# 📋 MANUAL DE INSTALAÇÃO - SISTEMA SALONTIME

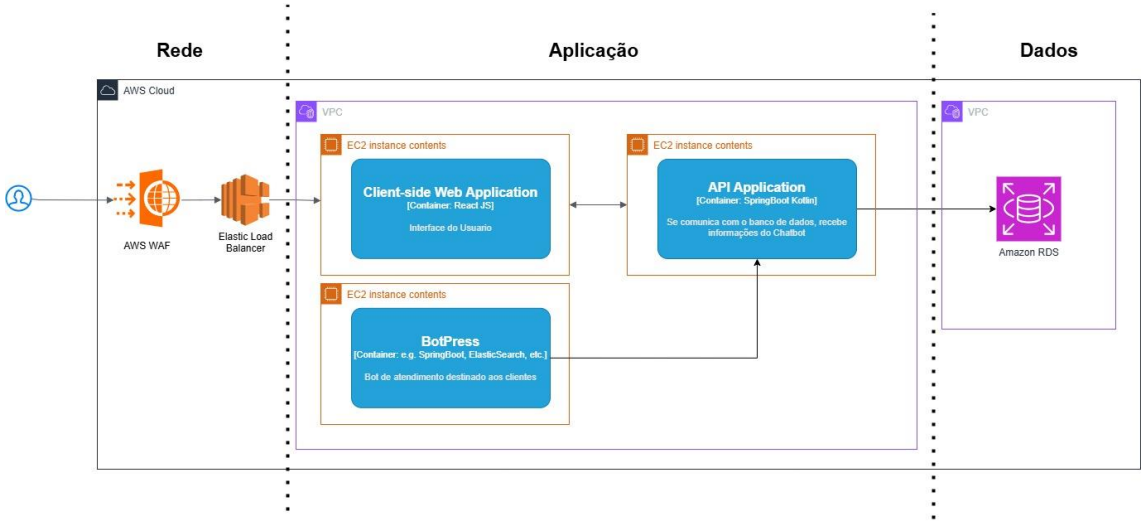
## Arquitetura de Implantação AWS

## Sumário

## 1. VISÃO GERAL DA ARQUITETURA

O sistema SalonTime é composto por:

- **Camada de Rede**: AWS WAF + Elastic Load Balancer
- **Camada de Aplicação**: 3 Instâncias EC2 (Front-end React, API Kotlin, BotPress)
- **Camada de Dados**: Amazon RDS (MySQL)




## 2. PRÉ-REQUISITOS

- Conta AWS ativa com permissões administrativas
- AWS CLI instalado e configurado
- Conhecimentos básicos de AWS Console
- Arquivos dos repositórios clonados localmente

## 3. CONFIGURAÇÃO DO AMAZON RDS (Banco de Dados)

### 3.1 **Criar Instância RDS MySQL**

3.1.1 Acesse o **Console AWS** → **RDS** → **Create database**

3.1.2 Configure os seguintes parâmetros:

Code

Engine type: MySQL

Version: 8.0 ou superior

Template: Production (ou Dev/Test para ambientes não-produtivos)

Settings:

- DB instance identifier: salontime-db

- Master username: admin

- Master password: [Defina uma senha forte]

DB instance class:

- db.t3.medium (ou conforme necessidade)

Storage:

- Storage type: General Purpose SSD (gp3)

- Allocated storage: 20 GB

- Enable storage autoscaling: Yes

- Maximum storage threshold: 100 GB

Connectivity:

- VPC: [Selecione sua VPC ou crie uma nova]

- Subnet group: Create new DB subnet group

- Public access: No

- VPC security group: Create new

  - Name: salontime-rds-sg

  - Inbound rules: MySQL/Aurora (3306) from application security group

#### 3.1.3 Configurações adicionais:

Code

Initial database name: SalonTime

Backup:

- Enable automated backups: Yes

- Backup retention period: 7 days

- Backup window: [Escolha um horário de baixo tráfego]

Monitoring:

- Enable Enhanced monitoring: Yes

3.2 Clique em **Create database** e aguarde a criação (10-15 minutos)

3.3 **Executar Scripts SQL**

3.3.1 Após a instância estar disponível, anote o **Endpoint** da RDS

3.3.2 Conecte-se ao banco usando MySQL Workbench ou linha de comando:

bash

mysql -h [RDS_ENDPOINT] -P 3306 -u admin -p

3.4 Execute os scripts do repositório salontime-banco-dados:

SQL

3.4.1 Criar estrutura do banco

source criacao_bd.sql

3.4.2 Inserir dados iniciais

source inserts_cenario01.sql

3.4.3 Verifique a criação:

SQL

USE SalonTime;

SHOW TABLES;

## 4. CONFIGURAÇÃO DA VPC E SECURITY GROUPS

4.1 **Configurar VPC**

Code

VPC:

- Name: salontime-vpc

- IPv4 CIDR: 10.0.0.0/16

Subnets:

- Public Subnet 1: 10.0.1.0/24 (us-east-1a)

- Public Subnet 2: 10.0.2.0/24 (us-east-1b)

- Private Subnet 1: 10.0.10.0/24 (us-east-1a)

- Private Subnet 2: 10.0.11.0/24 (us-east-1b)

Internet Gateway:

- Name: salontime-igw

- Attach to VPC

4.2 **Criar Security Groups**

**Security Group para ALB:**

Code

Name: salontime-alb-sg

Inbound rules:

- Type: HTTP (80), Source: 0.0.0.0/0

- Type: HTTPS (443), Source: 0.0.0.0/0

**Security Group para EC2:**

Code

Name: salontime-ec2-sg

Inbound rules:

- Type: Custom TCP, Port: 3000, Source: salontime-alb-sg (React)

- Type: Custom TCP, Port: 8080, Source: salontime-alb-sg (Kotlin API)

- Type: Custom TCP, Port: 3001, Source: salontime-alb-sg (BotPress)

- Type: SSH (22), Source: [Seu IP]

**Security Group para RDS:**

Code

Name: salontime-rds-sg

Inbound rules:

- Type: MySQL/Aurora (3306), Source: salontime-ec2-sg

# 5. CONFIGURAÇÃO DAS INSTÂNCIAS EC2

## 5.1 Criar Instância EC2 - Client-side Web Application (React)

5.1.1 **Launch Instance**:

Code

Name: salontime-frontend

AMI: Amazon Linux 2023 AMI

Instance type: t3.small

Key pair: [Criar ou selecionar key pair existente]

Network settings:

- VPC: salontime-vpc

- Subnet: Public Subnet 1

- Auto-assign public IP: Enable

- Security group: salontime-ec2-sg

5.1.2 **User Data** (Advanced details):

bash

#!/bin/bash

# Atualizar sistema

yum update -y

# Instalar Node.js 18.x

curl -sL https://rpm.nodesource.com/setup_18.x | bash -

yum install -y nodejs git

# Criar diretório da aplicação

mkdir -p /home/ec2-user/app

cd /home/ec2-user/app

# Clonar repositório

git clone https://github.com/Grupo1-Semestre3/salontime-front-end-react.git .

# Instalar dependências

npm install

npm install axios moment chart.js react-chartjs-2 react-big-calendar sweetalert2 tailwindcss @tailwindcss/vite

# Configurar variáveis de ambiente

cat > .env << EOF

VITE_API_URL=http://[ALB_DNS_NAME]/api

EOF

# Build da aplicação

npm run build

### 5.2 Criar Instância EC2 - API Application (Spring Boot Kotlin)

5.2.1 **Launch Instance**:

Code

Name: salontime-api

AMI: Amazon Linux 2023 AMI

Instance type: t3.medium

Key pair: [Mesma key pair]

Network settings:

- VPC: salontime-vpc

- Subnet: Public Subnet 1

- Auto-assign public IP: Enable

- Security group: salontime-ec2-sg

5.2.2 **User Data**:

bash

#!/bin/bash

```
# Atualizar sistema
yum update -y

# Instalar Java 17
yum install -y java-17-amazon-corretto-devel maven git

# Criar diretório da aplicação
mkdir -p /home/ec2-user/app
cd /home/ec2-user/app

# Clonar repositório
git clone https://github.com/Grupo1-Semestre3/salontime-app-kotlin.git .
cd salonTime

# Configurar application.properties
cat > src/main/resources/application.properties << EOF
spring.application.name=SalonTime
server.port=8080

# Database Configuration
spring.datasource.url=jdbc:mysql://[RDS_ENDPOINT]:3306/SalonTime
spring.datasource.username=admin
spring.datasource.password=[RDS_PASSWORD]
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# JPA Configuration
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

```
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

# Mail Configuration (configure conforme necessário)
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=[EMAIL]
spring.mail.password=[APP_PASSWORD]
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
EOF

# Build da aplicação
mvn clean package -DskipTests

# Criar serviço systemd
cat > /etc/systemd/system/salontime-api.service << SERVICE_EOF
[Unit]
Description=SalonTime API
After=network.target

[Service]
Type=simple
User=ec2-user
WorkingDirectory=/home/ec2-user/app/salonTime
ExecStart=/usr/bin/java -jar target/salonTime-0.0.1-SNAPSHOT.jar
Restart=always

[Install]
```

WantedBy=multi-user.target

SERVICE_EOF

# Iniciar serviço

systemctl daemon-reload

systemctl start salontime-api

systemctl enable salontime-api

### 5.2 Criar Instância EC2 - BotPress

#### 5.2.1 **Launch Instance**:

Code

Name: salontime-botpress

AMI: Amazon Linux 2023 AMI

Instance type: t3.small

Key pair: [Mesma key pair]

Network settings:

- VPC: salontime-vpc

- Subnet: Public Subnet 1

- Auto-assign public IP: Enable

- Security group: salontime-ec2-sg

#### 5.2.2 **User Data**:

bash

#!/bin/bash

# Atualizar sistema

yum update -y

# Instalar Node.js e dependências

curl -sL https://rpm.nodesource.com/setup_18.x | bash -

yum install -y nodejs git unzip

```
# Criar diretório da aplicação
mkdir -p /home/ec2-user/botpress
cd /home/ec2-user/botpress

# Clonar repositório
git clone https://github.com/Grupo1-Semestre3/salontime-bot-atendimento.git .

# Extrair arquivo do bot
if [ -f bot_salnotimeatendimento_1750276326410.tgz ]; then
   tar -xzf bot_salnotimeatendimento_1750276326410.tgz
fi

# Instalar Botpress (versão mais recente)
wget https://s3.amazonaws.com/botpress-binaries/botpress-v12_latest-linux-x64.zip
unzip botpress-v12_latest-linux-x64.zip
chmod +x bp

# Configurar variáveis de ambiente
cat > .env << EOF
PORT=3001
DATABASE_URL=postgres://[Se necessário]
API_URL=http://[API_PRIVATE_IP]:8080
EOF

# Criar serviço systemd
cat > /etc/systemd/system/botpress.service << SERVICE_EOF
```

[Unit]

Description=Botpress Server

After=network.target

[Service]

Type=simple

User=ec2-user

WorkingDirectory=/home/ec2-user/botpress

ExecStart=/home/ec2-user/botpress/bp

Restart=always

[Install]

WantedBy=multi-user.target

SERVICE_EOF

# Iniciar serviço

systemctl daemon-reload

systemctl start botpress

systemctl enable botpress

## 6. CONFIGURAÇÃO DO ELASTIC LOAD BALANCER (ALB)

### 6.1 **Criar Application Load Balancer**

**EC2 Console → Load Balancers → Create Load Balancer → Application Load Balancer**

Code

Name: salontime-alb

Scheme: Internet-facing

IP address type: IPv4

Network mapping:

- VPC: salontime-vpc

- Availability Zones:

  - us-east-1a: Public Subnet 1

  - us-east-1b: Public Subnet 2

Security groups:

- salontime-alb-sg

6.2 **Criar Target Groups**

**Target Group 1 - Frontend:**

Code

Name: salontime-frontend-tg

Target type: Instances

Protocol: HTTP

Port: 3000

VPC: salontime-vpc

Health check:

- Protocol: HTTP

- Path: /

- Healthy threshold: 2

- Interval: 30 seconds

Register targets:

- Selecionar: salontime-frontend

**Target Group 2 - API:**

Code

Name: salontime-api-tg

Target type: Instances

Protocol: HTTP

Port: 8080

VPC: salontime-vpc

Health check:

- Protocol: HTTP

- Path: /actuator/health (ou /api/health se implementado)

- Healthy threshold: 2

- Interval: 30 seconds

Register targets:

- Selecionar: salontime-api

**Target Group 3 - BotPress:**

Code

Name: salontime-botpress-tg

Target type: Instances

Protocol: HTTP

Port: 3001

VPC: salontime-vpc

Health check:

- Protocol: HTTP

- Path: /

- Healthy threshold: 2

- Interval: 30 seconds

Register targets:

- Selecionar: salontime-botpress

### 6.3 **Configurar Listeners e Rules**

**Listener HTTP:80:**

Code

Default action: Forward to salontime-frontend-tg

Rules:

1. Path: /api/*

   Action: Forward to salontime-api-tg

2. Path: /bot/*

   Action: Forward to salontime-botpress-tg

3. Default

   Action: Forward to salontime-frontend-tg

## 7. CONFIGURAÇÃO DO AWS WAF

### 7.1 **Criar Web ACL**

**WAF & Shield Console → Create web ACL**

Code

Name: salontime-waf

Resource type: Application Load Balancer

Region: [Sua região]

Associated AWS resources:

- Add AWS resources: Selecionar salontime-alb

### 7.2 **Adicionar Rules**

**Rule 1 - Rate Limiting:**

Code

Name: RateLimitRule

Type: Rate-based rule

Rate limit: 2000 requests per 5 minutes

Action: Block

**Rule 2 - AWS Managed Rules - Core Rule Set:**

Code

Name: AWSManagedRulesCommonRuleSet

Rule group: AWS managed rule groups

- Core rule set (CRS)

Action: Block

**Rule 3 - AWS Managed Rules - Known Bad Inputs:**

Code

Name: AWSManagedRulesKnownBadInputsRuleSet

Rule group: AWS managed rule groups

- Known bad inputs

Action: Block

**Rule 4 - SQL Injection Protection:**

Code

Name: SQLInjectionProtection

Rule group: AWS managed rule groups

- SQL database

Action: Block

**Rule 5 - Geographic Restriction (Opcional):**

Code

Name: GeoBlockRule

Type: Rule builder

- If: Geographic match → Countries → [Selecionar países bloqueados]

- Then: Block

7.3 **Configurar Logging (Opcional mas Recomendado)**

Code

Logging destination: S3 bucket

Bucket name: salontime-waf-logs

Log retention: 30 days

---

## 8. VALIDAÇÃO E TESTES

8.1 **Verificar Status dos Componentes**

**RDS:**

bash

# Testar conexão

mysql -h [RDS_ENDPOINT] -P 3306 -u admin -p -e "SELECT 1;"

**EC2 Instances:**

bash

# Conectar via SSH

ssh -i [KEY_FILE].pem ec2-user@[INSTANCE_IP]

# Frontend - Verificar NGINX

sudo systemctl status nginx

curl http://localhost:3000

# API - Verificar Spring Boot

sudo systemctl status salontime-api

curl http://localhost:8080/actuator/health

# BotPress - Verificar serviço

sudo systemctl status botpress

curl http://localhost:3001

**Load Balancer:**

Code

AWS Console → EC2 → Load Balancers → salontime-alb

- State: Active

- Verificar Target Groups: Todos os targets "healthy"

**WAF:**

Code

AWS Console → WAF & Shield → Web ACLs → salontime-waf

- Verificar métricas de requests

- Testar regras de bloqueio

### 8.2 **Testar Aplicação**

#### 8.2.1 **Acessar Frontend:**

Code

http://[ALB_DNS_NAME]

**Testar API:**

bash

curl http://[ALB_DNS_NAME]/api/[endpoint]

#### 8.2.2 **Testar Bot:**

Code

http://[ALB_DNS_NAME]/bot

#### 8.2.3 **Testes de Segurança**

bash

# Testar rate limiting (deve bloquear após exceder limite)

for i in {1..3000}; do curl http://[ALB_DNS_NAME]; done

# Testar SQL Injection (deve ser bloqueado pelo WAF)

curl "http://[ALB_DNS_NAME]/api/users?id=1' OR '1'='1"

## 9. TROUBLESHOOTING

### 9.1 **Problema: Target Unhealthy no ALB**

Code

- Verificar Security Groups

- Verificar Health Check paths

- Validar que aplicações estão rodando nas portas corretas

### 9.2 **Problema: Erro de Conexão com RDS**

Code

- Verificar Security Group do RDS permite conexões do EC2 SG

- Validar endpoint e credenciais no application.properties

- Verificar se RDS está na mesma VPC

### 9.3 **Problema: WAF Bloqueando Requisições Legítimas**

Code

- Analisar logs do WAF

- Ajustar rules ou adicionar exceções (whitelist)

- Revisar rate limiting thresholds

## 10. Indicadores de Prontidão

- RDS criado e acessível
- Scripts SQL executados com sucesso
- 3 instâncias EC2 criadas e rodando aplicações
- Security Groups configurados corretamente
- Target Groups com targets "healthy"
- ALB criado e distribuindo tráfego
- WAF configurado e protegendo ALB
- Aplicação acessível via ALB DNS
- CloudWatch alarmes configurados
- Logs habilitados
- Documentação de credenciais guardada

**Manual criado com base na arquitetura de implantação do Sistema SalonTime**
**Data: 17/11/2025**