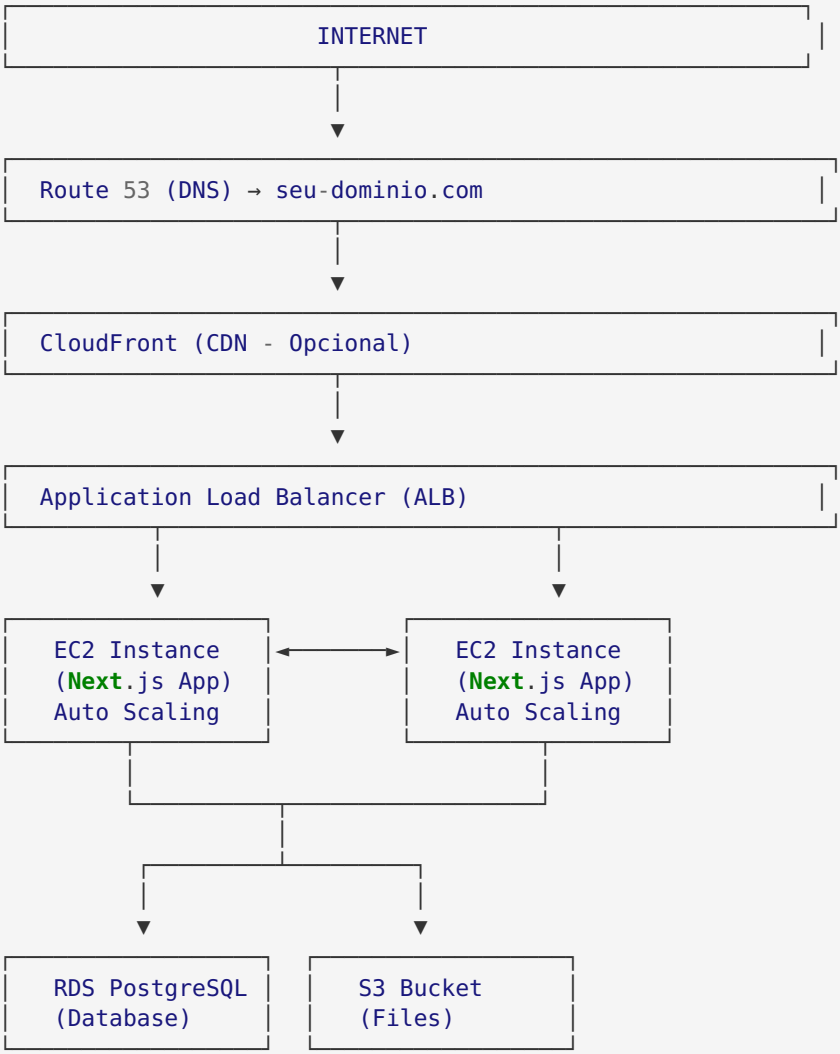
☁️ Guia Completo: Deploy do RecruitAI na AWS

Este guia detalha **passo a passo** como fazer o deploy completo do projeto RecruitAI na AWS.

📋 Índice

🏗️ Arquitetura AWS



✅ Pré-requisitos

1. Conta AWS

- Crie uma conta em aws.amazon.com (https://aws.amazon.com)
- Configure MFA (autenticação de dois fatores) para segurança
- Tenha um cartão de crédito válido cadastrado

2. AWS CLI Instalado

```
# Instalar AWS CLI
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install

# Verificar instalação
aws --version
```

3. Credenciais AWS

1. Acesse AWS IAM Console (https://console.aws.amazon.com/iam/)
2. Crie um usuário com permissões:
   - `AmazonEC2FullAccess`
   - `AmazonRDSFullAccess`
   - `AmazonS3FullAccess`
   - `IAMReadOnlyAccess`
3. Gere Access Keys (guarde em local seguro!)

```
# Configurar AWS CLI
aws configure

# Será solicitado:
# AWS Access Key ID: [Cole sua Access Key]
# AWS Secret Access Key: [Cole sua Secret Key]
# Default region name: us-east-1
# Default output format: json
```

🗄️ Parte 1: Configurar RDS (PostgreSQL)

Passo 1.1: Acessar Console RDS

1. Acesse AWS RDS Console (https://console.aws.amazon.com/rds/)
2. Clique em **"Create database"**

Passo 1.2: Configuração Básica

Engine Options:

```
Engine type: PostgreSQL
Version: PostgreSQL 15.x (mais recente)
```

Templates:

```
⚪ Production (para produção real)
🔘 Free tier (para testes - 750h/mês grátis no primeiro ano)
⚪ Dev/Test
```

Passo 1.3: Settings

```
DB instance identifier: recruitai-db
Credentials:
  Master username: recruitai_admin
  Master password: [Crie senha forte - mínimo 8 caracteres]
  Confirm password: [Repita a senha]
```

⚠️ **IMPORTANTE: Guarde essas credenciais em local seguro!**

Passo 1.4: Instance Configuration

Para Free Tier:

```
DB instance class: db.t3.micro (750h grátis/mês)
Storage type: General Purpose SSD (gp2)
Allocated storage: 20 GB
```

Para Produção (recomendado):

```
DB instance class: db.t3.small ou db.t3.medium
Storage type: General Purpose SSD (gp3)
Allocated storage: 100 GB
Enable storage autoscaling: Sim
Maximum storage threshold: 200 GB
```

Passo 1.5: Connectivity

```
Compute resource: Don't connect to an EC2 compute resource
Network type: IPv4
VPC: default
Public access: Yes (para facilitar, mas configure Security Group corretamente!)
VPC security group: Create new
New security group name: recruitai-db-sg
```

Passo 1.6: Database Authentication

```
Database authentication: Password authentication
```

Passo 1.7: Additional Configuration

```
Initial database name: recruitai_prod
DB parameter group: default.postgres15
Backup:
  Enable automated backups: Yes
  Backup retention period: 7 days
  Backup window: 03:00-04:00 (UTC)
Monitoring:
  Enable Enhanced monitoring: Yes (recomendado)
Maintenance:
  Enable auto minor version upgrade: Yes
  Maintenance window: Domingo 04:00-05:00 (UTC)
```

Passo 1.8: Criar Database

1. Clique em **“Create database”**
2. Aguarde 5-10 minutos até o status mudar para **“Available”**

Passo 1.9: Obter Connection String

1. Clique no nome do banco criado (`recruitai-db`)
2. Na seção **“Connectivity & security”**, copie o **Endpoint**

Exemplo:

```
recruitai-db.abc123xyz.us-east-1.rds.amazonaws.com
```

1. Construa sua `DATABASE_URL`:

```
DATABASE_URL="postgresql://recruitai_admin:SUA_SENHA@recruitai-db.abc123xyz.us-east-1.rds.amazonaws.com:5432/recruitai_prod?schema=public"
```

Passo 1.10: Configurar Security Group

1. Clique em **"VPC security groups"** > `recruitai-db-sg`
2. Vá na aba **"Inbound rules"**
3. Clique em **"Edit inbound rules"**
4. Adicione regra:

```
Type: PostgreSQL
Protocol: TCP
Port Range: 5432
Source:
  - Seu IP (para testes): [Seu IP]/32
  - Security Group da EC2 (produção): sg-xxxxxxxxx
```

1. Salve as regras

📦 Parte 2: Configurar S3 (Armazenamento)

Passo 2.1: Criar Bucket

1. Acesse AWS S3 Console (https://s3.console.aws.amazon.com/s3/)
2. Clique em **"Create bucket"**

Passo 2.2: Configuração do Bucket

```
Bucket name: recruitai-platform-files-[seu-nome-unico]
  Exemplo: recruitai-platform-files-prod-2024
  ⚠️ Nome deve ser ÚNICO globalmente

AWS Region: us-east-1 (mesma região do RDS)

Object Ownership: ACLs disabled (recommended)

Block Public Access settings:
  ✅ Block all public access (recomendado)

Bucket Versioning: Disable (ou Enable para backup)

Default encryption:
  Encryption type: Server-side encryption with Amazon S3 managed keys (SSE-S3)
```

Passo 2.3: Criar Bucket

1. Clique em **"Create bucket"**
2. Anote o nome do bucket criado

Passo 2.4: Criar Estrutura de Pastas

1. Clique no bucket criado
2. Clique em **“Create folder”**
3. Crie as seguintes pastas:

```
ats-platform/
├── uploads/
│   ├── resumes/
│   └── logos/
└── temp/
```

Passo 2.5: Configurar CORS (para uploads diretos)

1. Clique no bucket
2. Vá na aba **“Permissions”**
3. Role até **“Cross-origin resource sharing (CORS)”**
4. Clique em **“Edit”**
5. Cole o seguinte JSON:

```
[
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "GET",
            "PUT",
            "POST",
            "DELETE",
            "HEAD"
        ],
        "AllowedOrigins": [
            "http://localhost:3000",
            "https://seu-dominio.com"
        ],
        "ExposeHeaders": [
            "ETag"
        ],
        "MaxAgeSeconds": 3000
    }
]
```

1. Salve as alterações

Passo 2.6: Criar IAM User para S3

1. Acesse IAM Console (https://console.aws.amazon.com/iam/)
2. Vá em **“Users”** > **“Add users”**

```
User name: recruitai-s3-access
Access type: Programmatic access (Access key)
```

1. **Permissions:**
   - Clique em **“Attach existing policies directly”**
   - Procure e selecione: `AmazonS3FullAccess`

2. **Tags** (opcional):

```
Key: Environment
Value: Production
```

1. Clique em **“Create user”**

2. ⚠️ **IMPORTANTE:** Na tela de confirmação:
   - Baixe o arquivo CSV com as credenciais
   - Copie:
     - `Access key ID`
     - `Secret access key`

Exemplo:

```
AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

Passo 2.7: Testar Acesso S3

```
# Configurar credenciais S3 no CLI
aws configure set aws_access_key_id AKIAIOSFODNN7EXAMPLE
aws configure set aws_secret_access_key wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

# Listar buckets
aws s3 ls

# Fazer upload de teste
echo "test" > test.txt
aws s3 cp test.txt s3://seu-bucket-name/test.txt

# Verificar
aws s3 ls s3://seu-bucket-name/
```

🖥️ Parte 3: Configurar EC2 (Servidor)

Passo 3.1: Lançar Instância EC2

1. Acesse EC2 Console (https://console.aws.amazon.com/ec2/)
2. Clique em **“Launch instance”**

Passo 3.2: Nome e Tags

```
Name: RecruitAI-Production
```

Passo 3.3: Application and OS Images (AMI)

```
Quick Start: Ubuntu
AMI: Ubuntu Server 22.04 LTS (HVM), SSD Volume Type
Architecture: 64-bit (x86)
```

Passo 3.4: Instance Type

Para testes (Free Tier):

```
Instance type: t2.micro (750h grátis/mês no 1º ano)
  - 1 vCPU
  - 1 GB RAM
```

Para produção (recomendado):

```
Instance type: t3.small ou t3.medium
  t3.small:  2 vCPU, 2 GB RAM (~$15/mês)
  t3.medium: 2 vCPU, 4 GB RAM (~$30/mês)
```

Passo 3.5: Key Pair (Login)

1. Clique em **“Create new key pair”**

```
Key pair name: recruitai-key
Key pair type: RSA
Private key file format: .pem
```

1. Clique em **“Create key pair”**
2. ⚠️ **IMPORTANTE:** Salve o arquivo `.pem` em local seguro!

```
# Mover para pasta SSH
mkdir -p ~/.ssh
mv ~/Downloads/recruitai-key.pem ~/.ssh/
chmod 400 ~/.ssh/recruitai-key.pem
```

Passo 3.6: Network Settings

```
VPC: default
Subnet: No preference
Auto-assign public IP: Enable

Firewall (Security groups): Create security group
Security group name: recruitai-web-sg
Description: Allow HTTP, HTTPS and SSH

Inbound Security Group Rules:
```

Type	Protocol	Port	Source	Description
SSH	TCP	22	My IP	SSH acesso
HTTP	TCP	80	Anywhere (0.0.0.0/0)	Web HTTP
HTTPS	TCP	443	Anywhere (0.0.0.0/0)	Web HTTPS
Custom TCP	TCP	3000	Anywhere (0.0.0.0/0)	Next.js (temporário)

Passo 3.7: Configure Storage

```
Volume Type: gp3 (General Purpose SSD)
Size: 30 GB (mínimo 20 GB)
Delete on termination: Yes
```

Passo 3.8: Advanced Details (Opcional mas Recomendado)

Role até **“User data”** e cole:

```
#!/bin/bash
# Script de inicialização automática

# Atualizar sistema
apt-get update -y
apt-get upgrade -y

# Instalar Node.js 18.x
curl -fsSL https://deb.nodesource.com/setup_18.x | bash -
apt-get install -y nodejs

# Instalar Yarn
npm install -g yarn

# Instalar PM2 (gerenciador de processos)
npm install -g pm2

# Instalar Nginx
apt-get install -y nginx

# Instalar Git
apt-get install -y git

# Criar pasta para aplicação
mkdir -p /var/www/recruitai
chown -R ubuntu:ubuntu /var/www/recruitai

# Configurar firewall
ufw allow 22
ufw allow 80
ufw allow 443
ufw allow 3000
ufw --force enable

echo "Servidor configurado com sucesso!" > /home/ubuntu/setup-complete.txt
```

Passo 3.9: Lançar Instância

1. No canto direito, veja o resumo
2. Clique em **“Launch instance”**
3. Aguarde 2-3 minutos

Passo 3.10: Obter IP Público

1. Clique em **“Instances”** no menu lateral
2. Selecione sua instância `RecruitAI-Production`
3. Copie o **“Public IPv4 address”**

Exemplo: `54.123.45.67`

Passo 3.11: Conectar via SSH

```
# Conectar à instância
ssh -i ~/.ssh/recruitai-key.pem ubuntu@54.123.45.67

# Se der erro "Permission denied":
chmod 400 ~/.ssh/recruitai-key.pem
ssh -i ~/.ssh/recruitai-key.pem ubuntu@54.123.45.67
```

Saída esperada:

```
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-1047-aws x86_64)
...
ubuntu@ip-172-31-xx-xx:~$
```

Passo 3.12: Verificar Instalações

```
# Node.js
node --version  # Deve mostrar v18.x.x

# Yarn
yarn --version  # Deve mostrar 1.22.x

# PM2
pm2 --version   # Deve mostrar 5.x.x

# Git
git --version   # Deve mostrar git version 2.x.x

# Nginx
nginx -v        # Deve mostrar nginx version
```

🚀 Parte 4: Deploy da Aplicação

Passo 4.1: Clonar Repositório

```
# Conectado na EC2 via SSH

# Navegar para pasta da aplicação
cd /var/www/recruitai

# Clonar do GitHub (substitua SEU_USUARIO)
git clone https://github.com/SEU_USUARIO/recruit-ai-platform.git .

# Se o repositório for privado, você precisará configurar SSH Key ou usar token
```

Para repositório privado com token:

```
git clone https://SEU_TOKEN@github.com/SEU_USUARIO/recruit-ai-platform.git .
```

Passo 4.2: Criar arquivo .env

```
cd /var/www/recruitai/nextjs_space

# Criar arquivo .env
nano .env
```

Cole as variáveis (substitua pelos valores reais):

```
# ====================================
# BANCO DE DADOS (RDS)
# ====================================
DATABASE_URL="postgresql://recruitai_admin:SUA_SENHA@recruitai-db.abc123xyz.us-
east-1.rds.amazonaws.com:5432/recruitai_prod?schema=public"

# ====================================
# NEXTAUTH
# ====================================
NEXTAUTH_SECRET="seu-secret-gerado-com-openssl"
NEXTAUTH_URL="http://54.123.45.67:3000"
# Ou com domínio: NEXTAUTH_URL="https://seu-dominio.com"

# ====================================
# STRIPE
# ====================================
STRIPE_SECRET_KEY="sk_live_seu_secret_key"
STRIPE_PUBLISHABLE_KEY="pk_live_sua_publishable_key"
STRIPE_WEBHOOK_SECRET="whsec_seu_webhook_secret"

# ====================================
# AWS S3
# ====================================
AWS_REGION="us-east-1"
AWS_BUCKET_NAME="recruitai-platform-files-prod-2024"
AWS_FOLDER_PREFIX="ats-platform/"
AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"

# ====================================
# GOOGLE OAUTH
# ====================================
GOOGLE_CLIENT_ID="seu-client-id.apps.googleusercontent.com"
GOOGLE_CLIENT_SECRET="seu-client-secret"

# ====================================
# LINKEDIN OAUTH
# ====================================
LINKEDIN_CLIENT_ID="seu-client-id"
LINKEDIN_CLIENT_SECRET="seu-client-secret"

# ====================================
# ABACUS.AI
# ====================================
ABACUSAI_API_KEY="sua-api-key"

# ====================================
# CRON SECRET
# ====================================
CRON_SECRET="seu-cron-secret"
```

Salve (`Ctrl+O` , `Enter` , `Ctrl+X`)

Passo 4.3: Gerar NEXTAUTH_SECRET

```
# Gerar secret seguro
openssl rand -base64 32

# Copie o resultado e atualize no .env
nano .env
# Cole em NEXTAUTH_SECRET="aqui"
```

Passo 4.4: Instalar Dependências

```
# Instalar dependências do projeto
yarn install

# Deve levar 2-5 minutos
```

Passo 4.5: Configurar Banco de Dados

```
# Executar migrações Prisma
yarn prisma db push

# Seed inicial (dados de teste)
yarn tsx scripts/seed.ts
```

Saída esperada:

```
✓ Planos criados com sucesso!
✓ Superadmin criado com sucesso!
✓ Usuário de teste criado com sucesso!
✓ Assinatura trial criada com sucesso!
...
```

Passo 4.6: Build da Aplicação

```
# Fazer build de produção
yarn build

# Deve levar 1-3 minutos
```

Saída esperada:

```
✓ Compiled successfully
✓ Generating static pages
Route (app)                                 Size     First Load JS
...
```

Passo 4.7: Configurar PM2

```
# Criar arquivo de configuração PM2
nano ecosystem.config.js
```

Cole:

```
module.exports = {
  apps: [{
    name: 'recruitai',
    script: 'yarn',
    args: 'start',
    cwd: '/var/www/recruitai/nextjs_space',
    instances: 1,
    autorestart: true,
    watch: false,
    max_memory_restart: '1G',
    env: {
      NODE_ENV: 'production',
      PORT: 3000
    }
  }]
};
```

Salve (`Ctrl+O` , `Enter` , `Ctrl+X`)

Passo 4.8: Iniciar Aplicação com PM2

```
# Iniciar aplicação
pm2 start ecosystem.config.js

# Configurar PM2 para iniciar no boot
pm2 startup
# Copie e execute o comando que aparecerá

pm2 save

# Ver logs
pm2 logs recruitai

# Verificar status
pm2 status
```

Saída esperada:

```
┌────┬────────────┬─────────┬──────┬─────────┬──────────┐
│ id │ name       │ mode    │ ↺    │ status  │ cpu      │
├────┼────────────┼─────────┼──────┼─────────┼──────────┤
│ 0  │ recruitai  │ fork    │ 0    │ online  │ 0%       │
└────┴────────────┴─────────┴──────┴─────────┴──────────┘
```

Passo 4.9: Testar Aplicação

```
# Testar se está rodando
curl http://localhost:3000

# Deve retornar HTML da página inicial
```

No navegador, acesse:

```
http://54.123.45.67:3000
```

⚙️ Parte 5: Configurações Finais

Passo 5.1: Configurar Nginx como Proxy Reverso

```
# Criar arquivo de configuração Nginx
sudo nano /etc/nginx/sites-available/recruitai
```

Cole:

```
server {
    listen 80;
    server_name seu-dominio.com www.seu-dominio.com;
    # Ou use o IP: server_name 54.123.45.67;

    # Logs
    access_log /var/log/nginx/recruitai-access.log;
    error_log /var/log/nginx/recruitai-error.log;

    # Proxy para Next.js
    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    # Cache de assets estáticos
    location /_next/static {
        proxy_pass http://localhost:3000;
        add_header Cache-Control "public, max-age=31536000, immutable";
    }

    # Tamanho máximo de upload (para currículos)
    client_max_body_size 10M;
}
```

Salve (`Ctrl+O` , `Enter` , `Ctrl+X`)

Passo 5.2: Ativar Configuração Nginx

```
# Criar link simbólico
sudo ln -s /etc/nginx/sites-available/recruitai /etc/nginx/sites-enabled/

# Remover configuração padrão
sudo rm /etc/nginx/sites-enabled/default

# Testar configuração
sudo nginx -t

# Reiniciar Nginx
sudo systemctl restart nginx
sudo systemctl enable nginx
```

Passo 5.3: Configurar SSL com Let's Encrypt (HTTPS)

Pré-requisito: Ter um domínio apontando para o IP da EC2

```
# Instalar Certbot
sudo apt-get install -y certbot python3-certbot-nginx

# Obter certificado SSL
sudo certbot --nginx -d seu-dominio.com -d www.seu-dominio.com

# Será solicitado:
# Email: seu-email@exemplo.com
# Termos: (A)gree
# Compartilhar email: (Y)es ou (N)o
# Redirect HTTP to HTTPS: 2 (Redirect)
```

Renovação automática:

```
# Testar renovação
sudo certbot renew --dry-run

# Configurar renovação automática (já configurado por padrão)
sudo systemctl status certbot.timer
```

Passo 5.4: Configurar Firewall UFW

```
# Permitir apenas portas necessárias
sudo ufw allow 22    # SSH
sudo ufw allow 80    # HTTP
sudo ufw allow 443   # HTTPS
sudo ufw enable

# Verificar status
sudo ufw status
```

Passo 5.5: Configurar Swap (para instâncias pequenas)

```
# Criar arquivo swap de 2GB
sudo fallocate -l 2G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile

# Tornar permanente
echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab

# Verificar
free -h
```

Passo 5.6: Configurar Logs Rotativos

```
# Criar configuração de rotação de logs
sudo nano /etc/logrotate.d/recruitai
```

Cole:

```
/var/www/recruitai/nextjs_space/*.log {
    daily
    rotate 14
    compress
    delaycompress
    notifempty
    create 0640 ubuntu ubuntu
    sharedscripts
    postrotate
        pm2 reloadLogs
    endscript
}
```

Passo 5.7: Configurar Backup Automático

```
# Criar script de backup
nano /home/ubuntu/backup-db.sh
```

Cole:

```
#!/bin/bash
# Script de backup do banco de dados

DATE=$(date +%Y%m%d_%H%M%S)
BACKUP_DIR="/home/ubuntu/backups"
DB_NAME="recruitai_prod"
DB_USER="recruitai_admin"
DB_HOST="recruitai-db.abc123xyz.us-east-1.rds.amazonaws.com"
S3_BUCKET="recruitai-platform-files-prod-2024"

# Criar diretório se não existir
mkdir -p $BACKUP_DIR

# Fazer backup
PGPASSWORD="SUA_SENHA" pg_dump -h $DB_HOST -U $DB_USER $DB_NAME | gzip > $BACKUP_DIR/
backup_$DATE.sql.gz

# Enviar para S3
aws s3 cp $BACKUP_DIR/backup_$DATE.sql.gz s3://$S3_BUCKET/backups/

# Manter apenas últimos 7 backups locais
cd $BACKUP_DIR
ls -t | tail -n +8 | xargs -r rm

echo "Backup concluído: backup_$DATE.sql.gz"
```

Salve e configure:

```
# Dar permissão de execução
chmod +x /home/ubuntu/backup-db.sh

# Instalar PostgreSQL client
sudo apt-get install -y postgresql-client

# Testar backup
./backup-db.sh

# Agendar backup diário (3h da manhã)
crontab -e
# Adicione a linha:
0 3 * * * /home/ubuntu/backup-db.sh >> /home/ubuntu/backup.log 2>&1
```

📊 Monitoramento e Manutenção

Comandos Úteis

```
# Ver status da aplicação
pm2 status

# Ver logs em tempo real
pm2 logs recruitai

# Reiniciar aplicação
pm2 restart recruitai

# Parar aplicação
pm2 stop recruitai

# Ver uso de recursos
pm2 monit

# Ver logs do Nginx
sudo tail -f /var/log/nginx/recruitai-access.log
sudo tail -f /var/log/nginx/recruitai-error.log

# Ver uso de CPU/RAM
htop

# Ver espaço em disco
df -h

# Ver processos
ps aux | grep node
```

Atualizar Aplicação

```
# Conectar via SSH
ssh -i ~/.ssh/recruitai-key.pem ubuntu@54.123.45.67

# Ir para pasta do projeto
cd /var/www/recruitai/nextjs_space

# Puxar atualizações do GitHub
git pull origin main

# Instalar novas dependências (se houver)
yarn install

# Executar migrações (se houver)
yarn prisma db push

# Fazer novo build
yarn build

# Reiniciar aplicação
pm2 restart recruitai

# Verificar logs
pm2 logs recruitai --lines 50
```

Configurar CloudWatch (Monitoramento AWS)

1. Acesse CloudWatch Console (https://console.aws.amazon.com/cloudwatch/)
2. Vá em **“Alarms”** > **“Create alarm”**
3. Configure alarmes para:
   - CPU > 80%
   - RAM > 80%
   - Disco > 85%
   - Status Check Failed

Configurar Auto Scaling (Opcional - Produção)

Para alta disponibilidade, configure:
1. **Launch Template** (template da EC2)
2. **Target Group** (grupo de instâncias)
3. **Application Load Balancer** (distribuir tráfego)
4. **Auto Scaling Group** (escalar automaticamente)

💰 Custos Estimados AWS

Cenário 1: Free Tier (Primeiro Ano)

Serviço	Configuração	Custo Mensal
EC2 (t2.micro)	750h grátis/mês	$0
RDS (db.t3.micro)	750h grátis/mês	$0
S3	5GB storage + 20k requests	$0
TOTAL		**~$0-5/mês**

Cenário 2: Startup (Pós Free Tier)

Serviço	Configuração	Custo Mensal
EC2 (t3.small)	2 vCPU, 2GB RAM	~$15
RDS (db.t3.small)	2 vCPU, 2GB RAM, 100GB storage	~$30
S3	50GB storage + 100k requests	~$1.50
Data Transfer	100GB out	~$9
TOTAL		**~$55-60/mês**

Cenário 3: Produção

Serviço	Configuração	Custo Mensal
EC2 (2x t3.medium)	2 vCPU, 4GB RAM cada	~$60
RDS (db.t3.medium)	2 vCPU, 4GB RAM, 200GB storage	~$70
Application Load Balancer		~$16
S3	200GB storage + 500k requests	~$5
Data Transfer	500GB out	~$45
CloudWatch	Logs + metrics	~$5
TOTAL		**~$200-220/mês**

💡 **Dicas para Reduzir Custos:**
- Use **Reserved Instances** (1-3 anos): economize até 75%
- Configure **Auto Scaling**: pague apenas pelo que usar
- Use **S3 Intelligent-Tiering**: otimize custos de storage
- Delete snapshots antigos do RDS
- Configure **CloudWatch Alarms** para monitorar gastos

✅ Checklist Final

Infraestrutura

- [ ] RDS PostgreSQL criado e acessível
- [ ] S3 Bucket criado com pastas configuradas
- [ ] IAM User para S3 com credenciais geradas
- [ ] EC2 Instance criada e rodando
- [ ] Security Groups configurados corretamente
- [ ] Key Pair (.pem) salva em local seguro

Aplicação

- [ ] Código clonado do GitHub
- [ ] Arquivo .env configurado com todas as variáveis
- [ ] Dependências instaladas (`yarn install`)
- [ ] Banco de dados migrado (`yarn prisma db push`)
- [ ] Seed executado (`yarn tsx scripts/seed.ts`)
- [ ] Build de produção concluído (`yarn build`)
- [ ] Aplicação rodando com PM2
- [ ] PM2 configurado para iniciar no boot

Servidor Web

- [ ] Nginx instalado e configurado
- [ ] Proxy reverso funcionando
- [ ] SSL/HTTPS configurado (se tem domínio)
- [ ] Firewall (UFW) ativado
- [ ] Logs configurados

Manutenção

- [ ] Backup automático configurado
- [ ] Logs rotativos configurados
- [ ] CloudWatch alarms configurados (opcional)
- [ ] Documentação de credenciais em local seguro

Testes

- [ ] Site acessível via HTTP/HTTPS
- [ ] Login funcionando
- [ ] Upload de currículo funcionando
- [ ] Envio de email funcionando
- [ ] Integração Stripe funcionando
- [ ] Todos os fluxos principais testados

🎉 Parabéns!

Seu projeto RecruitAI agora está rodando na AWS! 🚀

Próximos Passos Recomendados:

1. **Configurar Domínio Personalizado**
   - Registre um domínio
   - Configure DNS apontando para o IP da EC2
   - Obtenha certificado SSL

2. **Implementar CI/CD**
   - GitHub Actions para deploy automático
   - Pipeline de testes antes do deploy

3. **Monitoramento Avançado**
   - Sentry para error tracking
   - Google Analytics para métricas de uso
   - AWS CloudWatch Dashboards personalizados

4. **Otimizações**
   - CloudFront (CDN) para assets estáticos
   - ElastiCache (Redis) para cache
   - RDS Read Replicas para escalabilidade

5. **Segurança**
   - AWS WAF (Web Application Firewall)

- Auditorias de segurança regulares
- Backup recovery tests

📞 Suporte

Se precisar de ajuda:
- 📧 AWS Support: console.aws.amazon.com/support (https://console.aws.amazon.com/support)
- 📚 Documentação AWS: docs.aws.amazon.com (https://docs.aws.amazon.com)
- 💬 Comunidade: stackoverflow.com (https://stackoverflow.com/questions/tagged/aws)

Boa sorte com seu deploy! 🎊
EOF