# TESTE TÉCNICO DevOps

## Requisitos

1. Utilizar infraestrutura AWS para a parte prática. (Nível gratuito)
2. Demonstrar conhecimento prático de configurações de IaC.
3. As soluções práticas devem ser documentadas e versionadas no GitHub.

## Instruções

1. O teste pode ser realizado em um único projeto, mas cada exercício deve estar claramente identificado.
2. O candidato deve fornecer instruções detalhadas no README para reproduzir as soluções.

## Exercícios e Respostas

### 1. Infraestrutura AWS

**Pergunta:** Crie uma configuração básica utilizando o uma ferramenta de IaC para provisionar uma instância EC2 com RDS (MySQL) e S3 dentro da AWS. Providencie políticas básicas de segurança para acessar a instância.

**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 aplicação
2. Um banco MySQL (RDS) para guardar os dados
3. Um bucket S3 para arquivos

Segurança básica implementada: - Só permite SSH do meu IP - HTTP permitido apenas da rede da empresa - Banco de dados só pode ser acessado pela EC2 - S3 com acesso restrito

# Criar EC2 para a aplicação  
resource "aws\_instance" "app" {  
 ami = "ami-0c55b159cbfafe1f0" # Amazon Linux 2  
 instance\_type = "t2.micro" # Versão gratuita  
   
 vpc\_security\_group\_ids = [aws\_security\_group.app.id]  
}  
  
# Regras de acesso  
resource "aws\_security\_group" "app" {  
 name = "app-sg"  
  
 # SSH - Só 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 Código (IaC)

**Pergunta:** Utilize uma ferramenta de IaC para criar um load balancer na AWS que distribui o tráfego entre duas instâncias EC2. Explique o processo e as decisões de configuração no README. Explique porque escolheu a ferramenta de IaC usada.

**Resposta:** Escolhi Terraform como ferramenta IaC pelos seguintes motivos: - Fácil de aprender e usar - Boa documentação e comunidade ativa - Integração nativa com AWS - Permite ver as mudanças antes de aplicar

Processo de implementação:

1. Decisões de Configuração:
   * Application Load Balancer (ALB) por suportar HTTP/HTTPS
   * Duas EC2 em zonas diferentes para alta disponibilidade
   * Health check na porta 80 para garantir que aplicação está respondendo
   * Security groups permitindo apenas tráfego necessário

# Criar Application Load Balancer  
resource "aws\_lb" "app\_lb" {  
 name = "app-lb"  
 internal = false # LB público  
 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 Negócio

**Pergunta:** Descreva um plano básico para garantir a continuidade dos serviços críticos no AWS em caso de falha na região principal. Que serviços você usaria e como os configuraria?

**Resposta:** Para garantir a continuidade dos serviços, implementei dois níveis de proteção:

1. Em caso de falha de uma Zona de Disponibilidade:
2. Multi-AZ na mesma região:
   * EC2 distribuídas em duas AZs (us-east-1a e us-east-1b)
   * RDS com Multi-AZ para failover automático
   * Load Balancer distribuindo tráfego entre AZs
3. Em caso de falha da região principal (us-east-1):
4. Backup em Região Secundária (us-west-2):
   * Replicação cross-region do RDS
   * Replicação do bucket S3
   * AMIs copiadas para região secundária
5. Processo de Failover:
   * Route53 redireciona tráfego para região secundária
   * RDS promove réplica para master
   * EC2 é iniciada usando AMIs da região secundária

### 4. Monitoramento e Logging

**Pergunta:** Configure uma solução de monitoramento para uma instância EC2 utilizando AWS CloudWatch. Colete métricas básicas (CPU, memória) e registre logs da aplicação 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

1. Configurar coleta de métricas:

{  
 "metrics": {  
 "append\_dimensions": {  
 "InstanceId": "${aws:InstanceId}"  
 },  
 "metrics\_collected": {  
 "cpu": {  
 "measurement": ["cpu\_usage\_idle", "cpu\_usage\_user", "cpu\_usage\_system"]  
 },  
 "mem": {  
 "measurement": ["mem\_used\_percent"]  
 }  
 }  
 }  
}

1. Criar logs da aplicação:

sudo mkdir -p /var/log/app  
echo "$(date) - Aplicação iniciada" >> /var/log/app/app.log

### 5. Pipeline CI/CD

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

**Resposta:** Criei um pipeline básico 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. Segurança no S3

**Pergunta:** Quais práticas de segurança você aplicaria para garantir que dados armazenados no S3 sejam acessados apenas por usuários autorizados?

**Resposta:** Para garantir a segurança dos dados no S3, implementaria:

1. Políticas de Bucket:
   * Bloqueio de acesso público
   * Permissões granulares por IAM
   * Criptografia em repouso
2. Controle de Acesso:
   * Usar IAM Roles
   * Bucket policies restritas
   * Logs de acesso habilitados

### 7. Otimização de Performance

**Pergunta:** Descreva procedimentos para otimizar a performance de uma aplicação web com milhares de acessos simultâneos.

**Resposta:** Para otimizar a performance, podemos configurar:

1. Distribuição de Carga:
   * Load Balancer entre múltiplas EC2s
   * Auto Scaling baseado em demanda
   * CloudFront para conteúdo estático
2. Cache:
   * ElastiCache para dados frequentes
   * CloudFront para assets
   * Cache no banco de dados

### 8. Resolução de Problemas

**Pergunta:** Diante de logs que indicam falha intermitente em um serviço, que passos você seguiria?

**Resposta:** Para resolver uma falha intermitente no serviço, eu seguiria estes passos:

1. Coleta de Dados:
   * Analisar logs do CloudWatch
   * Verificar métricas de recursos
   * Identificar padrões de erro
2. Análise:
   * Correlacionar eventos
   * Verificar dependências
   * Testar em ambiente isolado

### 9. Lambda vs EC2

**Pergunta:** Explique a diferença entre EC2 e Lambda e em quais cenários você usaria cada um.

**Resposta:** EC2 é como um servidor virtual na AWS. É bom para: - Sites e aplicações que precisam ficar online 24 horas - Sistemas que precisam de configuração específica - Aplicações que rodam constantemente

Lambda é para execução de código sem servidor. Ideal para: - Processamento em lote - APIs com tráfego variável - Automação de tarefas

### 10. Automação

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

**Resposta:** No meu trabalho com pipeline CI/CD, percebi várias tarefas que podemos automatizar:

1. Testes:
   * Testes unitários automáticos
   * Análise de código estática
   * Verificação de segurança
2. Deploy:
   * Build automático
   * Deploy em ambientes de teste
   * Rollback em caso de erro

### 11. Experiência Profissional

**Pergunta:** Descreva uma experiência em que você teve que implementar uma solução de infraestrutura crítica.

**Resposta:** Na minha última experiência, trabalhei em um projeto onde precisei migrar uma aplicação PHP para containers Docker. Foi um desafio interessante porque:

1. Desafios:
   * Sistema legado sem documentação
   * Dependências antigas
   * Zero downtime necessário
2. Soluções:
   * Documentei toda a aplicação
   * Modernizei as dependências
   * Implementei CI/CD