5G Network Configurations with NVIDIA AI LLM Agents and Kinetica Accelerated Database

Agenda

- Introduction
- 5G-slicing-lab Overview
- Introduction to LLM Agents
- Agent Building Essentials
- 5G Network Agent Overview
- Next Steps

Workshop Objectives

By the end of this workshop you will

- **Know more about 5G Network Simulator**
  - How to create Scenarios
  - How to extract Data
- **Know how to build LLM Agents**
- **Know how to use tools like:**
  - LLM NIM Inference Endpoints
  - Langgraph
  - Langchain

Prerequisites

- Fundamentals of Deep Learning
  - Deep learning 101 in Python
  - Participants should have a firm grasp of the neural network training process.

- Rapid Application Development with Large Language Models (LLMs)
  - LLM fundamentals
  - Participants should be familiar with transformer architecture and the process of converting text into tokens and embeddings.

- Intermediate Python Experience
- Familiarity with 5G Networking Fundamentals

Core Tools

python™

kinetica

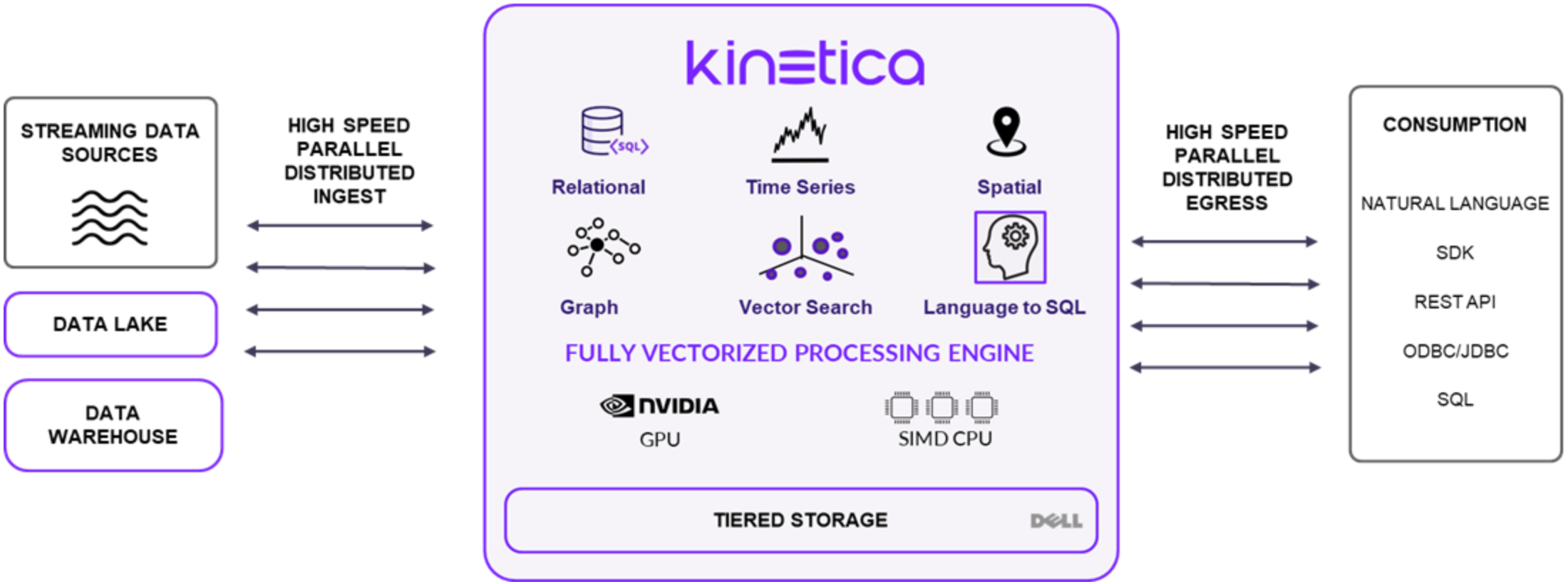
LangChain

OPEN AIR INTERFACE 5G

LangGraph

Kinetica Database

Kinetica powers sub-second analytics and vector search on massive amounts of real-time data



PHINE.TECH

5G startup to enable and accelerate 5G Use Cases

- Founded in Austria in 2023
- Experts in 5G, 5G Use Cases and Software Development
- OpenAirInterface Core Network Key contributors since 2022
  - Edge Computing in 5G
  - Quality of Service
  - User-friendly YAML-based configuration
- Offers Virtual 5G Lab
  - Developer-friendly 5G Lab-as-a-Service
  - Simplifies configuration of 5G deployments for 5G Use Case Development
  - AI, Robotics, NTN, Automotive

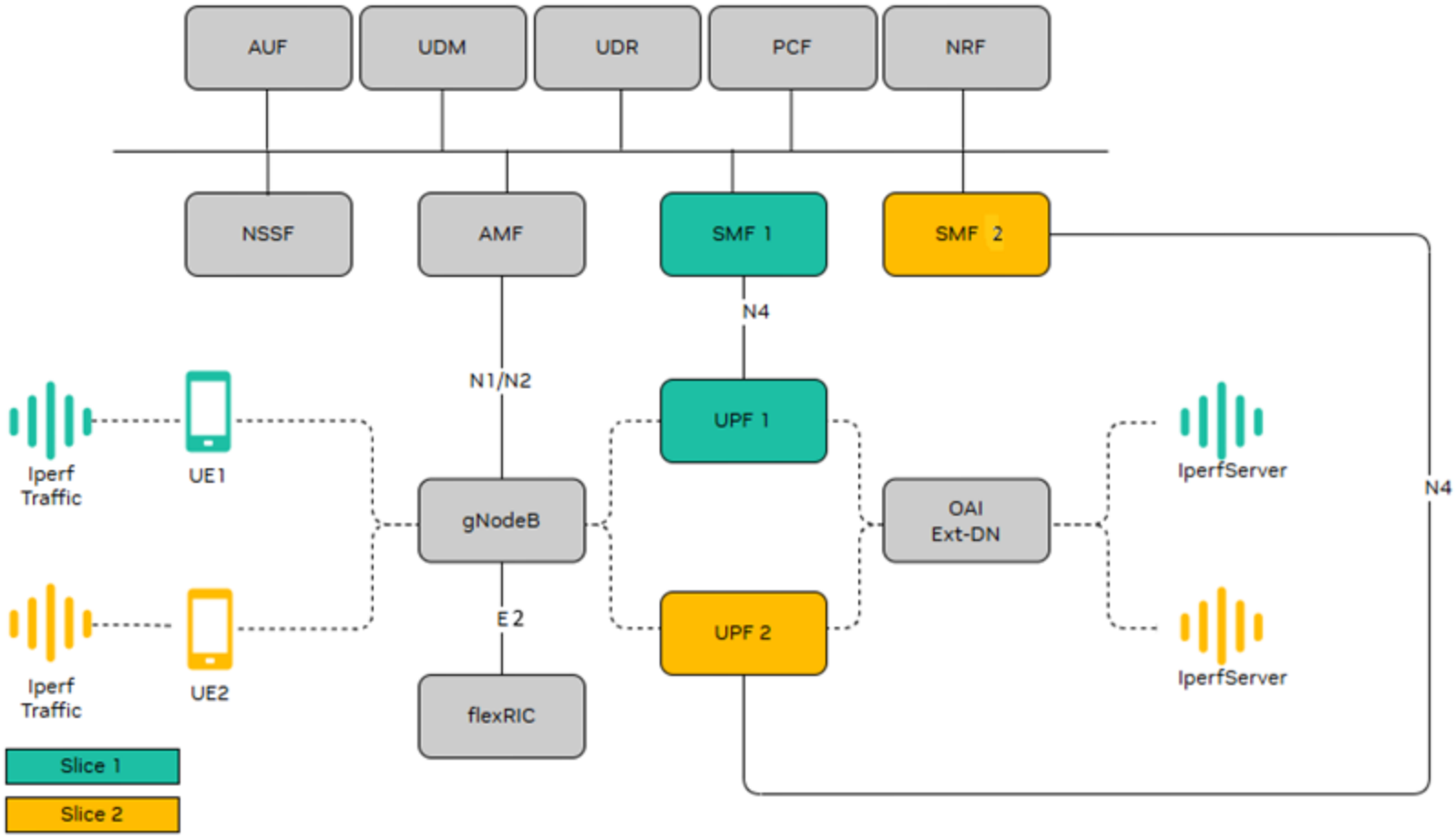
VIRTUAL 5G LAB BY PHINE.TECH

5g Network Slicing Lab Overview

5G Lab Scenario Setting

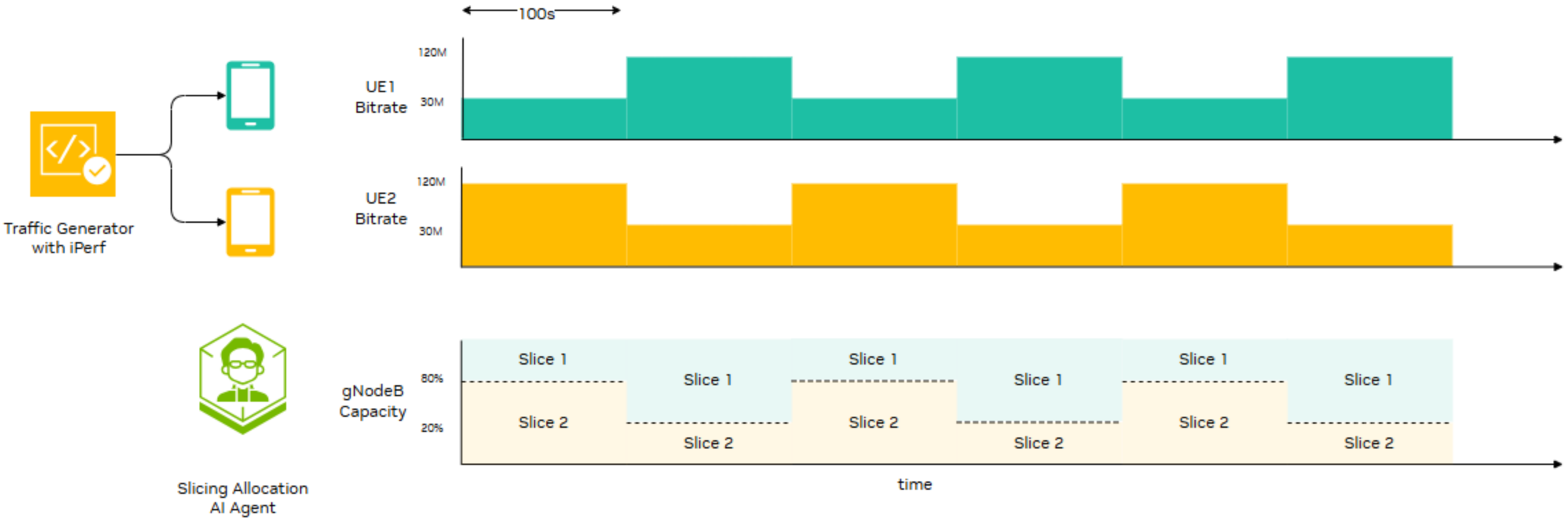
We have created a 5G Scenario using Open Air Interface Software Stack

The Open Air Interface (OAI) is an open-source software stack designed to implement the 3GPP wireless standards, facilitating research and development in mobile telecommunications technologies.



Experiment Setting Dynamic Slicing

The experiment will involve two User Equipments (UEs) across two different network slices, each dynamically adjusting their traffic rate between 30 Mbps and 120 Mbps every 100 milliseconds, with an AI agent dynamically managing slice allocations within the gNodeB



Why use LLM Agents?

Although this is a simple use case in reality the complexity of the possible scenarios creates an ideal situation for LLMs

Perception/Monitoring

Action/Configuration

UE-Level Metrics:

- RRC State Changes
- Throughput
- Packet Loss/Delay
- Signal Quality

Cell-Level Metrics:

- Resource Utilization
- Traffic Volume
- Handover Statistics
- Interference Levels

Network-Wide Metrics:

- Energy Consumption
- QoS Parameters

Reasoning

UE- Level Actions:

- Handover[1]
- RC
- QoS Flow Mapping[2]
- UE Power Control
- RRC State Transition
- Beam Assignment

Cell-Level Actions:

- Resource Allocation Adjustment[3]
- Power Control
- Cell Sleep/Wake
- Interference Management
- Beamforming Configuration

Network-Wide Actions:

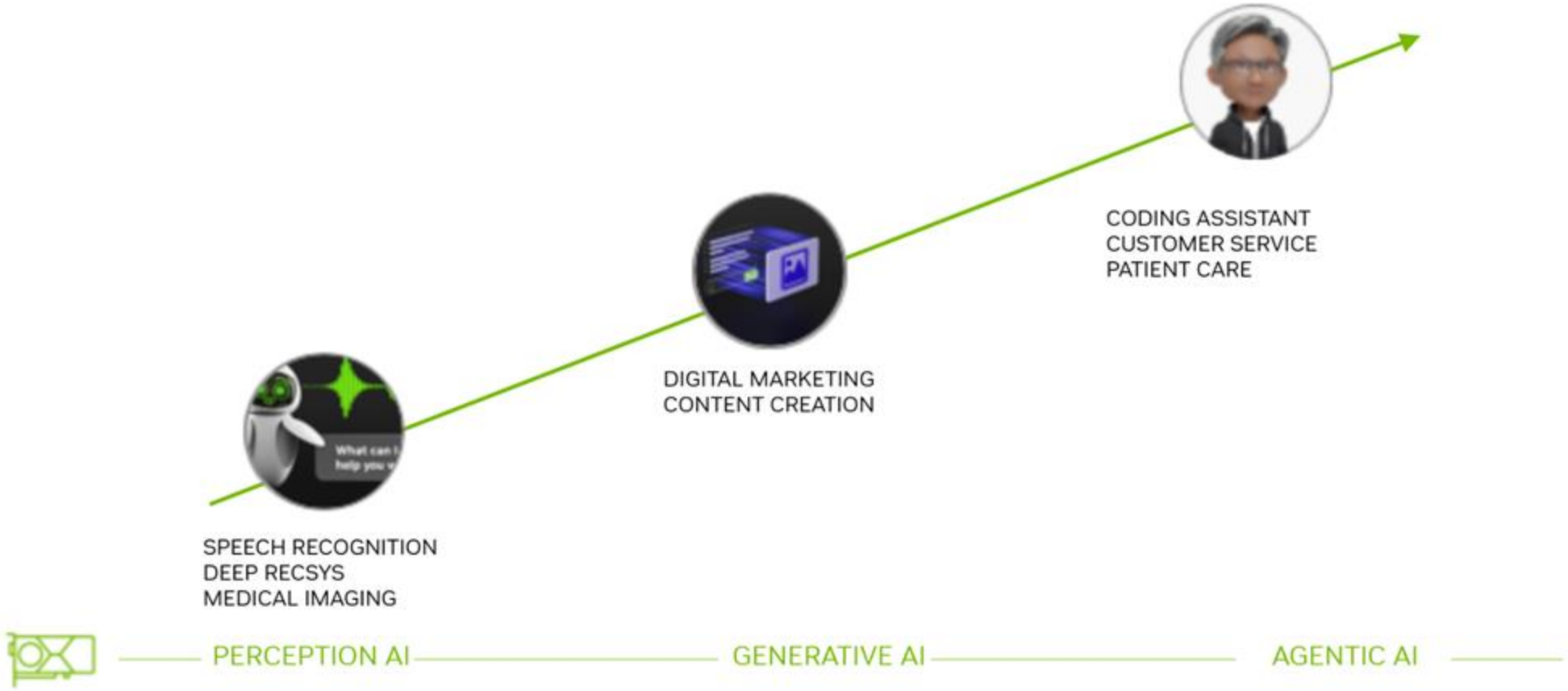
- Network Slicing

1,2,3 implemented in the standard flexRIC deployment

Introduction to LLM Agents

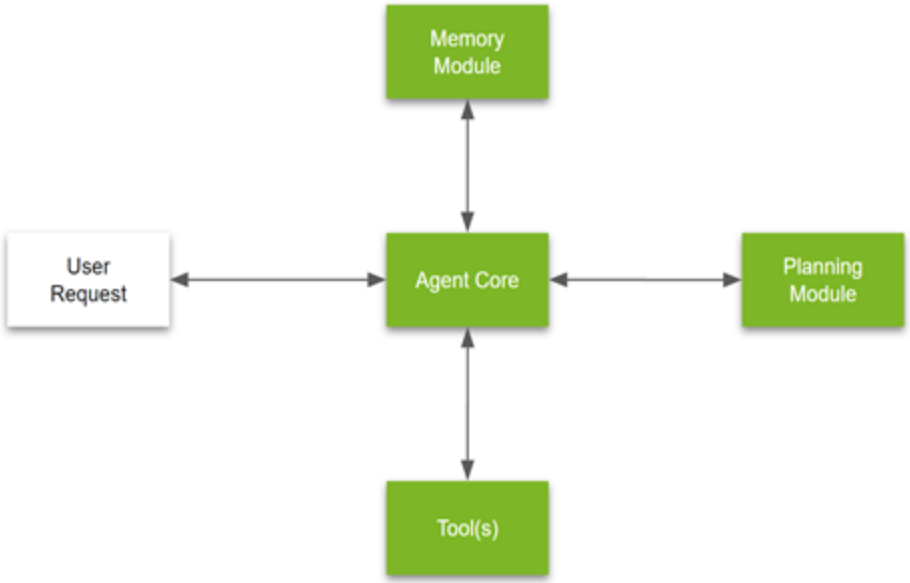
Evolution of AI

Agentic AI Enables More Powerful AI Applications



Introduction to LLM Agents

- LLM Agents are AI-powered systems that leverage Large Language Models (LLMs) to autonomously perform tasks, make decisions, and interact with users or other systems.
- They combine reasoning, memory, and action execution to go beyond simple text generation.

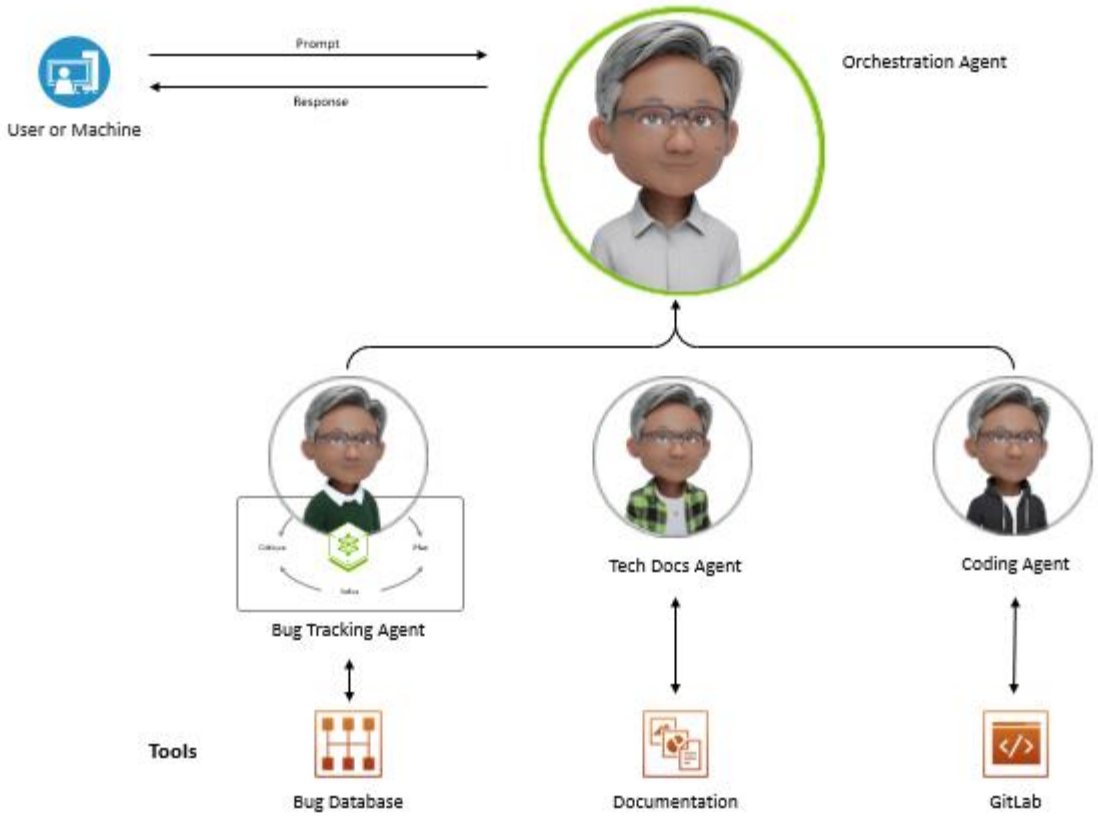


Key Components

- **LLM/Agent Core** – The central engine responsible for understanding and generating text. It defines the agent's behavior and serves as the primary decision-making unit.
- **Planning Module** – Determines the sequence of actions needed to accomplish a goal. This may involve breaking down complex queries, reflecting on previous steps, and optimizing task execution.
- **Memory Module** – Retains past interactions and contextual information to enhance continuity and decision-making.
- **Tools** – Connects to external APIs, databases, or systems, enabling the agent to retrieve, process, and manipulate data as needed for specific tasks.

https://developer.nvidia.com/blog/introduction-to-llm-agents/#what_is_an_ai_agent

Specialized Agents Form Teams to Automate Work



Agentic Frameworks

How do we build these agents?

What are Agentic Frameworks?

- Agentic frameworks provide structured environments for building AI agents that can reason, plan, and execute tasks autonomously.
- They help orchestrate interactions between LLMs, memory, tools, and external environments.

Why Do We Need Them?

- Enable scalable and structured AI workflows.
- Enable communication between agents though states.
- Facilitate multi-step reasoning and decision-making.
- Support tool integration, state management, and automation.

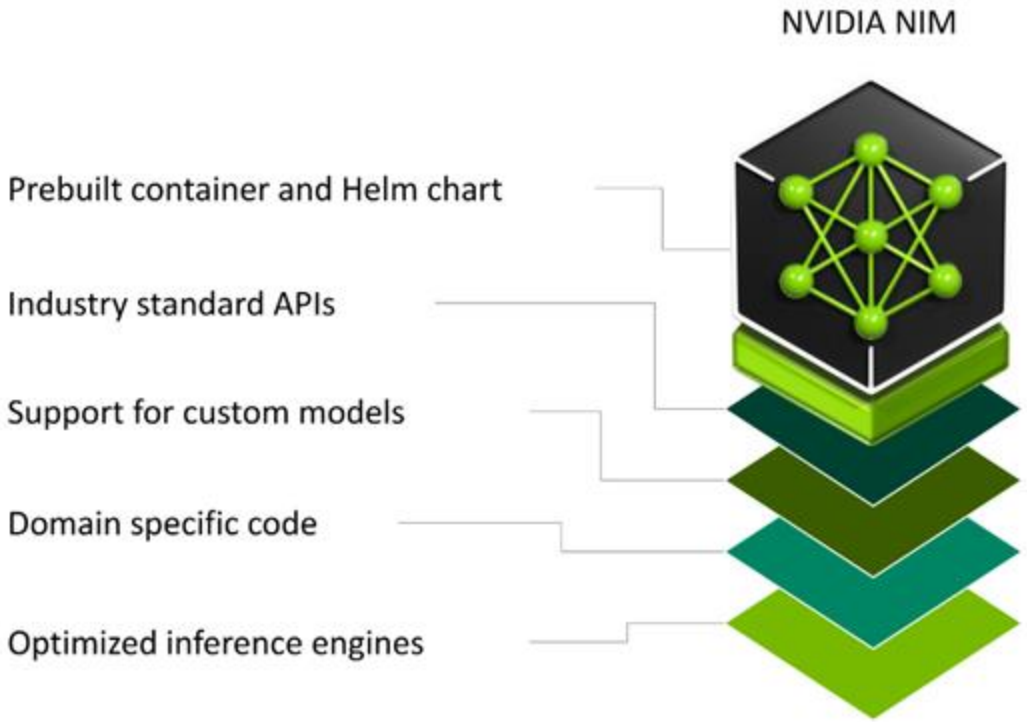
Langchain vs Langgraph

Comparing popular agentic frameworks

Feature	LangGraph	LangChain
Execution Model	**Graph-based**, parallel & stateful.	**Sequential**, tool-driven.
Best For	Multi-agent workflows, parallel tasks.	Single-agent apps, RAG, chatbots.
State Management	Stateful, retains intermediate results.	Stateless (can store memory).
Scalability	Handles complex workflows efficiently.	Designed for simpler use cases.
Tool Calling	Supports **multiple tools in one step**.	Calls tools sequentially.
Ease of Use	More structured, explicit workflows.	Easier for quick prototyping.

NVIDIA NIM Optimized Inference Microservices

Accelerated runtime for generative AI



Portable Deploy anywhere with security and control of AI applications and data

Easy to Use Speed time to market with prebuilt, continuously maintained microservices

Enterprise Supported Gain confidence with API stability, security patching, quality assurance and support

Performance Optimize throughput to maximize token generation for lower TCO and ROI

Microsoft Azure · aws · Google Cloud · ORACLE · DGX & DGX Cloud · DELL Technologies · Hewlett Packard Enterprise · Lenovo · SUPERMICRO

[Try NVIDIA NIM APIs]

NVIDIA

ChatNVIDIA

ChatNVIDIA is a LangChain integration that enables easy interaction with NVIDIA's chat models deployed via NIM.

How It Works:

- Uses **langchain-nvidia-ai-endpoints** to connect to NVIDIA-hosted models.
- Provides a **unified interface** for integrating **LLMs into AI applications**.
- Works with **NIM-hosted models**, enabling developers to quickly build and test conversational AI

Agent Building Essentials

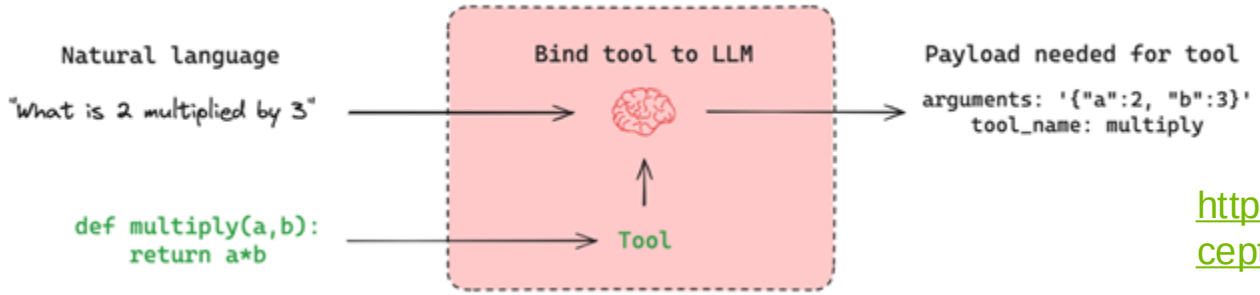
Tool calling

What is Tool Calling?

- In LangChain, tool calling allows LLMs to **invoke external functions, APIs, or utilities** dynamically.
- The LangChain framework is designed so that when a tool and a model are combined, the model not only gains access to the tool, its description (docstring), expected input/output formats, and name. By incorporating this information into the model's context, the model can determine what the tool does, when to use it, and how to correctly format tool calls—including passing the appropriate inputs based on the user's prompt.

Key Concepts:

- Tool Creation: Use the @tool decorator to create a tool. A tool is an association between a function and its schema.
- Tool Binding: The tool needs to be connected to a model that supports tool calling. This gives the model awareness of the tool and the associated input schema required by the tool.
- Tool Calling: When appropriate, the model can decide to call a tool and ensure its response conforms to the tool's input schema.
- Tool Execution: The tool can be executed using the arguments provided by the model.



https://python.langchain.com/docs/concepts/tool_calling/

ReAct Agent in Langgraph

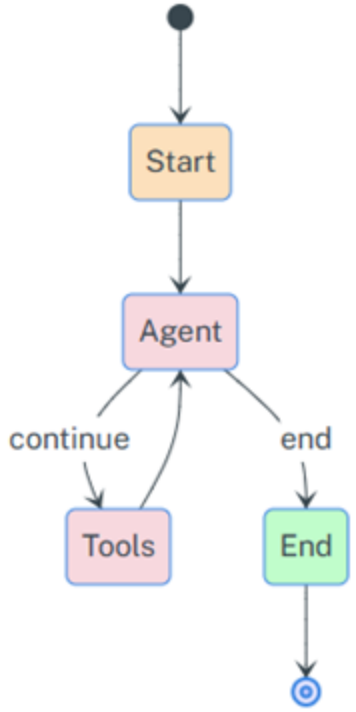
ReAct (Reasoning + Acting) is an agent architecture that combines step-by-step reasoning with tool use. LangGraph provides a prebuilt function create_react_agent() to easily implement this architecture.

Key Features:

- Step-by-step reasoning
- Dynamic tool selection and usage
- Iterative response refinement

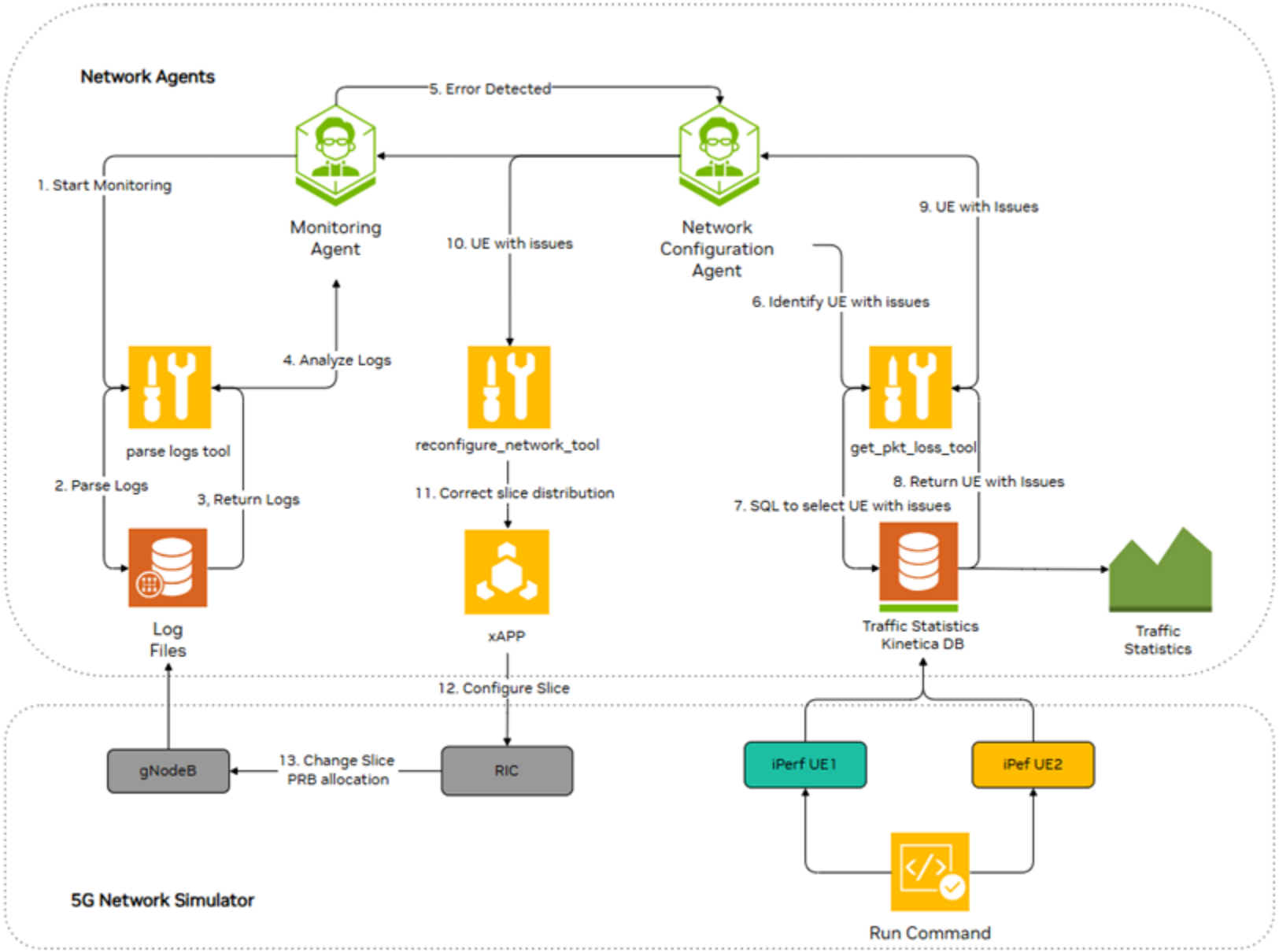
How it works:

1. Agent receives a query
2. Agent reasons about the query
3. Agent decides whether to use a tool or respond
4. If using a tool:
    i. Tool is executed
    ii. Result is passed back to the agent
5. If not using a tool:
    i. Agent formulates final response to user
6. Process repeats from step 2 until task is complete



https://langchain-ai.github.io/langgraph/how-tos/create-react-agent/

5g-Network Architecture



Agents Overview

We have defined 2 agents here :

1. Monitoring Agent:
    a. Continuously reads gNodeB logs from ../llm-slicing-5g-lab/logs/gNodeB.log.
    b. Analyzes each chunk for SDU buffer full errors.
    c. Routes to the Configuration Agent if an error is detected.

1. Configuration Agent:
    a. Called when an error is detected in the gNodeB logs.
    b. Has two tools bound to it: **get_packet_loss** and **reconfigure_network**.
    c. First, retrieves the latest packet loss logs from the database using the get_packet_loss tool.
    d. Analyzes the logs and determines which UE needs more bandwidth. Based on this, it assigns higher bandwidth to the selected UE.
    e. Calls the **reconfigure_network** tool to use xAPP and reconfigure the network.
    f. Returns control to the Monitoring Agent to continue monitoring.

Agents are defined in **./agentic-llms/agents.py**

Tools overview

We have defined 2 tools in this workflow:

1. get_packet_loss tool:
    a. Queries the database and retrieves a DataFrame containing per-UE packet loss statistics.
2. reconfigure_network tool:
    a. Calls the xAPP with optimized slicing parameters to adjust network configurations

Tools are defined in **./agentic-llms/tools.py**

Langgraph workflow is defined in **./agentic-llms/langgraph_agent.py**

Streamlit UI is defined in **./agentic-llms/chatbot_DLI.py**

5g Lab + Kinetica database Implementation

LLM Agent implementation

Conclusion

Congratulations on completing the workshop!

This application was just a first step in demonstrating how AI-agents can help solve real-world challenges in automating mobile networks.

Next steps:

1. Dive deeper into agentic AI :
   a. https://developer.nvidia.com/blog/introduction-to-llm-agents/#what_is_an_ai_agent
   b. Check out Nvidia developer blogs: https://developer.nvidia.com/blog/
2. Learn more about Open Air Interface: https://openairinterface.org/
3. Explore other applications of agentic AI in networking.

Thank you!