区域和多账户环境中，Transit Gateway 提供了一种高效的网络管理方式，确保网络的连通性和安全性。

4. 灾难恢复和业务连续性

- 通过跨区域和跨账户的网络连接，您可以构建高可用的灾难恢复方案，确保业务连续性。

小结

AWS Transit Gateway 是一个功能强大的网络服务，提供集中管理、多种连接选项、高可用性和弹性、动态路由更新以及成本优化等优势，使得跨多个 VPC、本地网络和云服务的网络架构变得更加高效和安全。

配置 AWS Transit Gateway 路由规则，允许跨 VPC 的流量通过 Transit Gateway 进行路由，您需要在每个 VPC 的路由表中添加相应的路由条目。以下是详细步骤：

步骤 1：创建并附加 Transit Gateway

1. 创建 Transit Gateway（如果尚未创建）：

- 在 AWS 管理控制台，导航到 VPC 服务，然后选择 “Transit Gateway”。
- 点击 “Create Transit Gateway”，填写必要的信息，然后创建。

2. 附加 VPC 到 Transit Gateway：

- 选择创建的 Transit Gateway，然后点击 “Create VPC Attachment”。
- 选择您要附加的 VPC 及其相应的子网，然后完成附加过程。

步骤 2：配置 VPC 路由表

1. 导航到路由表：

- 在 AWS 管理控制台，导航到 VPC 服务，然后选择 “Route Tables”。

2. 选择路由表：

- 选择要配置的 VPC 的路由表。这通常是与您的 VPC 关联的主路由表，或者是您为特定子网配置的自定义路由表。

3. 添加路由条目：

- 在路由表的详细信息页面，点击 “Routes” 标签，然后点击 “Edit routes”。
- 点击 “Add route”。

步骤 3：配置路由规则

- Destination：填写目标 VPC 的 CIDR 范围。这个范围是您希望通过 Transit Gateway 访问的 VPC 的 IP 地址块。例如，如果目标 VPC 的 CIDR 范围是 10.1.0.0/16，那么您将输入 10.1.0.0/16 作为目标。
- Target：选择 “Transit Gateway” 并选择您创建的 Transit Gateway ID。

示例

假设有两个 VPC：

- VPC A：CIDR 范围为 10.0.0.0/16
- VPC B：CIDR 范围为 10.1.0.0/16

您希望通过 Transit Gateway 实现 VPC A 和 VPC B 之间的通信。

1. 在 VPC A 的路由表中：
- Destination：10.1.0.0/16
- Target：选择您创建的 Transit Gateway
2. 在 VPC B 的路由表中：
- Destination：10.0.0.0/16
- Target：选择您创建的 Transit Gateway

确认配置

1. 保存路由条目：
• 完成路由条目的添加后，点击 “Save routes” 以保存配置。
2. 测试连接：
• 通过在 VPC A 中的一个实例 ping VPC B 中的一个实例的 IP 地址，测试跨 VPC 的连接。

小结

通过在每个 VPC 的路由表中添加相应的路由条目，指向 Transit Gateway，您可以实现跨 VPC 的流量通过 Transit Gateway 进行路由。这不仅简化了网络架构，还提供了集中化的管理和控制。

2.11 Peering Two VPCs Together for Inter-VPC Network Communication

选择使用 VPC Peering Connection 还是 AWS Transit Gateway

选择使用 VPC Peering Connection 还是 AWS Transit Gateway 取决于您的具体需求和网络架构的复杂性。以下是两者的主要区别和使用场景，以帮助您决定何时使用哪种解决方案：

VPC Peering Connection

VPC Peering Connection 是一种简单的方式，用于在两个 VPC 之间创建私有连接，使得它们可以通过私有 IP 地址直接通信。
使用场景
1. 小规模网络架构：
• 适用于相对简单和小规模的网络架构，比如只有几个 VPC 需要互相通信的情况。
2. 点对点连接：
• 当只有两个 VPC 需要互相通信时，VPC Peering 是一个简单且成本较低的选择。
3. 特定用途的直接连接：

- 如果您只需要在两个 VPC 之间共享资源或服务，而不需要复杂的路由和管理。

优点

- 简单直接：配置和管理相对简单，适合小型和点对点连接。
- 低成本：对于少量 VPC 之间的连接，成本相对较低。

缺点

- 可扩展性有限：对于多个 VPC 需要互相通信的情况，配置和管理变得复杂。
- 无中心控制：缺乏集中化管理和路由控制。

AWS Transit Gateway

AWS Transit Gateway 是一个中心枢纽，用于连接多个 VPC、本地网络和其他网络服务，提供集中化管理和高效的网络连接。

使用场景

1. 大规模网络架构：
   - 适用于需要连接多个 VPC 和本地网络的大规模网络架构，提供更高的可扩展性和管理效率。
2. 多区域和多账户连接：
   - 需要在多个区域和多个账户之间实现网络连接和共享的情况。
3. 复杂网络需求：
   - 需要复杂的路由控制、动态路由更新（例如通过 BGP）、以及与第三方网络设备集成的情况。

优点

- 高可扩展性：适用于大规模和复杂网络架构，支持多个 VPC 和本地网络的连接。
- 集中管理：提供集中化的路由控制和管理，简化网络配置和管理。
- 跨区域和跨账户支持：支持在多个区域和多个账户之间实现连接和共享。

缺点

- 成本较高：相比 VPC Peering，成本较高，适用于需要更多功能和扩展性的场景。
- 配置复杂：配置和管理相对复杂，适合需要复杂网络架构的企业。

选择指南

- 使用 VPC Peering Connection：

- 如果只有少量 VPC 需要互相通信，且网络架构相对简单。
- 需要点对点的直接连接，不需要复杂的路由和管理。

- 使用 AWS Transit Gateway：
  - 需要连接多个 VPC 和本地网络，并希望集中管理和控制。
  - 需要跨区域或跨账户的网络连接和共享。
  - 网络架构复杂，需要高可扩展性和灵活的路由控制。

小结

- VPC Peering Connection 适用于简单的点对点连接和小规模网络架构。
- AWS Transit Gateway 适用于需要高可扩展性、集中管理和复杂网络需求的大规模网络架构。