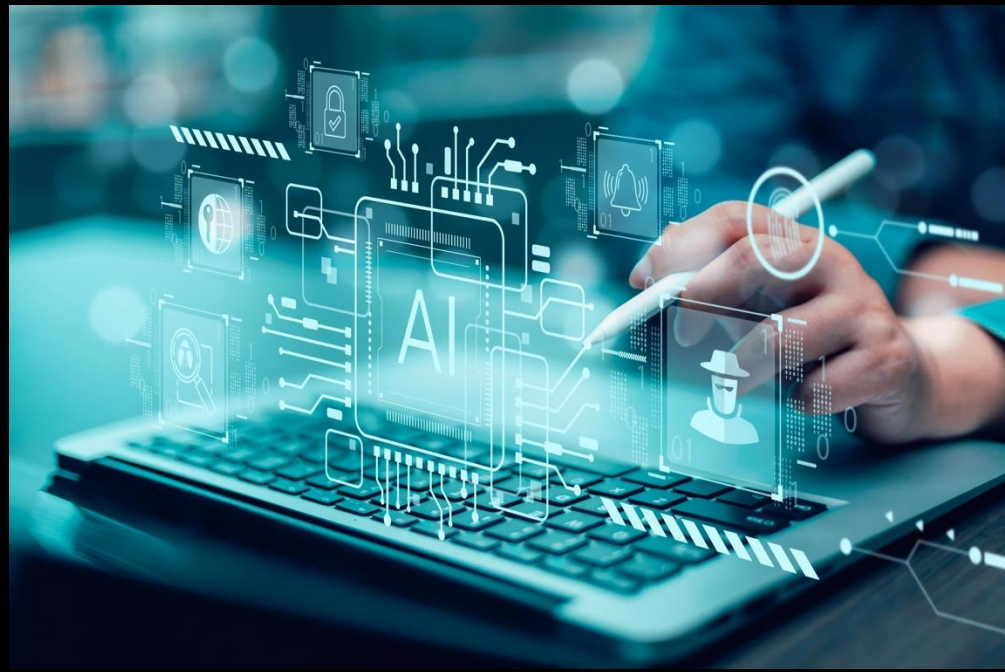
TATA CONSULTANCY SERVICES

TATA

AI Friday – Dry Run



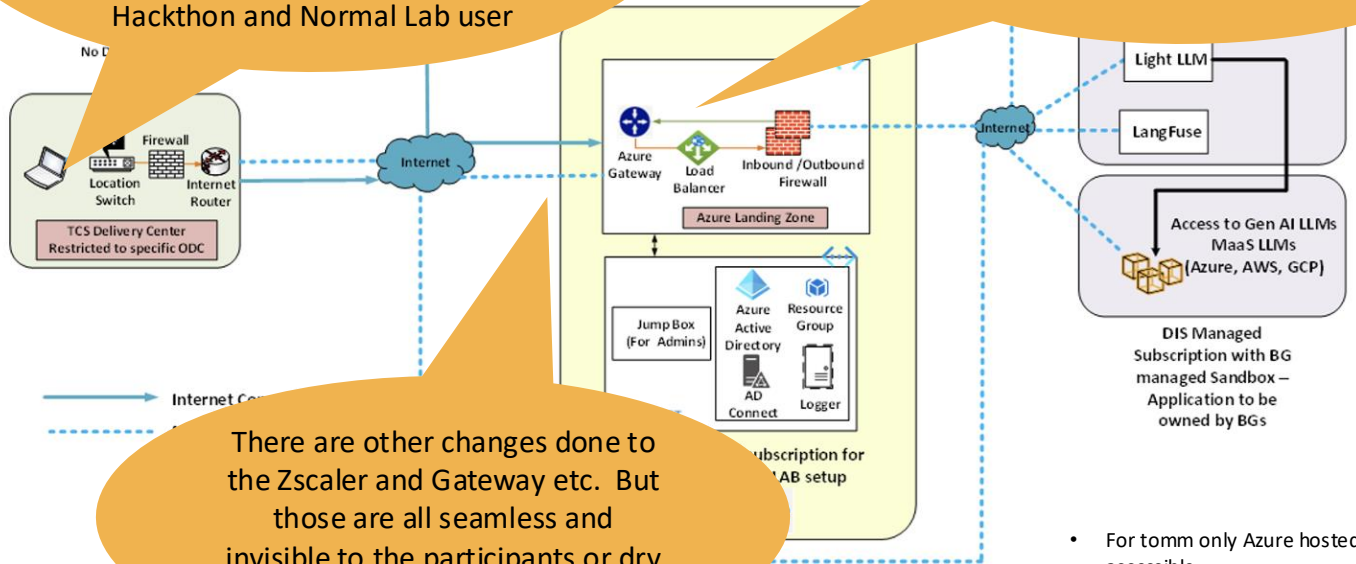
July 3, 2025

Building on belief

Changes done to the Environment – For your Understanding

Access to the LAPTOP is thru local id than a AD or domain id. Local DIS team will help you with creation of the Local ID. Local user id is also differentiated for two purposes – Hackthon and Normal Lab user

Team should be able to see a shared folder for the Hackathon or LAB based on id they are using.



There are other changes done to the Zscaler and Gateway etc. But those are all seamless and invisible to the participants or dry run team

- For tomm only Azure hosted Models will be accessible
- Models are listed in the next Page

High Level SOP for Validation of the Infra at each Location

- Participants will use laptops provided by the DIS team at the designated location.
- Laptops will be connected to a private network with pre-configured Zscaler policies for appropriate access.
- Internet access will be available through this policy but ensure it is used for events purpose.
- **User IDs and passwords will be made available thru the Local DIS team. Validate the same to login into the LAPTOP without any issues.**
- **Location DIS team will help in configuring the first time setup of Zscaler on these LAPTOP**
- **Laptops will have pre-installed software, including Visual Studio Code, Python, and Ollama**.
- A test program can be build (attached in this same pack). **Access to the Models are using the TCS Gateway, configured to work with Azure-hosted models.**
- Gateway URL: https://genailab.tcs.in
- **A shared folder is provisioned for team collaboration; validate with the local DIS team and Check for easy access to the folder.**

Sample code to do a test of the whole environment

```
!pip install langchain-openai
from langchain_openai import ChatOpenAI
import os
import httpx

client = httpx.Client(verify=False)

# os.environ['HTTP_PROXY']="http://proxy.tcs.com:8080"
# os.environ['HTTPS_PROXY']="http://proxy.tcs.com:8080"
# os.environ['http_proxy']="http://proxy.tcs.com:8080"
# os.environ['https_proxy']="http://proxy.tcs.com:8080"

llm = ChatOpenAI(
base_url="https://cin-genailab-maas-litellm-ca.victoriousground-d739afd7.centralindia.azurecontainerapps.io", # set
openai_api_base to the LiteLLM Proxy
base_url="https://genailab.tcs.in"
model = "azure_ai/genailab-maas-DeepSeek-V3-0324",
api_key="sk-h4SzToxOqOneSAXq191PXA",   Provided Key is for this purposes only and should not be used for any
unauthorized purposes
http_client = client
)

llm.invoke("Hi")
```

Sample code to test along with triggering of embedding models

```
import streamlit as st
from pdfminer.high_level import extract_text
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_openai import ChatOpenAI, OpenAIEmbeddings
from langchain_community.vectorstores import Chroma
from langchain.chains import RetrievalQA
import tempfile
import os
import httpx

import tiktoken
tiktoken_cache_dir = "./token"
os.environ["TIKTOKEN_CACHE_DIR"] = tiktoken_cache_dir
client = httpx.Client(verify=False)

# LLM and Embedding setup
llm = ChatOpenAI(
  base_url="https://genailab.tcs.in",
  model="azure_ai/genailab-maas-DeepSeek-V3-0324",
  api_key="sk-h4SzToxOqOneSAXq191PXA",
  http_client=client
)
embedding_model = OpenAIEmbeddings(
  base_url="https://genailab.tcs.in",
  model="azure/genailab-maas-text-embedding-3-large",
  api_key="sk-h4SzToxOqOneSAXq191PXA",
  http_client=client
)
st.set_page_config(page_title="RAG PDF Summarizer")
st.title("📄 RAG-powered PDF Summarizer")
upload_file = st.file_uploader("Upload a PDF", type="pdf")

if upload_file:
  with tempfile.NamedTemporaryFile(delete=False, suffix=".pdf") as temp_file:
    temp_file.write(upload_file.read())
    temp_file_path = temp_file.name
  # Step 1: Extract text
  raw_text = extract_text(temp_file_path)
  # Step 2: Chunking
  text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
  chunks = text_splitter.split_text(raw_text)
  # Step 3: Embed and store in Chroma
  with st.spinner("Indexing document..."):
    vectordb = Chroma.from_texts(chunks, embedding_model,
persist_directory="./chroma_index")
    vectordb.persist()
  # Step 4: RAG QA Chain
  #retriever = vectordb.as_retriever(search_type="similarity", search_kwargs={"k": 5})
  retriever = vectordb.as_retriever()
  rag_chain = RetrievalQA.from_chain_type(
    llm=llm,
    retriever=retriever,
    return_source_documents=True
  )
  # Step 5: Ask summarization prompt
  summary_prompt = "Please summarize this document based on the key topics:"
  with st.spinner("Running RAG summarization..."):
    # result = rag_chain.run(summary_prompt)
    result = rag_chain.invoke(summary_prompt)
  st.subheader("📌 Summary")
  st.write(result)
```

TATA CONSULTANCY SERVICES

During the last dry run our teams had quite a lot of learnings and this time sharing this code as well. But there should be no change to the way we are programming etc.

TATA

List of Models (Current available)

- azure/genailab-maas-gpt-35-turbo
- azure/genailab-maas-gpt-4o
- azure/genailab-maas-gpt-4o-mini
- azure/genailab-maas-text-embedding-3-large
- azure/genailab-maas-whisper
- azure_ai/genailab-maas-DeepSeek-R1
- azure_ai/genailab-maas-DeepSeek-V3-0324
- azure_ai/genailab-maas-Llama-3.2-90B-Vision-Instruct
- azure_ai/genailab-maas-Llama-3.3-70B-Instruct
- azure_ai/genailab-maas-Llama-4-Maverick-17B-128E-Instruct-FP8
- azure_ai/genailab-maas-Phi-3.5-vision-instruct
- azure_ai/genailab-maas-Phi-4-reasoning

TATA CONSULTANCY SERVICES

TATA

Thank you