TESTE TÃ‰CNICO DevOps

Requisitos

1. Utilizar infraestrutura AWS para a parte prÃ¡tica. (NÃvel gratuito)
2. Demonstrar conhecimento prÃ¡tico de configuraÃ§Ãµes de IaC.
3. As soluÃ§Ãµes prÃ¡ticas devem ser documentadas e versionadas no GitHub.

InstruÃ§Ãµes

1. O teste pode ser realizado em um Ãºnico projeto, mas cada exercÃcio deve estar claramente identificado.
2. O candidato deve fornecer instruÃ§Ãµes detalhadas no README para reproduzir as soluÃ§Ãµes.

ExercÃcios e Respostas

1. Infraestrutura AWS

Pergunta: Crie uma configuraÃ§Ã£o bÃ¡sica utilizando o uma ferramenta de IaC para provisionar uma instÃ¢ncia EC2 com RDS (MySQL) e S3 dentro da AWS. Providencie polÃticas bÃ¡sicas de seguranÃ§a para acessar a instÃ¢ncia.

Resposta: Escolhi o Terraform para criar a infraestrutura por ser uma ferramenta simples e muito usada no mercado. Vou criar:

1. Uma EC2 para rodar a aplicaÃ§Ã£o
2. Um banco MySQL (RDS) para guardar os dados
3. Um bucket S3 para arquivos

SeguranÃ§a bÃ¡sica implementada: - SÃ³ permite SSH do meu IP - HTTP permitido apenas da rede da empresa - Banco de dados sÃ³ pode ser acessado pela EC2 - S3 com acesso restrito

```
# Criar EC2 para a aplicaÃ§Ã£o
resource "aws_instance" "app" {
  ami           = "ami-0c55b159cbfafe1f0"  # Amazon Linux 2
  instance_type = "t2.micro"               # VersÃ£o gratuita

  vpc_security_group_ids = [aws_security_group.app.id]
}

# Regras de acesso
resource "aws_security_group" "app" {
  name = "app-sg"

  # SSH - SÃ³ meu IP pode acessar
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["${var.meu_ip}/32"]
  }

  # HTTP - Acesso web
  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["${var.rede_empresa}"]
  }
}
```

2. Infraestrutura como CÃ³digo (IaC)

Pergunta: Utilize uma ferramenta de IaC para criar um load balancer na AWS que distribui o trÃ¡fego entre duas instÃ¢ncias EC2. Explique o processo e as decisÃµes de configuraÃ§Ã£o no README. Explique porque escolheu a ferramenta de IaC usada.

Resposta: Escolhi Terraform como ferramenta IaC pelos seguintes motivos: - FÃ¡cil de aprender e usar - Boa documentaÃ§Ã£o e comunidade ativa - IntegraÃ§Ã£o nativa com AWS - Permite ver as mudanÃ§as antes de aplicar

Processo de implementaÃ§Ã£o:

1. DecisÃµes de ConfiguraÃ§Ã£o:
    - Application Load Balancer (ALB) por suportar HTTP/HTTPS

- Duas EC2 em zonas diferentes para alta disponibilidade
- Health check na porta 80 para garantir que aplicaÃ§Ã£o estÃ¡ respondendo
- Security groups permitindo apenas trÃ¡fego necessÃ¡rio

```
# Criar Application Load Balancer
resource "aws_lb" "app_lb" {
  name               = "app-lb"
  internal           = false                    # LB pÃºblico
  load_balancer_type = "application"           # ALB para HTTP/HTTPS
  security_groups    = [aws_security_group.lb_sg.id]
  subnets            = [aws_subnet.public_1.id, aws_subnet.public_2.id]  # Multi-AZ
}
```

3. Continuidade de NegÃ³cio

Pergunta: Descreva um plano bÃ¡sico para garantir a continuidade dos serviÃ§os crÃticos no AWS em caso de falha na regiÃ£o principal. Que serviÃ§os vocÃª usaria e como os configuraria?

Resposta: Para garantir a continuidade dos serviÃ§os, implementei dois nÃveis de proteÃ§Ã£o:

A. Em caso de falha de uma Zona de Disponibilidade:

1. Multi-AZ na mesma regiÃ£o:
   - EC2 distribuÃdas em duas AZs (us-east-1a e us-east-1b)
   - RDS com Multi-AZ para failover automÃ¡tico
   - Load Balancer distribuindo trÃ¡fego entre AZs

B. Em caso de falha da regiÃ£o principal (us-east-1):

1. Backup em RegiÃ£o SecundÃ¡ria (us-west-2):
   - ReplicaÃ§Ã£o cross-region do RDS
   - ReplicaÃ§Ã£o do bucket S3
   - AMIs copiadas para regiÃ£o secundÃ¡ria
2. Processo de Failover:
   - Route53 redireciona trÃ¡fego para regiÃ£o secundÃ¡ria
   - RDS promove rÃ©plica para master
   - EC2 Ã© iniciada usando AMIs da regiÃ£o secundÃ¡ria

4. Monitoramento e Logging

Pergunta: Configure uma soluÃ§Ã£o de monitoramento para uma instÃ¢ncia EC2 utilizando AWS CloudWatch. Colete mÃ©tricas bÃ¡sicas (CPU, memÃ³ria) e registre logs da aplicaÃ§Ã£o simulada.

Resposta: Para implementar o monitoramento da EC2 usando CloudWatch, seguirei estas etapas:

1. Instalar o CloudWatch Agent:

```
sudo yum install -y amazon-cloudwatch-agent
```

2. Configurar coleta de mÃ©tricas:

```
{
  "metrics": {
    "append_dimensions": {
      "InstanceId": "${aws:InstanceId}"
    },
    "metrics_collected": {
      "cpu": {
        "measurement": ["cpu_usage_idle", "cpu_usage_user", "cpu_usage_system"]
      },
      "mem": {
        "measurement": ["mem_used_percent"]
      }
    }
  }
}
```

3. Criar logs da aplicaÃ§Ã£o:

```
sudo mkdir -p /var/log/app
echo "$(date) - AplicaÃ§Ã£o iniciada" >> /var/log/app/app.log
```

5. Pipeline CI/CD

Pergunta: Desenvolva um pipeline bÃ¡sico de CI/CD com Git, utilizando um projeto de API simples em Node.js ou PHP.

Resposta: Criei um pipeline bÃ¡sico usando GitHub Actions para automatizar o deploy:

```
name: CI/CD Pipeline

on:
  push:
    branches: [ main ]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2

    - name: Setup Node.js
      uses: actions/setup-node@v2
      with:
        node-version: '14'

    - name: Install dependencies
      run: npm install

    - name: Run tests
      run: npm test
```

6. SeguranÃ§a no S3

Pergunta: Quais prÃ¡ticas de seguranÃ§a vocÃª aplicaria para garantir que dados armazenados no S3 sejam acessados apenas por usuÃ¡rios autorizados?

Resposta: Para garantir a seguranÃ§a dos dados no S3, implementaria:

1. PolÃticas de Bucket:
   - Bloqueio de acesso pÃºblico
   - PermissÃµes granulares por IAM
   - Criptografia em repouso
2. Controle de Acesso:
   - Usar IAM Roles
   - Bucket policies restritas
   - Logs de acesso habilitados

7. OtimizaÃ§Ã£o de Performance

Pergunta: Descreva procedimentos para otimizar a performance de uma aplicaÃ§Ã£o web com milhares de acessos simultÃ¢neos.

Resposta: Para otimizar a performance, podemos configurar:

1. DistribuiÃ§Ã£o de Carga:
   - Load Balancer entre mÃºltiplas EC2s
   - Auto Scaling baseado em demanda
   - CloudFront para conteÃºdo estÃ¡tico
2. Cache:
   - ElastiCache para dados frequentes
   - CloudFront para assets
   - Cache no banco de dados

8. ResoluÃ§Ã£o de Problemas

Pergunta: Diante de logs que indicam falha intermitente em um serviÃ§o, que passos vocÃª seguiria?

Resposta: Para resolver uma falha intermitente no serviÃ§o, eu seguiria estes passos:

1. Coleta de Dados:
   - Analisar logs do CloudWatch
   - Verificar mÃ©tricas de recursos
   - Identificar padrÃµes de erro
2. AnÃ¡lise:
   - Correlacionar eventos

- Verificar dependÃªncias
- Testar em ambiente isolado

9. Lambda vs EC2

Pergunta: Explique a diferenÃ§a entre EC2 e Lambda e em quais cenÃ¡rios vocÃª usaria cada um.

Resposta: EC2 Ã© como um servidor virtual na AWS. Ã‰ bom para: - Sites e aplicaÃ§Ãµes que precisam ficar online 24 horas - Sistemas que precisam de configuraÃ§Ã£o especÃfica - AplicaÃ§Ãµes que rodam constantemente

Lambda Ã© para execuÃ§Ã£o de cÃ³digo sem servidor. Ideal para: - Processamento em lote - APIs com trÃ¡fego variÃ¡vel - AutomaÃ§Ã£o de tarefas

10. AutomaÃ§Ã£o

Pergunta: DÃª exemplos de tarefas repetitivas que poderiam ser automatizadas em um pipeline CI/CD.

Resposta: No meu trabalho com pipeline CI/CD, percebi vÃ¡rias tarefas que podemos automatizar:

1. Testes:
   - Testes unitÃ¡rios automÃ¡ticos
   - AnÃ¡lise de cÃ³digo estÃ¡tica
   - VerificaÃ§Ã£o de seguranÃ§a
2. Deploy:
   - Build automÃ¡tico
   - Deploy em ambientes de teste
   - Rollback em caso de erro

11. ExperiÃªncia Profissional

Pergunta: Descreva uma experiÃªncia em que vocÃª teve que implementar uma soluÃ§Ã£o de infraestrutura crÃtica.

Resposta: Na minha Ãºltima experiÃªncia, trabalhei em um projeto onde precisei migrar uma aplicaÃ§Ã£o PHP para containers Docker. Foi um desafio interessante porque:

1. Desafios:
   - Sistema legado sem documentaÃ§Ã£o
   - DependÃªncias antigas
   - Zero downtime necessÃ¡rio
2. SoluÃ§Ãµes:
   - Documentei toda a aplicaÃ§Ã£o
   - Modernizei as dependÃªncias
   - Implementei CI/CD