웅진 IT

Databricks GenAI 애플리케이션 개발 가이드 (For Beginners)

Databricks 활용 가이드

박형식

2025-6-30

목차

Databricks GenAI 애플리케이션 개발 가이드 (For Beginners)

문서 최종 업데이트: 2025년 6월 27일 (v2.0 - VSCode 연동 개발 환경 상세 가이드 추가)

들어가며

이 문서는 Databricks를 처음 사용하는 초보자를 대상으로 작성되었습니다. Databricks의 기본 개념 이해부터 실제 생성형 AI(Generative AI) 기반의 RAG(검색 증강 생성) 애플리케이션을 직접 구축하는 전 과정을 단계별로 안내합니다. 이 가이드만 따라오시면 데이터 준비부터 AI 모델 연동, 효율적인 개발 워크플로우 구축까지 완벽하게 파악할 수 있습니다.

Chapter 1: Databricks란 무엇인가?

Databricks는 데이터 저장, 가공, 분석, 머신러닝, AI 모델 개발 등 데이터와 관련된 모든 작업을 **하나의 통합된 플랫폼**에서 수행할 수 있도록 해주는 클라우드 기반 서비스입니다.

- **핵심 컨셉: "레이크하우스 (Lakehouse)"**
    - **데이터 레이크(Lake):** 모든 종류의 원본 데이터를 저렴하게 저장하는 '호수'. 유연하지만 데이터 관리가 복잡합니다.
    - **데이터 웨어하우스(Warehouse):** 정제된 데이터를 빠르고 안정적으로 분석하는 '창고'. 신뢰성이 높지만 비싸고 비정형 데이터를 다루기 어렵습니다.
    - **레이크하우스(Lakehouse):** 이 둘의 장점만을 결합한 아키텍처입니다. Databricks는 데이터 레이크 위에 웨어하우스의 안정성과 성능을 구현하여, 모든 데이터를 한 곳에서 관리하고 AI 모델 개발까지 연결할 수 있게 해줍니다.

Chapter 2: 시작하기 - 무료 체험 환경 준비

Databricks는 14일간 모든 기능을 사용해볼 수 있는 무료 평가판(Free Trial)을 제공합니다.

1. **Databricks 평가판 신청 페이지**에 접속하여 가입을 진행합니다.
2. AWS, Azure, GCP 중 본인이 사용하는 클라우드 계정을 선택하여 연동합니다.

🚨 매우 중요한 비용 절약 Tip

- **클라우드 비용:** Databricks 평가판 자체는 무료지만, 내부적으로 사용하는 클라우드 인프라(가상머신, 스토리지) 비용은 발생할 수 있습니다. (대부분의 클라우드는 신규 가입 시 제공하는 무료 사용량(Free Tier) 내에서 해결 가능합니다.)
- **클러스터 자동 종료 설정:** 비용 발생의 주원인은 '클러스터(분석용 컴퓨터)'입니다. 클러스터를 생성할 때 **'Terminate after ___ minutes of inactivity'(비활성 상태일 때 자동 종료)** 옵션을 **30분**과 같이 짧게 설정하세요. 사용하지 않을 때 자동으로 꺼져 불필요한 비용을 90% 이상 막을 수 있습니다.

Chapter 3: RAG 애플리케이션 구축 실습

이제 PDF 파일을 기반으로 질문에 답변하는 RAG 챗봇을 직접 만들어 보겠습니다.

1단계: 데이터 준비 및 Delta Table 생성

RAG의 핵심은 AI가 참조할 데이터를 잘 준비하는 것입니다. Databricks에서는 이 데이터를 **Delta Table**로 관리하는 것이 가장 좋습니다.

2단계: 벡터 스토어 생성 (Databricks Vector Search)

준비된 Delta Table을 기반으로 텍스트를 벡터로 변환하고 저장할 공간(벡터 스토어)을 만듭니다.

1. **Vector Search 엔드포인트 생성 (UI, 최초 한 번)**
    - 왼쪽 메뉴 컴퓨트(Compute) > Vector Search 탭으로 이동합니다.
    - Create endpoint를 클릭하여 my-rag-endpoint와 같은 이름으로 엔드포인트를 생성합니다.
2. **Vector Search 인덱스 생성 (Code)**
    - 아래 코드를 실행하여 Delta Table과 인덱스를 동기화합니다.

```
Python

# 0. 필요한 라이브러리 설치

%pip install --upgrade databricks-vectorsearch



# 1. 인덱스 생성

from databricks.vector_search.client import VectorSearchClient

vsc = VectorSearchClient()



vsc.create_delta_sync_index(

    endpoint_name="my-rag-endpoint", # 위에서 만든 엔드포인트 이름

    source_table_name="main.default.rag_source_documents", # 원본 델타 테이블

    index_name="main.default.rag_document_index", # 생성할 인덱스 이름

    pipeline_type="TRIGGERED",
```

```
    primary_key="doc_id",
    embedding_source_column="text_chunk",
    embedding_model_endpoint_name="databricks-bge-large-en" # Databricks 기본 임베딩 모델
)
```

3단계: RAG 체인(Chain) 구축 및 LLM 연동

이제 모든 조각을 LangChain으로 연결합니다. LLM은 여러 옵션 중 선택할 수 있습니다.

Option A (권장): Databricks 네이티브 모델 사용

```
Python
# 필요한 라이브러리 임포트
from langchain_community.chat_models import ChatDatabricks
from langchain_community.vectorstores import DatabricksVectorSearch
from langchain_community.embeddings import DatabricksEmbeddings
from langchain.chains import RetrievalQA

# 1. 벡터 스토어에 연결하여 '검색기(Retriever)' 생성
vector_store = DatabricksVectorSearch(
    endpoint_name="my-rag-endpoint",
    index_name="main.default.rag_document_index",
    embedding=DatabricksEmbeddings(endpoint="databricks-bge-large-en")
)
```

```
retriever = vector_store.as_retriever()



# 2. Databricks 의 기본 LLM 모델 설정

llm = ChatDatabricks(endpoint="databricks-dbrx-instruct", max_tokens=200)



# 3. RAG 체인 조립 및 실행

qa_chain = RetrievalQA.from_chain_type(llm=llm, chain_type="stuff",
retriever=retriever)

response = qa_chain.invoke({"query": "이 문서의 핵심 내용은 무엇인가요?"})

print(response["result"])
```

Option B: Azure OpenAI Service 연동

Databricks 외부의 Azure OpenAI 모델을 사용하고 싶다면 아래와 같이 LLM과 Embedding 부분을 설정합니다. (※ langchain-openai, openai 패키지 설치 필요)

```
Python

from langchain_openai import AzureChatOpenAI, AzureOpenAIEmbeddings



# Azure Portal 에서 확인한 정보 (Databricks Secrets 사용 권장)

api_key = dbutils.secrets.get(scope="azure-openai", key="api-key")

azure_endpoint = dbutils.secrets.get(scope="azure-openai", key="endpoint")

api_version = "2024-02-01"



# Azure OpenAI LLM 및 Embedding 설정
```

```
llm = AzureChatOpenAI(
    azure_endpoint=azure_endpoint,
    openai_api_key=api_key,
    openai_api_version=api_version,
    azure_deployment="my-gpt4-deployment" # Azure 에서 배포한 모델 이름
)
# embeddings = AzureOpenAIEmbeddings(...) # 임베딩 모델도 동일하게 설정
# ... 이후 과정은 동일 ...
```

Chapter 4: 개발 환경 구축 - 나에게 맞는 방법 선택하기

Databricks 코드를 작성하고 실행하는 방법은 크게 두 가지입니다. 각 방식의 장단점을 이해하고 자신에게 맞는 방법을 선택하세요.

4.1 Databricks 노트북 (Web UI)

Databricks 작업 공간에 접속하여 웹 브라우저에서 직접 코드를 작성하고 실행하는 가장 기본적인 방법입니다.

- **장점:**
    - 별도 설정이 전혀 필요 없는 제로-셋업(Zero-setup).
    - 시각화 결과나 데이터프레임 출력을 즉시 확인할 수 있어 탐색적 분석에 유리.
- **단점:**
    - VSCode와 같은 전문 에디터의 강력한 기능(자동완성, 리팩토링 등) 사용이 제한적.
    - Git을 통한 체계적인 코드 버전 관리가 다소 번거로울 수 있음.

4.2 로컬/VM + VSCode (전문가를 위한 강력한 선택)

많은 분들이 "VSCode에서 코드를 짜서 수동으로 업로드해야 하나?"라고 오해하지만, 이는 과거의 방식입니다. **Databricks 공식 확장 프로그램**을 사용하면 VSCode와 Databricks가 실시간으로 연동되어, 로컬 환경에서 원격의 강력한 Databricks 컴퓨팅 자원을 직접 제어하며 개발할 수 있습니다.

- **핵심 개념: "조종석과 엔진"**
    - **나의 컴퓨터(VM) + VSCode:** 코드를 작성하고 명령을 내리는 **'조종석'** 입니다.
    - **Databricks 작업 공간:** 실제 데이터 처리와 연산이 일어나는 **'강력한 엔진'** 입니다.
    - VSCode라는 '조종석'에서 실행 버튼을 누르면, 명령이 즉시 '엔진'으로 전달되어 실행되고 그 결과가 '조종석의 계기판(VSCode 터미널)'에 실시간으로 나타납니다.
- **Step-by-Step: VSCode 연동 환경 구축하기**

1단계: 사전 준비 - Databricks 액세스 토큰 발급

- Databricks 웹 UI 우측 상단의 사용자 아이콘 클릭 > User Settings > Developer 탭으로 이동합니다.
- Access tokens 섹션에서 Generate new token을 클릭하여 토큰을 생성합니다.
- 🚨 **생성된 토큰은 딱 한 번만 보이므로 반드시 안전한 곳에 복사해 두세요.**

2단계: 개발 환경 설정 - Databricks CLI 설치 및 구성

- VSCode를 사용할 컴퓨터(VM)의 터미널에서 아래 명령어로 CLI를 설치합니다.

```
Bash

pip install databricks-cli
```

- 아래 명령어로 Databricks 작업 공간 정보를 설정합니다.

```
Bash

databricks configure --token
```

  - Databricks Host: https://....cloud.databricks.com 형태의 작업 공간 URL을 입력합니다.
  - Token: 1단계에서 복사해 둔 액세스 토큰을 붙여넣습니다.

3단계: VSCode 설정 - 공식 확장 프로그램 설치

- VSCode의 확장 프로그램 마켓플레이스(Extensions Marketplace)에서 Databricks를 검색하여 **Microsoft에서 제작한 공식 확장 프로그램**을 설치합니다.

4단계: VSCode와 Databricks 연결 및 원격 실행

- VSCode를 열고 Ctrl+Shift+P를 눌러 명령 팔레트를 엽니다.
- Databricks: Configure를 검색하여 프로필을 설정합니다. (CLI 설정이 되어있다면 자동으로 인식됩니다.)
- VSCode 좌측의 Databricks 아이콘을 클릭하면, 작업 공간의 클러스터 목록과 파일들을 볼 수 있습니다. 여기서 사용할 **클러스터를 선택하고 실행(Start)**시킵니다.
- Python 파일(.py)을 열고 코드 상단에 나타나는 **Run on Databricks** 또는 우측 상단의 실행 버튼을 누릅니다.
- 코드가 즉시 원격의 Databricks 클러스터에서 실행되고, 그 결과가 VSCode 하단의 **터미널(Terminal) 창**에 나타나는 것을 확인할 수 있습니다. **수동 업로드 과정은 전혀 없습니다.**

- **VSCode 방식의 장점:**
  - **최고의 생산성:** 손에 익은 VSCode의 모든 기능(단축키, 자동완성, 디버깅 등)을 활용.
  - **체계적인 버전 관리:** Git과 완벽하게 연동하여 코드를 전문적으로 관리.

- **원활한 피드백:** 코드를 수정하고 실행 결과를 즉시 VSCode 내에서 확인하는 빠른 개발 사이클.

Chapter 5: 패키지 관리 핵심 요약

Databricks 환경에서는 패키지 관리가 매우 간단합니다.

- **Rule 1 (확인부터):** Databricks ML 런타임에는 langchain, pandas 등 대부분의 패키지가 이미 설치되어 있습니다.
- **Rule 2 (네이티브 기능 우선):** 벡터 스토어는 chromadb 대신 databricks-vectorsearch를 사용하는 것이 성능과 보안, 관리에 훨씬 유리합니다.
- **Rule 3 (최소한만 설치): 가장 일반적인 설치 명령어는 아래와 같습니다.**

```
%pip install --upgrade langchain databricks-vectorsearch pypdf
```

  - --upgrade langchain: 최신 기능 사용을 위해 LangChain을 업그레이드.
  - databricks-vectorsearch: Databricks 네이티브 벡터 DB 사용을 위해.
  - pypdf: PDF 데이터 처리를 위해. (데이터 소스에 따라 변경)

마치며

이 가이드는 Databricks를 사용하여 GenAI 애플리케이션을 구축하는 표준적인 여정을 안내했습니다. Databricks의 진정한 힘은 데이터 준비부터 AI 모델 서빙까지 모든 과정이 끊김 없이 하나의 플랫폼에서 이루어진다는 점에 있습니다. 이 가이드를 시작으로 다양한 데이터를 활용하고, 더 복잡한 AI 에이전트를 만들어보는 등 무궁무진한 가능성을 탐험해 보시길 바랍니다.