Plano de Ação: Do Código à Nuvem

Este guia irá te levar desde a configuração inicial do projeto em sua máquina até a implantação e execução na AWS.

Fase 1: Preparação do Ambiente

1. **Configurar o Projeto Node.js:**
    - Em seu terminal, crie a pasta do projeto e navegue até ela: mkdir bitcoin-alerta && cd bitcoin-alerta.
    - Inicie o projeto Node.js: npm init -y.
    - Instale as dependências que usaremos: npm install axios twilio.
2. **Criar os Arquivos Essenciais:**
    - Crie os arquivos de código conforme a estrutura que definimos:
        - src/handler.js
        - src/price-checker.js
        - src/notifier.js
    - Crie um arquivo para suas variáveis de ambiente na raiz do projeto: .env. Você vai colocar suas chaves da Twilio e o preço-alvo aqui, para mantê-los seguros.
3. **Configurar as Contas:**
    - Crie uma conta gratuita na **AWS**.
    - Crie uma conta gratuita na **Twilio** e obtenha suas credenciais (Account SID, Auth Token e o número de telefone de teste).

Fase 2: Desenvolvimento do Código

1. **Lógica do Checador de Preço (src/price-checker.js):**
    - Crie uma função que faça uma chamada GET usando axios para a API da CoinGecko.
    - Essa função deve retornar o preço do Bitcoin.
2. **Lógica do Notificador (src/notifier.js):**
    - Crie uma função que receba uma mensagem como parâmetro.
    - Use o cliente da **Twilio** para enviar a mensagem para o seu número de telefone, usando as credenciais do seu arquivo .env.
3. **O Orquestrador (src/handler.js):**
    - Este será o ponto de entrada da sua função Lambda.

- Importe suas funções price-checker e notifier.
- Defina as variáveis de ambiente para o preço-alvo de compra e venda.
- Crie a função exports.handler que será executada pela AWS. Dentro dela:
  - Chame a função price-checker para obter o preço atual.
  - Use uma estrutura condicional (if/else if) para comparar o preço atual com os preços-alvo.
  - Se uma condição for atendida, chame a função notifier com a mensagem de alerta apropriada.

Fase 3: Implantação e Configuração na AWS

1. **Compactar o Código:**
   - Antes de enviar para a AWS, você precisa compactar sua pasta src e a pasta node_modules em um único arquivo ZIP. Isso inclui todas as suas dependências.
2. **Criar a Função Lambda:**
   - No console da AWS, vá para o serviço **Lambda** e clique em "Create function".
   - Escolha "Author from scratch", dê um nome ao seu projeto e selecione "Node.js 18.x" (ou a versão mais recente).
3. **Configurar Variáveis de Ambiente:**
   - Na página da sua função Lambda, vá para a aba "Configuration" e depois em "Environment variables".
   - Adicione suas chaves da Twilio e seus preços-alvo (por exemplo, TWILIO_ACCOUNT_SID, TWILIO_AUTH_TOKEN, ALVO_COMPRA_BTC, etc.).
4. **Upload do Código:**
   - Na aba "Code", clique em "Upload from" e selecione o arquivo ZIP que você criou.
5. **Configurar o Gatilho (Trigger) com EventBridge:**
   - Na sua função Lambda, clique em "Add trigger".
   - Selecione **EventBridge (CloudWatch Events)**.
   - Crie uma nova regra e escolha "Schedule expression".
   - Use a expressão rate(15 seconds) para garantir que sua função seja executada a cada 15 segundos.

Fase 4: Testes e Validação

1. **Teste Manual:**
   - Vá na aba "Test" da sua função Lambda e clique em "Test" para executar a função manualmente. Verifique os logs (em CloudWatch) para ver se a chamada à API funcionou corretamente.

2. **Teste de Notificação:**
    - Aguarde a execução automática do EventBridge para ver se a notificação por SMS é enviada quando o preço se aproxima ou atinge seu alvo.

Boa sorte! Siga este plano e você terá seu projeto rodando em pouco tempo. Lembre-se de documentar cada etapa para o seu portfólio.