Arquitetura AWS - Soluções para problemas simulados

Data: 25/04/25 | Por: Thaise Oliveira - 2code4coffee

- ## Objetivo

  Este artigo foi formulado com base em questões do simulado Solutions Architect Associate da Escola da Nuvem, com finalidade educativa e portfólio, não estando afiliado, associado, autorizado, endossado ou de qualquer forma oficialmente conectado a organizações, empresas ou pessoa juridica. A responsabilidade de uso de qualquer solução ou código deste documento em problemas reais é inteiramente do usuário.

- ## Índice

- ## 1. Proteção de Dados no S3

Caso de uso

Proteger objetos no Amazon S3 contra exclusão acidental em conformidade com regulamentações.

Solução Proposta

✔ Habilitar controle de versão

```
#Habilitando controle de versão
#bash
aws s3api put-bucket-versioning --bucket meu-bucket --versioning-configuration Status=Enabled
```

✔ Ativar MFA Delete

```
#Ativando MFA delete
#bash
aws s3api put-bucket-versioning --bucket meu-bucket \
    --versioning-configuration Status=Enabled,MFADelete=Enabled \
    --mfa "arn:aws:iam::123456789012:mfa/usuario 654321"
```

Fluxo de Recuperação

```
flowchart LR;
    A[Exclusão Acidental]-->B[Restaurar Versão Anterior];
    B-->C[Confirmar MFA];
```

- ## 2. Autenticação em API Gateway

Caso de Uso

Gerenciamento centralizado de usuários para APIs.

Solução Proposta

✔ Amazon Cognito User Pools

Com Grupos de usuários (ex.: admin, usuário)

Integração nativa via JSON Web Tokens (JWT):

```
#Autorizador API utilizando o serviço AWS Cognito User Pools
#yaml
x-amazon-apigateway-authorizer:
  type: cognito_user_pools
  providerARNs:
    - arn:aws:cognito-idp:us-east-1:123456789012:userpool/us-east-1_abc123
```

- ## 3. Migração de Dados em Larga Escala

Caso de Uso

Transferir 5 PB de dados para armazenamento durável.

Solução Proposta

✔ Dispositivos Snowball Edge (80 TB cada)

✔ Política de Ciclo de Vida para Glacier Deep Archive

✔ Automatização de verificação da transferência

Passo a Passo:

```
1. Priorizar migração via Snowball.

2. Implementar políticas de ciclo de vida.

3.  Automatizar verificação da transferência
```

1. Criar job de importação no AWS Snowball

```
#Migração para Snowball
#bash
aws snowball create-job \
  --job-type IMPORT \
  --resources "S3Bucket=meu-bucket-destino" \
  --description "Migração 5PB dados mídia" \
  --kms-key-arn arn:aws:kms:us-east-1:123456789012:key/abcd1234 \
  --storage-optimized \
  --shipping-details file://shipping.json
```

1.2 Verificar status do job

```
#Verificaçao da migração
#bash
aws snowball describe-job --job-id JOBID123-4567-890
```

2. Implementar Políticas de Ciclo de Vida

```
#Configuração de ciclo de vida
#xml
<LifecycleConfiguration>
    <Transition>
        <Days>1</Days>
        <StorageClass>DEEP_ARCHIVE</StorageClass>
    </Transition>
</LifecycleConfiguration>
```

```
#Automatizar transição para Amazon Glacier Deep Archive após validação
#xml
<!-- lifecycle-policy.xml -->
<LifecycleConfiguration>
  <Rule>
    <ID>Transicao-Glacier</ID>
    <Status>Enabled</Status>
    <Filter>
      <Prefix>dados-midia/</Prefix>
    </Filter>
    <Transition>
      <Days>7</Days>  <!-- Período de validação -->
      <StorageClass>DEEP_ARCHIVE</StorageClass>
    </Transition>
    <AbortIncompleteMultipartUpload>
      <DaysAfterInitiation>30</DaysAfterInitiation>
    </AbortIncompleteMultipartUpload>
  </Rule>
</LifecycleConfiguration>
```

```
#Aplicar política via AWS CLI
#bash
aws s3api put-bucket-lifecycle-configuration \
  --bucket meu-bucket-destino \
  --lifecycle-configuration file://lifecycle-policy.xml
```

Dados chegam no S3 via Snowball

3. Automatizar verificação da transferência

```
#python
#Script de verificação automatizado
import boto3

s3 = boto3.client('s3')

def verify_snowball_transfer(bucket):
    response = s3.list_objects_v2(Bucket=bucket)
    if response['KeyCount'] == 0:
        raise Exception("Bucket vazio - verifique transferência")
    else:
        print(f"Transferência validada: {response['KeyCount']} objetos")
```

Vantagens:

✔ Redução de 90% no custo de transferência

✔ Tempo otimizado: semanas vs. meses

---

4. Monitoramento Automatizado de EC2

Caso de Uso

Instâncias congelando sem alertas.

Solução Proposta

✔ **Alarme CloudWatch com ação de reinicialização**

```
#Alarme no CloudWatch para reiniciar EC2 em caso de falhas
#bash
aws cloudwatch put-metric-alarm \
    --alarm-name "EC2-Auto-Reboot" \
    --metric-name StatusCheckFailed \
    --threshold 1 \
    --alarm-actions "arn:aws:swf:us-east-1:123456789012:action/actions/AWS_EC2.InstanceId.Reboot/1.0"
```

Vantagens:

✔Usa uma combinação de operadores de comparação e limites

✔Quando a verificação de status falha, o alarme é ativado

✔Uma ação será executada quando o status check falhar: reiniciar automaticamente a instância

---

- ## 5. Segurança em Bancos de Dados RDS

Caso de Uso

Estabelecer e verificar uma conexão segura SSL com um banco de dados PostgreSQL na AWS RDS.

Solução Proposta

✔ Habilitar SSL/TLS para PostgreSQL

✔ Validar Certificados SSL no RDS

```
#Estabelecer uma conexão segura
#bash
psql "host=meudb.rds.amazonaws.com sslmode=verify-full sslrootcert=rds-ca-2019-root.pem"
```

```
#Verificar status da conexão SSL da sessão de banco de dados atual
#SQL
    SELECT * FROM pg_stat_ssl WHERE pid = pg_backend_pid();
```

```
#bash
# Verificar status SSL no RDS PostgreSQL
aws rds describe-db-instances \
  --db-instance-identifier meu-instancia-rds \
  --query 'DBInstances[0].CertificateDetails'

# Testar conexão SSL
psql "host=meudb.rds.amazonaws.com dbname=mydb \
  user=myuser password=mypassword \
  sslmode=verify-full sslrootcert=/path/to/rds-ca-2019-root.pem"

# Script de monitoramento contínuo:
#!/bin/bash
openssl s_client -connect meudb.rds.amazonaws.com:5432 \
  -starttls postgres -CAfile rds-ca-2019-root.pem 2>/dev/null | \
  openssl x509 -noout -dates
```

```
#Alarme no CloudWatch para monitorar a expiração do certificado SSL da instância RDS (30 dias)
#bash
aws cloudwatch put-metric-alarm \
  --alarm-name "RDS-SSL-Expiration" \
  --metric-name "CertificateValidityPeriod" \
  --namespace "AWS/RDS" \
  --dimensions "Name=DBInstanceIdentifier,Value=meu-instancia-rds" \
  --statistic Minimum \
  --period 86400 \
  --evaluation-periods 1 \
  --threshold 2592000 \  # 30 dias em segundos
  --comparison-operator LessThanThreshold
```

Checklist de Validação:

- Confirmar que o certificado está ativo `aws rds describe-certificates`
- Verificar data de expiração (mínimo 6 meses de validade).
- Testar conexão com `sslmode=verify-full`
- Configurar alarme CloudWatch para expiração
- Monitora o período de validade do certificado SSL do RDS
- Alerta quando o certificado está próximo da expiração (no caso, 30 dias antes)

Vantagens:

✔Garante comunicação criptografada

✔Previne interceptação de dados

✔Valida autenticidade do servidor

✔Atende requisitos de compliance

Artigo por Thaise Oliveira - 2code4coffee
