# Comprehensive AI-Driven MLOps Command Reference Guide

AI-Driven Automating Banking Applications

## 1. Large Language Models (LLM)

LLMs power AI-driven automation, chatbots, and banking workflows.

from transformers import AutoModelForCausalLM, AutoTokenizer  
  
model\_name = "gpt-4"  
tokenizer = AutoTokenizer.from\_pretrained(model\_name)  
model = AutoModelForCausalLM.from\_pretrained(model\_name)  
  
input\_text = "How can AI automate banking transactions?"  
inputs = tokenizer(input\_text, return\_tensors="pt")  
output = model.generate(\*\*inputs)  
print(tokenizer.decode(output[0]))

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 2. Retrieval-Augmented Generation (RAG)

RAG improves AI responses by integrating external knowledge sources.

from langchain.document\_loaders import PyPDFLoader  
from langchain.vectorstores import FAISS  
from langchain.embeddings import OpenAIEmbeddings  
  
loader = PyPDFLoader("banking\_policies.pdf")  
docs = loader.load()  
  
vector\_store = FAISS.from\_documents(docs, OpenAIEmbeddings())  
retrieved\_docs = vector\_store.similarity\_search("What are fraud detection policies?")  
print(retrieved\_docs[0].page\_content)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 3. AI Agents

AI Agents autonomously handle financial workflows and customer queries.

from langchain.agents import initialize\_agent, AgentType  
from langchain.llms import OpenAI  
  
llm = OpenAI(model\_name="gpt-4")  
agent = initialize\_agent(llm=llm, agent=AgentType.ZERO\_SHOT\_REACT\_DESCRIPTION, verbose=True)  
  
response = agent.run("Check today's exchange rate for USD to GBP")  
print(response)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 4. Multi-Modality (Text, Image, Audio)

from transformers import BlipProcessor, BlipForConditionalGeneration  
  
processor = BlipProcessor.from\_pretrained("Salesforce/blip-image-captioning-base")  
model = BlipForConditionalGeneration.from\_pretrained("Salesforce/blip-image-captioning-base")  
  
image = Image.open("check\_signature.jpg")  
inputs = processor(images=image, return\_tensors="pt")  
output = model.generate(\*\*inputs)  
print(processor.decode(output[0]))

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 5. Fine-Tuning LLMs

from transformers import Trainer, TrainingArguments  
  
training\_args = TrainingArguments(output\_dir="./model\_output", per\_device\_train\_batch\_size=2, num\_train\_epochs=3)  
trainer = Trainer(model=model, args=training\_args, train\_dataset=banking\_data)  
trainer.train()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 6. Prompt Engineering

prompt = '''You are an AI banking assistant. Your job is to answer customer queries with utmost clarity.  
Question: {user\_input}'''  
formatted\_prompt = prompt.format(user\_input="How to apply for a mortgage loan?")  
print(formatted\_prompt)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 7. Microsoft Azure Commands

az login  
az account set --subscription "BankingAI"  
az aks create --resource-group AI\_Infra --name AKSCluster --node-count 3  
az storage account list

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 8. Python & Vector Databases (FAISS, ChromaDB)

from chromadb import Client  
  
client = Client()  
collection = client.create\_collection("banking\_faq")  
collection.add(texts=["What is KYC?", "How to apply for a credit card?"], ids=["1", "2"])  
result = collection.query("Tell me about KYC")  
print(result['documents'])

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 9. API Keys & Tokenization

import openai  
openai.api\_key = "your-api-key-here"  
  
def tokenize\_text(text):  
 return text.split(" ")  
  
tokens = tokenize\_text("Secure financial transactions are important.")  
print(tokens)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 10. Docker & Kubernetes for AI Models

docker build -t banking-llm .  
docker run -p 8080:8080 banking-llm  
kubectl apply -f ai-deployment.yaml  
kubectl get pods

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_