Documentação Técnica: Sistema de Geração de Histórias Personalizadas no Universo Star Wars com AWS e LLM

(Versão 01)

Introdução

Eu escolhi 2 serviços em nuvem para garantir segurança, rapidez e todos os benefícios do processamento em cloud. Optei pela AWS porque foi a indicação pedida no email que recebi.

Objetivo do Projeto

O objetivo principal deste projeto é criar um modelo LLM capaz de gerar narrativas personalizadas com base nas preferências do usuário. Para garantir que as histórias sejam coerentes e dentro do universo Star Wars, o sistema:

- Utiliza Python como linguagem principal de desenvolvimento.
- Obtém dados a partir da API do Star Wars (SWAPI), fornecendo informações sobre personagens, planetas e naves.
- Gera narrativas personalizadas a partir de prompts detalhados, limitando o escopo do modelo à criação de histórias.
- Cria um endpoint na AWS que permite aos usuários consultar as histórias geradas enviando informações sobre personagens, planetas e naves no corpo da requisição.
- Utiliza os serviços AWS Lambda e API Gateway para garantir escalabilidade e flexibilidade na solução.

Segurança: Uso de Arquivo `.env`

Para evitar que credenciais e chaves de API fiquem expostas no script, eu utilizei um arquivo .env. Esse arquivo armazena informações sensíveis e é carregado pelo script usando a biblioteca dotenv. Dessa forma, as credenciais não são incluídas diretamente no código-fonte, aumentando a segurança e facilitando a gestão de configurações em diferentes ambientes.

Estrutura do Script

1. **Carregamento das Variáveis de Ambiente**
    - Eu importei a biblioteca dotenv para carregar as credenciais do arquivo .env.
    - Essa abordagem evita expor credenciais no repositório de código.

2. **Integração com a API do Star Wars (SWAPI)**
    - Eu utilizei a SWAPI para obter dados sobre personagens, espaçonaves e planetas do universo Star Wars.
    - Faço requisições HTTP POST para buscar essas informações e enriquecer a geração de histórias.

3. **Uso de um Modelo LLM (Large Language Model)**
    - O modelo LLM que escolhi é responsável por criar histórias personalizadas baseadas nos dados recuperados e nas preferências do usuário.
    - Para garantir que o modelo crie narrativas relevantes, elaborei prompts detalhados que limitam o escopo da geração de conteúdo.
    - Escolhi o LLM da OpenAI devido à facilidade de uso da infraestrutura, ao custo mais acessível e ao tempo disponível para desenvolvimento.
    - Optei pelo GPT-3.5 Turbo pois ele oferece um bom equilíbrio entre custo e desempenho, garantindo um serviço eficiente e escalável.

4. **Construção da Lógica de Geração da História**
    - Eu seleciono um conjunto de personagens, espaçonaves e eventos aleatórios da SWAPI com base nos dados enviados pelo usuário.
    - Construo um prompt estruturado para o modelo LLM, garantindo que a história siga uma estrutura lógica e atenda às preferências fornecidas.

5. **Criação da Role e Permissões para AWS Lambda**
    - Para que a AWS Lambda possa ser executada corretamente, criei uma role IAM específica, garantindo que o serviço tenha as permissões necessárias.
    - A role criada define que a Lambda pode ser assumida apenas pelo serviço lambda.amazonaws.com, evitando acessos indevidos.
    - As permissões concedidas seguem o conceito de privilégios mínimos, garantindo que a Lambda tenha acesso apenas ao necessário para executar suas funções.
    - As permissões incluem apenas:
        - A capacidade de gravar logs no Amazon CloudWatch (logs:CreateLogGroup, logs:CreateLogStream, logs:PutLogEvents)
        - A permissão para invocar outras funções Lambda, caso necessário (lambda:InvokeFunction)
        - A autorização para ser invocada pelo API Gateway (lambda:InvokeFunction por apigateway.amazonaws.com)
    - Essa abordagem melhora a segurança, reduzindo a superfície de ataque e minimizando riscos de escalonamento de privilégios.

6. **Disponibilização via AWS Lambda e API Gateway**
    - Estruturei o script para ser executado em um endpoint serverless usando AWS Lambda.
    - O API Gateway permite que qualquer usuário faça requisições HTTP e receba histórias geradas dinamicamente.
    - Criei um recurso (Resource) no API Gateway para estruturar corretamente as chamadas à API.
    - O método escolhido foi HTTP POST, pois as solicitações envolvem envio de dados para processamento, permitindo que o usuário personalize a história.
    - A integração entre Lambda e API Gateway foi feita via proxy, permitindo que o API Gateway encaminhe diretamente as solicitações e respostas.
    - Essa integração de proxy proporciona flexibilidade, pois a implementação da função Lambda pode ser alterada a qualquer momento sem a necessidade de redeploy da API.
    - Além disso, a integração suporta múltiplas linguagens compatíveis com o AWS Lambda, garantindo maior liberdade de desenvolvimento.
    - Isso garante escalabilidade e baixa latência no acesso ao serviço.

7. **Pacote de Dependências para AWS Lambda**
    - A AWS Lambda não possui um ambiente pré-configurado com bibliotecas e dependências.
    - Para garantir o funcionamento adequado, eu incluí todas as dependências necessárias dentro do arquivo ZIP que será enviado para a Lambda.
    - Isso evita falhas de execução e garante que todas as bibliotecas utilizadas estejam disponíveis no ambiente da Lambda.

8. **Tratamento de Erros e Logs**
    - O monitoramento e a depuração dos erros são feitos automaticamente pelo Amazon CloudWatch.
    - A AWS Lambda registra automaticamente logs no CloudWatch sempre que a função é executada, armazenando detalhes sobre cada invocação.
    - Os logs incluem informações sobre a execução da função, erros gerados e métricas relevantes.
    - Para facilitar a análise, é possível acessar esses logs pelo AWS Management Console ou utilizar consultas no CloudWatch Logs Insights.
    - Isso permite identificar rapidamente falhas na execução e otimizar o desempenho da aplicação.

Importância de Cada Passo

Cada etapa do script foi projetada para garantir um serviço eficiente, seguro e escalável. O uso do .env protege credenciais sensíveis, enquanto a SWAPI fornece dados para criação de histórias realistas e personalizadas conforme as preferências do usuário. A integração com AWS Lambda e API Gateway permite que o serviço seja acessado globalmente com alta disponibilidade. A aplicação do conceito de

privilégios mínimos nas permissões garante segurança no acesso, restringindo o que a Lambda pode fazer para evitar riscos desnecessários. O empacotamento das dependências dentro do ZIP evita problemas na execução da Lambda. O uso de prompts detalhados melhora a qualidade das histórias geradas pelo modelo LLM, garantindo que atendam às expectativas dos usuários. A integração de proxy entre Lambda e API Gateway proporciona flexibilidade ao permitir alterações na Lambda sem necessidade de reimplantação da API.

Conclusão

O script que desenvolvi representa uma solução completa para a geração de histórias no universo Star Wars, utilizando IA generativa, dados autênticos da SWAPI e infraestrutura escalável na AWS. A implementação segue boas práticas de segurança e desenvolvimento, garantindo um serviço confiável e de alta qualidade. O endpoint criado permite que o usuário personalize as histórias enviando informações sobre personagens, planetas e naves, tornando a experiência mais imersiva e envolvente.

Autora: Jaqueline Jardim