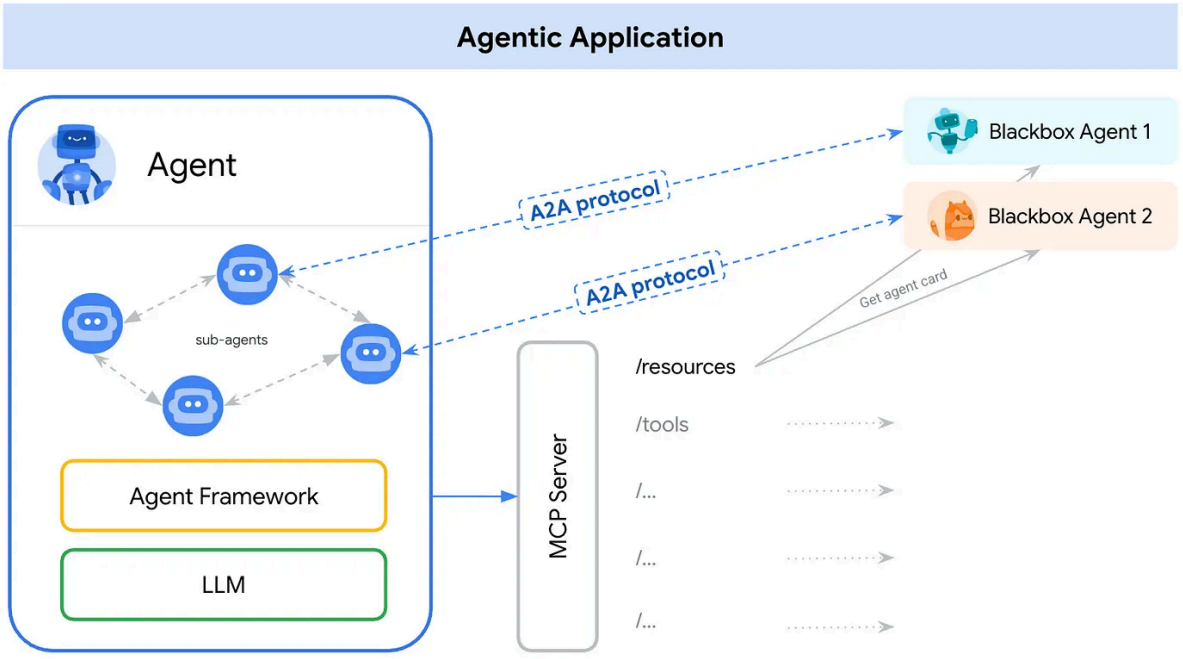
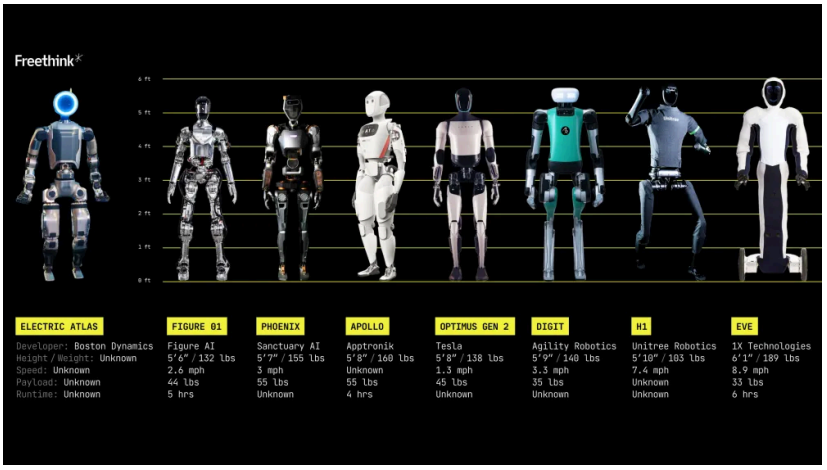
Certified Agentic & Robotic AI Engineer

Forge the Future of Intelligent Machines: Become a Certified Pioneer in the $100 Trillion Agentic AI & Cloud Native Era





Agentia: The Future of Cloud Native Agentic AI — Build It Today

Version: 21.5 (Implementation and adoption starting from April 11, 2025)

Must Watch This Video To **Get Started**: Customer Experience Trends for 2025: The Rise of AI Agents and Agentic AI

Presentation Slide Deck:
https://docs.google.com/presentation/d/1VNFGsCYMDT1VTe8W1wxFbmAwYsJ1I0Y-6CnTvuCEn98/edit?usp=sharing

Listen to The Certified Agentic & Robotic AI Engineer Deep Dive Discussion **Podcast**:
https://notebooklm.google.com/notebook/a1c886ca-bcd3-4cb4-b1a7-4a6708992bc3/audio

Certification **Program Reviews** and Grading:

By the Latest ChatGPT 4.5:
https://chatgpt.com/share/67df1680-f674-8001-9466-8f9127a289c5

By the Latest Grok 3.0:
https://grok.com/share/bGVnYWN5_3b6f703a-1900-4108-976c-0c887f692f4f

Our Vision: Agentia World

Imagine a world where everything is an AI agent, from your coffee machine to your car, from businesses to entire cities. Picture a world transformed into Agentia—a dynamic, living network of intelligent AI agents seamlessly integrated into our daily lives. From our homes and offices to entire cities, systems no longer communicate through outdated APIs but through sophisticated, intelligent dialogues driven by state-of-the-art AI frameworks. Agentia scales effortlessly across the globe, thanks to its foundation in cloud-native technologies. Agentia is more than digital—it's also physical, brought to life by robots that serve as embodied agents interacting with and enhancing our physical world.

Imagine a world characterized by:

- **Ubiquitous Intelligence:** Smart homes, autonomous vehicles, adaptive businesses, and responsive urban infrastructure all functioning as independent yet collaborative intelligent agents. Every aspect of life is augmented by autonomous, proactive intelligence.
- **Agentic Communication:** REST APIs become obsolete, replaced by fluid, context-aware agent-to-agent conversations. These intelligent dialogues leverage advanced Large Language Models and next-generation agentic frameworks, enabling systems to negotiate, coordinate, and cooperate with unprecedented sophistication.
- **Physical Embodiment:** Digital intelligence meets physical reality. Agentia extends beyond screens, employing humanoid robots and Physical AI to interact naturally with humans, accomplish tasks, and enrich our tangible world with intelligent presence.
- **Intelligent Dialogues, Beyond Endpoints:** Agents engage in deep, meaningful interactions, comprehending context, intent, nuance, and complexities far beyond basic endpoint invocations. These intelligent conversations empower systems to collaboratively achieve intricate and meaningful goals.
- **Scalability and Cloud-Native Foundation**: Agents thrive on a global scale, seamlessly expanding through cloud-native technologies that ensure flexibility, resilience, and efficiency across diverse environments and regions.

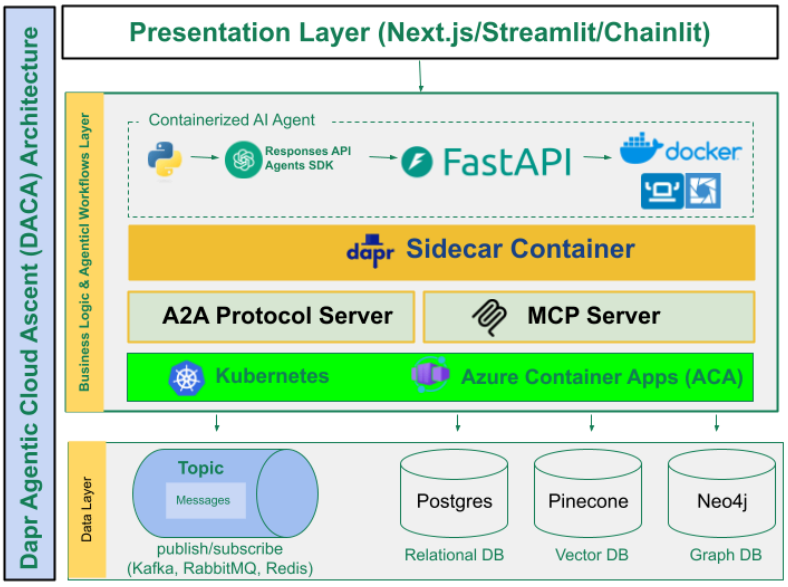
This initiative tackles the critical challenge: “How do we design AI agents that can handle 10 million concurrent agents without failing?”

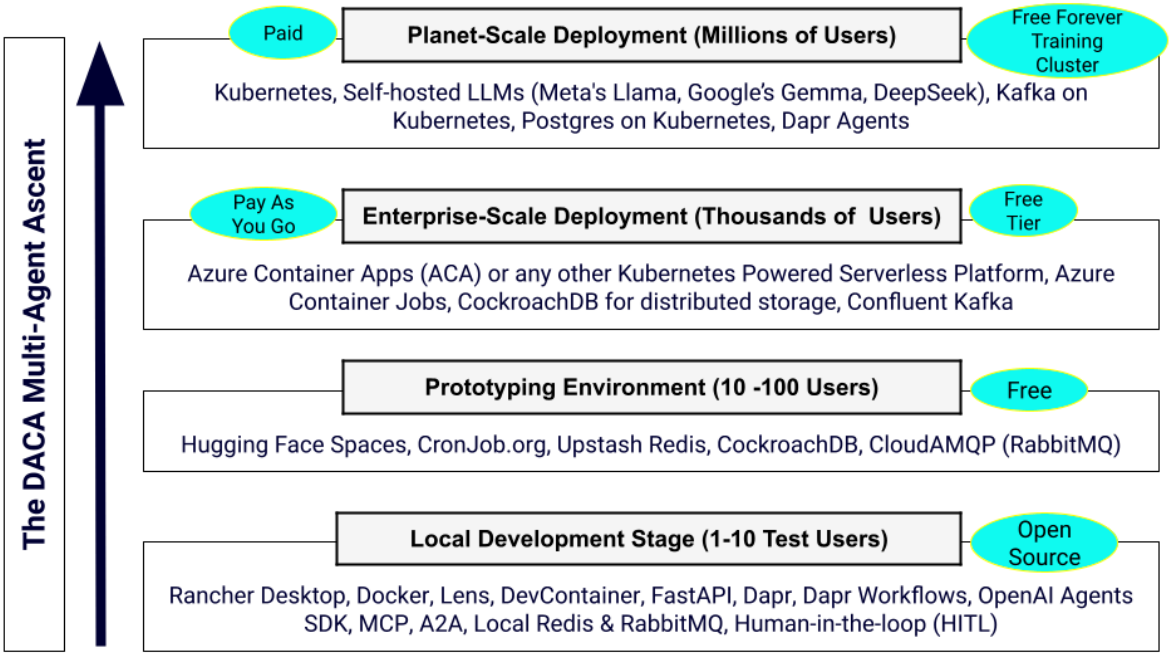
Note: The challenge is intensified as we must guide our students to solve this issue with minimal financial resources available during training.

This program is purposefully crafted to cultivate the next wave of visionary entrepreneurs and tech pioneers—the builders of Agentia. We provide comprehensive education, cutting-edge resources, and hands-on experiences to empower you not just to envision but to create the autonomous systems and intelligent infrastructure defining the Agentia era. As the architects of this interconnected AI future, you have the opportunity to lead innovation that fundamentally reshapes society.

Our Implementation: Dapr Agentic Cloud Ascent (DACA) Design Pattern

The **Dapr Agentic Cloud Ascent (DACA)** design pattern is a strategic framework for building and deploying scalable, resilient, and cost-effective agentic AI systems, grounded in **AI-first** and **cloud-first development** as its core tenets. It harnesses the simplicity of the OpenAI Agents SDK for intelligent agent logic, the Model Context Protocol (MCP) for standardized tool integration, the interoperability of Agent2Agent Protocol (A2A) for seamless agent communication, and the distributed capabilities of Dapr, all deployed through a progressive cloud-native pipeline that leverages free-tier services and Kubernetes to achieve planetary-scale intelligence. DACA integrates event-driven architecture (EDA), a three-tier microservices structure, stateless computing, scheduled workflows (CronJobs), and human-in-the-loop (HITL) oversight to meet the autonomy, real-time demands, scalability, and complexity of AI agents. This guide consolidates DACA’s architecture, components, deployment stages, and benefits, positioning AI-first and cloud-first principles, alongside A2A and MCP, as pivotal enablers of our vision: **Agentia World**.





Let's understand and learn about our winning design pattern for developing and deploying planet scale multi-agent systems:

https://github.com/panaversity/learn-agentic-ai/blob/main/comprehensive_guide_daca.md

AI-First and Agent-Native Cloud-First: Foundational Tenets of DACA
DACA's power lies in its dual commitment to AI-first and cloud-first development:

AI-First Development:

Why It Matters: AI agents are the system's brain, driving autonomy, decision-making, and adaptability. By prioritizing AI from the start, DACA ensures systems are inherently intelligent, capable of natural language dialogues, tool integration, and dynamic collaboration.
How It's Implemented: Uses the OpenAI Agents SDK for agent logic, A2A for agent-to-agent communication, and MCP for tool access, enabling agents to handle complex tasks (e.g., coordinating logistics or automating homes).
Agentia Alignment: Supports a world where every entity is an AI agent, interacting via intelligent dialogues rather than rigid APIs.

Agent-Native Cloud First Development:
Why It Matters: Infrastructure optimized for agents provides scalability and programmatic interfaces, unlike human-centric clouds.
How It's Implemented: Leverages containers (Docker/Rancher), orchestration (Kubernetes), serverless platforms (Azure Container Apps), and managed services (CockroachDB, Upstash Redis) to deploy and scale agents efficiently.
Agentia Alignment: Enables Agentia's global reach, ensuring agents can scale from prototypes to millions of users using cloud resources.

Together, these tenets make DACA a forward-looking framework, blending AI's intelligence with the cloud's scalability to create a cohesive, planet-scale agent ecosystem.

DACA reflects the implementation of our vision therefore it will be covered and we will learn to implement it in all our classes.

From Vertical LLM Agents to Humanoid Robotics

Our initial program focus is on sophisticated software agents, including **Vertical LLM Agents** and the design, development, and deployment of innovative Agentic AI solutions using cloud native technologies. **Subsequently, our journey will embrace humanoid robots and Physical AI**, closing the gap between digital cognition and physical interactions, thereby developing systems capable of genuine human-like understanding and responsiveness.

Our vision of **Agentia** and its implementation is ambitious, but not only feasible—it's aligned with the clear trajectory of technological innovation and current investment trends. With sustained commitment, targeted investment, and community collaboration, significant components of Agentia can realistically emerge. We believe **OpenAI Responses API** and **OpenAI Agents SDK** provide a solid foundation for building our Agentia World. Docker/Rancher and Kubernetes complement the OpenAI Agents SDK, broadening the stack's capabilities to global scalability. The following chart clearly identifies why we are using OpenAI Agents SDK as our main framework for Agentic development:

Table 1: Comparison of Abstraction Levels in AI Agent Frameworks

Framework	Abstraction Level	Key Characteristics	Learning Curve	Control Level	Simplicity
OpenAI Agents SDK	Minimal	Python-first, core primitives (Agents, Handoffs, Guardrails), direct control	Low	High	High
CrewAI	Moderate	Role-based agents, crews, tasks, focus on collaboration	Low-Medium	Medium	Medium
AutoGen	High	Conversational agents, flexible conversation patterns, human-in-the-loop support	Medium	Medium	Medium
Google ADK	Moderate	Multi-agent hierarchies, Google Cloud integration (Gemini, Vertex AI), rich tool ecosystem, bidirectional streaming	Medium	Medium-High	Medium
LangGraph	Low-Moderate	Graph-based workflows, nodes, edges, explicit state management	Very High	Very High	Low
Dapr Agents	Moderate	Stateful virtual actors, event-driven multi-agent workflows, Kubernetes integration, 50+ data connectors, built-in resiliency	Medium	Medium-High	Medium

The table clearly identifies why OpenAI Agents SDK should be the main framework for agentic development for most use cases:

- It excels in **simplicity** and **ease of use**, making it the best choice for rapid development and broad accessibility.
- It offers **high control** with **minimal abstraction**, providing the flexibility needed for agentic development without the complexity of frameworks like LangGraph.
- It outperforms most alternatives (CrewAI, AutoGen, Google ADK, Dapr Agents) in balancing usability and power, and while LangGraph offers more control, its complexity makes it less practical for general use.

If your priority is ease of use, flexibility, and quick iteration in agentic development, OpenAI Agents SDK is the clear winner based on the table. However, if your project requires enterprise-scale features (e.g., Dapr Agents) or maximum control for complex workflows (e.g., LangGraph), you might consider those alternatives despite their added complexity.

In short, not only is Agentia feasible—it is inevitable. **The future waits for no one, especially when it's agentically powered.** Become part of the generation that builds Agentia by implementing DACA design pattern—transforming imagination into reality, one intelligent agent at a time.

Panaversity's Innovative Teaching Strategy for Agentic AI
Agentic AI is advancing quickly, requiring smart teaching methods to prepare students with essential skills. This document offers a step-by-step educational plan for introducing beginners to Agentic AI, beginning with the core OpenAI Responses API and Agents SDK, then advancing to the more sophisticated cloud native frameworks. After mastering these agentic AI basics, students progress to deploying long-term, stateful AI agents in the cloud using Docker/Rancher, Serverless Containers, and Kubernetes. By starting with simple tools to establish clear concepts and hands-on abilities, this method sets the stage for tackling the more intricate features of advanced systems.

Material to Understand the Coming Agentic AI Age:

- Agentic AI Explained
- AI Agents Explained Like You're 5
- AI Is About To FLIP Your Life Upside Down
- The Future Is Agentic
- The agent economy
- Why Vertical LLM Agents Are The New $1 Billion SaaS Opportunities
- Vertical AI Agents Could Be 10X Bigger Than SaaS
- OpenAI's Path to AGI | Five Levels of Intelligence
- AI Agents: Are We Ready For Machines That Make Decisions?
- Function calling
- Generative AI's Act o1
- Watch AGI could Double GDP
- The INSANE Race for AI Humanoid Robots

This core program duration is one year, if you take one course at a time and equips you with the skills to thrive in the age of Conversational, Generative, Agentic, and Physical AI, and cloud native distributed computing. However, **you can reduce the duration of the program if you take multiple courses in a quarter**. You will become an expert AI Agent, and Humanoid Robotics Developer. The program is divided into two levels: core level and professional level. **Students will be able to start working after completing the core level. They will continue their professional level studies while working.**

Why This Program?

- **Cutting-Edge Skills:** Develop in-demand skills to build and deploy intelligent, scalable Agentic AI Cloud solutions.
- **Industry-Ready:** Prepare for global certifications, startup and freelance opportunities after just nine months.
- **Future-Proof Your Career:** Stay ahead of the curve in a rapidly evolving tech landscape.

What You'll Learn:

- **Multi-AI Agent Systems:** Learn key principles of designing effective AI agents, and organising a team of AI agents to perform complex, multi-step tasks. Build Knowledge Graphs. Apply these concepts to automate common business processes. We will start by learning OpenAI Responses API and OpenAI Agents SDK.
- **Physical AI and Humanoid Robotics**: We will learn to design, simulate, and deploy advanced humanoid robots capable of natural interactions.
- **Distributed System Design**: Designing systems that run on multiple computers (or nodes) simultaneously, interacting and coordinating their actions.
- **Designing AI Solutions using Design Thinking and Behaviour Driven Development (BDD)**: We will learn to leverage these methodologies to create AI solutions that are not only technically sound but also highly user-centric and aligned with real-world needs.
- **Fine-Tuning Open-Source Large Language Models using PyTorch**: We will learn to fine-tuning open-source Large Language Models (LLMs) like Meta LLaMA 3 using PyTorch.

  Flexible Learning:

- **Earn While You Learn:** Start freelancing or contributing to projects after the third quarter.

Also Focus on Communication Skills:

- **Technical + Communication:** [You can negotiate any reality you want]

I. Cloud Native Agentic AI Core Level

- ## AI-101: Modern AI Python Programming - Your Launchpad into Intelligent Systems

  AI-101 is your comprehensive gateway to Python programming for Artificial Intelligence. This course is laser-focused on equipping you with **Modern Python skills**, emphasizing **static typing**, the cornerstone of robust and scalable AI development. Uniquely, you will also learn to **harness the power of AI to write Python code**, accelerating your learning and development process. From foundational concepts to advanced techniques and practical web application development, AI-101 provides everything you need to excel in AI-driven projects and beyond.

Key Learning Modules:

- **Module 1: Python Fundamentals & Modern Typing:** Establish a strong foundation in Python syntax, data structures (lists, dictionaries, sets, tuples), control flow, and functions. Critically, we introduce **Python's type hinting system, Generics, and Decorators** emphasizing its importance for code clarity, error prevention, and maintainability, especially in complex AI projects.
- **Module 2: Object-Oriented Programming (OOP) in Python for AI:** Master the principles of OOP (classes, objects, inheritance, polymorphism, encapsulation, dataclasses) and understand how to apply them effectively in AI development. Learn to structure complex AI systems using object-oriented design for modularity and reusability.
- **Module 3: Advanced Python Concepts: Asynchronous Programming & Performance:** Dive into advanced Python features like asynchronous programming (`asyncio`) for building efficient and concurrent applications, crucial for handling large datasets and complex AI workloads. Explore performance optimization techniques and understand the role of Python's Global Interpreter Lock (GIL) and upcoming solutions like "No GIL" for enhanced concurrency.
- **Module 4: AI-Assisted Python Programming:** Leverage the power of AI tools to enhance your Python coding skills. This module will introduce you to techniques and tools that utilize AI to generate code snippets, debug programs, refactor code, and improve your overall Python development workflow. Learn to work *with* AI to become a more efficient Python programmer.
- **Module 5: Web Application Basics with Python (UV, Streamlit, & GitHub):** Gain practical experience in building basic web applications using Python. We will explore lightweight web frameworks and tools including UV for environment management, and Streamlit for rapid UI creation. You'll also learn essential version control using GitHub for collaborative development and project management.
- **Module 6: Future of Python & Python in AI:** Explore the evolving landscape of Python, including upcoming features and performance improvements. Discuss the continued dominance of Python in the AI field and its application in cutting-edge AI domains like Machine Learning, Deep Learning, and Agentic AI.

Course Outcomes:

Upon successful completion of this course, students will be able to:

- Write proficient Modern Python code utilizing static typing for robust and maintainable AI applications.
- Apply Object-Oriented Programming principles effectively in Python for structuring complex AI systems.
- Implement asynchronous programming in Python for building high-performance AI applications.
- Utilize AI-powered tools and techniques to enhance their Python coding efficiency and quality.
- Develop basic Python web applications using modern tools and understand fundamental web development concepts.
- Articulate the future directions of Python and its continued crucial role in the field of Artificial Intelligence.

    - **Certification:**
        - Certified Professional Python Programmer (CPPP1)

Learning Repo:
https://github.com/panaversity/learn-cloud-native-modern-python

Prerequisite: None

- # AI-201: Fundamentals of Agentic AI and DACA AI-First Development

Course Code: AI-201
Credits: 3
Duration: 14 weeks
Prerequisites: Basic programming knowledge (Python recommended), familiarity with software development concepts
Delivery Mode: In-person/Online (Hybrid)
Instructor: TBD
Course Level: Undergraduate/Graduate

Course Description

This course introduces students to the principles and practices of Agentic AI and Data-Augmented, Context-Aware (DACA) AI-first development. Students will explore foundational theories, modern tools, frameworks and design patterns for building intelligent, context-aware AI agents. The course also covers Agentic Protocols like Model Context Protocol (MCP) and Agent2Agent (A2A) Protocol in depth. Through hands-on projects, students

will gain practical experience in developing and deploying AI-driven applications.

Learning Outcomes

Upon successful completion of this course, students will be able to:

1. Explain the theoretical foundations of Agentic AI and DACA frameworks.
2. Utilize UV and OpenAI Agents SDK to build functional AI agents.
3. Apply agentic design patterns to solve real-world problems.
4. Implement memory management techniques using LangMem and mem0.
5. Use Model Context Protocol (MCP) to connect AI Agents and interact with external data sources, tools, and systems.
6. Use Agent-to-Agent (A2A) protocol is to facilitate secure, standardized communication and data exchange between autonomous AI agents.

Course Outline

Week 1: Agentic and DACA Theory

- Introduction to Agentic AI and Data-Augmented, Context-Aware (DACA) principles
- Core concepts and applications in modern AI development

Weeks 2–6: UV and OpenAI Agents SDK

- Overview of UV and OpenAI Agents SDK
- Building and configuring AI agents
- Hands-on projects with agent development and testing

Weeks 7–8: Agentic Design Patterns

- Study of common agentic design patterns
- Applying patterns to solve practical AI challenges

Weeks 9–11: Model Context Protocol (MCP)

- Enable integration with databases, APIs, and services for real-time data access.
- Support context management to maintain coherent and accurate interactions.
- Provides interoperability across diverse AI platforms and tools.

Weeks 12–13: Agent-to-Agent (A2A) Protocol

- Study how A2A facilitates direct, context-aware data exchange and coordination between AI agents across different systems or platforms.
- Supports real-time collaboration for tasks like distributed problem-solving or multi-agent workflows.

Week 14: Memory Management with LangMem and mem0

- Introduction to memory frameworks in AI
- Implementing LangMem and mem0 for context retention

Assessment Methods

- **Assignments**: Weekly coding and theoretical assignments (20%)
- **Projects**: Midterm project (agent development) and final project (end-to-end Agentic AI application) (20%)
- **Quizzes**: Midterm (25%) and Final (25%)
- **Participation**: Class discussions and lab participation (10%)

Required Materials

- Laptop with UV, VS Code, and Launcher Desktop installed
- Access to cloud platforms (Postgres, Redis, Hugging Face)
- Textbook: https://github.com/panaversity/learn-agentic-ai/

Recommended Materials

- Online documentation for UV, OpenAI Agents SDK, Design Patterns, MCP and A2A.

Policies

- **Attendance**: Regular attendance is expected; absences may impact participation grades.
- **Late Submissions**: 10% penalty per day for late assignments unless prior approval is granted.
- **Academic Integrity**: Plagiarism or unauthorized collaboration will result in disciplinary action per university policy.

Note: The instructor reserves the right to modify the course schedule or content as needed. Students will be notified of any changes in advance.

- # AI-202: DACA Cloud-First Agentic AI Development

Course Code: AI-202
Credits: 3
Duration: 14 weeks
Prerequisites: Successful completion of AI-201: Fundamentals of Agentic AI and DACA AI-First Development
Delivery Mode: In-person/Online (Hybrid)
Instructor: TBD
Course Level: Undergraduate/Graduate

Course Description

This advanced course builds on the foundations of AI-201, focusing on cloud-first development for Data-Augmented, Context-Aware (DACA) Agentic AI systems. Students will explore scalable, distributed AI architectures using local Kubernetes, advanced API development, and managed cloud services. The course covers distributed application runtime (Dapr), managed databases and messaging systems, model context protocols, and serverless container deployment. Through hands-on projects, students will design and deploy production-ready AI applications in cloud environments.

Learning Outcomes

Upon successful completion of this course, students will be able to:

1. Configure and manage local Kubernetes clusters using Rancher Desktop for AI development.
2. Develop advanced APIs for Agentic AI systems using FastAPI and Kubernetes.
3. Implement Dapr workflows, state management, pub/sub messaging, and secrets for distributed AI applications.
4. Integrate CockroachDB and RabbitMQ managed services into AI architectures.
5. Apply Model Context Protocol for enhanced AI agent communication.
6. Deploy serverless containerized AI applications using Azure Container Apps (ACA).

Course Outline

Weeks 1–4: Rancher Desktop with Local Kubernetes

- Introduction to Kubernetes for AI development

- Setting up and managing local Kubernetes clusters with Rancher Desktop
- Deploying AI workloads in a Kubernetes environment

Weeks 5–6: Advanced FastAPI with Kubernetes

- Advanced FastAPI features for scalable APIs
- Deploying FastAPI applications on Kubernetes
- Optimizing API performance for AI systems

Weeks 7–9: Dapr (Workflows, State, Pub/Sub, Secrets)

- Overview of Distributed Application Runtime (Dapr)
- Implementing workflows, state management, pub/sub messaging, and secrets
- Building distributed AI applications with Dapr

Weeks 10–11: CockroachDB and RabbitMQ Managed Services

- Introduction to CockroachDB and RabbitMQ for cloud-native AI
- Integrating managed database and messaging services into AI workflows
- Ensuring scalability and reliability in distributed systems

Weeks 12–13: Model Context Protocol (MCP)

- Understanding Model Context Protocol for AI agent communication
- Designing and implementing context-aware AI interactions
- Practical applications in Agentic AI systems

Week 14: Serverless Containers Deployment (ACA)

- Introduction to serverless container deployment with Azure Container Apps (ACA)
- Deploying AI applications in a serverless environment
- Monitoring and scaling serverless AI workloads

Assessment Methods

- **Assignments**: Weekly coding and theoretical assignments (20%)
- **Projects**: Midterm project (Kubernetes-based Agentic AI deployment) and final project (end-to-end Agentic AI application) (20%)
- **Quizzes**: Midterm (25%) and Final (25%)
- **Participation**: Class discussions and lab participation (10%)

Required Materials

- Laptop with UV, VS Code, and Launcher Desktop installed
- Access to cloud platforms (Postgres, Redis, Azure Container Apps)
- Textbook: https://github.com/panaversity/learn-agentic-ai/

Recommended Materials

- Online documentation for Kubernetes, Dapr, FastAPI, and Azure Container Apps
- Rancher Desktop installation

Policies

- **Attendance**: Regular attendance is expected; absences may impact participation grades.
- **Late Submissions**: 10% penalty per day for late assignments unless prior approval is granted.
- **Academic Integrity**: Plagiarism or unauthorized collaboration will result in disciplinary action per university policy.

Note: The instructor reserves the right to modify the course schedule or content as needed. Students will be notified of any changes in advance.

- # AI-301: DACA Planet-Scale Distributed AI Agents

Course Code: AI-301
Credits: 3
Duration: 14 weeks
Prerequisites: Successful completion of AI-201: Fundamentals of Agentic AI and DACA AI-First Development and AI-202: DACA Cloud-First Agentic AI Development
Delivery Mode: In-person/Online (Hybrid)
Instructor: TBD
Course Level: Graduate

Course Description

This advanced course focuses on the design and deployment of planet-scale, distributed AI agents within Data-Augmented, and DACA frameworks. Students will gain expertise in Kubernetes application development, agent-to-agent (A2A) communication protocols, voice-enabled agents, and distributed agent systems using Dapr Workflows and Agents. The course also covers self-hosting large language models (LLMs) and fine-tuning LLMs for

specialized applications. Through hands-on projects, students will build and deploy scalable, distributed AI systems capable of operating at a global scale.

Learning Outcomes

Upon successful completion of this course, students will be able to:

1. Develop and deploy Kubernetes-based AI applications in preparation for Certified Kubernetes Application Developer (CKAD) certification.
2. Integrate Dapr, MCP, and A2A communication protocols for distributed AI systems.
3. Design and deploy voice-enabled AI agents.
4. Build distributed AI agents using Dapr Workflows and Agents.
5. Host and manage self-hosted large language models (LLMs).
6. Fine-tune LLMs for domain-specific AI applications.

Course Outline

Weeks 1–6: Certified Kubernetes Application Developer (CKAD)

- Advanced Kubernetes concepts for AI application development
- Deploying and managing AI workloads on Kubernetes
- Preparing for CKAD certification through practical exercises

Weeks 7–10: Dapr Agents and Workflows

- Building distributed AI agents with Dapr Actors and Agents
- Integrating Dapr, MCP, and A2A

Weeks 11–12: Voice Agents

- Fundamentals of voice-enabled AI agents
- Developing and integrating voice interfaces for AI systems
- Testing and optimizing voice agent performance

Week 13: Self-LLMs Hosting

- Introduction to self-hosting large language models
- Setting up and managing LLM infrastructure
- Ensuring scalability and performance for self-hosted LLMs

Weeks 14: Fine-Tuning LLMs

- Techniques for fine-tuning large language models

- Customizing LLMs for specific use cases and domains
- Evaluating and deploying fine-tuned LLMs in production

Assessment Methods

- **Assignments**: Weekly coding and theoretical assignments (20%)
- **Projects**: Midterm project (Kubernetes-based distributed agent system) and final project (fine-tuned LLM deployment) (20%)
- **Quizzes**: Midterm (25%) and Final (25%)
- **Participation**: Class discussions and lab participation (10%)

Required Materials

- Laptop with UV, VS Code, Kubernetes installed
- Access to cloud platforms (Kubernetes clusters, Oracle Cloud Infrastructure (OCI) Forever Free Offer, LLM hosting services)
- Textbooks:

  https://github.com/panaversity/learn-agentic-ai/

  Certified Kubernetes Application Developer (CKAD) Study Guide: In-Depth Guidance and Practice Second Edition by Benjamin Muschko

Recommended Materials

- Online documentation for Kubernetes, Dapr, Google ADK, and LLM frameworks
- Additional CKAD certification study resources

Policies

- **Attendance**: Regular attendance is expected; absences may impact participation grades.
- **Late Submissions**: 10% penalty per day for late assignments unless prior approval is granted.
- **Academic Integrity**: Plagiarism or unauthorized collaboration will result in disciplinary action per university policy.

Note: The instructor reserves the right to modify the course schedule or content as needed. Students will be notified of any changes in advance.

II. Professional Level

- **AI-451: Physical and Humanoid Robotics AI**

  Artificial intelligence (AI) has experienced remarkable advancements in recent years. However, the future of AI extends beyond the digital space into the physical world, driven by robotics. This new frontier, known as “Physical AI,” involves AI systems that can function in the real world and comprehend physical laws. This marks a notable transition from AI models confined to digital environments. Humanoid robots are poised to excel in our human-centred world because they share our physical form and can be trained with abundant data from interacting in human environments.

  This course provides an in-depth exploration of humanoid robotics, focusing on the integration of ROS 2 (Robot Operating System), Gazebo Robot Simulator, and NVIDIA Isaac™ AI robot development platform. Students will learn to design, simulate, and deploy advanced humanoid robots capable of natural interactions. The curriculum covers essential topics such as ROS 2 for robotic control, simulations with Gazebo and Unity, and using OpenAI’s GPT models for conversational AI. Through practical projects and real-world applications, students will develop the skills needed to drive innovation in humanoid robotics.

  Learning Repo:
  https://github.com/panaversity/learn-physical-ai-humanoid-robotics

  Prerequisite: AI-101

- **AI-461: Distributed AI Computing**

  Ray is the AI Compute Engine. Ray manages, executes, and optimises compute needs across AI workloads. It unifies infrastructure via a single, flexible framework—enabling any AI workload from data processing to model training to model serving and beyond. This course provides an in-depth exploration of distributed computing using Ray, a framework for building and scaling distributed Python applications. Students will learn to develop, deploy, and optimise distributed systems using Ray, with applications in machine learning, data processing, and reinforcement learning. Ray is an open-source distributed computing framework designed to simplify the development and scaling of machine learning (ML) and Python applications.

  The following examples highlight Ray’s widespread adoption and its effectiveness in enhancing scalability, performance, and cost-efficiency in AI and machine learning workloads.

- OpenAI: Utilises Ray to train large models, including ChatGPT, enabling faster iteration at scale.
- Amazon Web Services (AWS): Employs Ray to enhance scalability, reduce latency by over 90%, and improve cost efficiency by over 90% in specific applications.
- Ant Group: Deployed Ray Serve on 240,000 cores for model serving, achieving a peak throughput of 1.37 million transactions per second during high-demand periods.
- Uber: Leverages Ray to rapidly pretrain, fine-tune, and evaluate large language models (LLMs).
- Instacart: Uses Ray to run deep learning workloads 12 times faster, reduce costs by 8 times, and train models on 100 times more data. (Ray)
- Samsara: Implemented Ray to scale the training of deep learning models to hundreds of millions of inputs, accelerating deployment and cutting inference costs by 50%.
- Cohere: Utilises Ray to simplify the development of scalable distributed programs for large language model pipelines. (Ray)

Prerequisite: AI-101

- **AI-500: AI Ethics and Governance: Principles and Practices**

  In the rapidly evolving field of artificial intelligence, understanding the ethical implications and governance frameworks is crucial. This course delves into the core principles of AI ethics, including fairness, transparency, accountability, and privacy. Participants will explore the societal impacts of AI across various sectors, examine international and national governance structures, and learn strategies for integrating ethical considerations into AI design and deployment. Through case studies and a capstone project, students will gain practical skills to navigate and address ethical challenges in AI, preparing them to lead responsible AI initiatives in their organisations.

- **AI-501: Distributed Machine Learning**

  Generative AI tools like ChatGPT, Gemini, and DALL-E have revolutionised our professional landscape. This hands-on course guides you through the exciting distributed process of building and training AI models using Python and the versatile, open-source PyTorch and Ray frameworks. You'll delve into the core concepts of Generative Adversarial Networks (GANs), Transformers, Large Language Models (LLMs), variational autoencoders, diffusion models, and more. Along the way, you'll gain practical experience and a deep understanding of these cutting-edge technologies.

Learning Repo: https://github.com/panaversity/genai-with-pytorch

Prerequisite: AI-101, AI-461

- **AI-502: Customising Open Source LLMs**

  This comprehensive course is designed to guide learners through the process of fine-tuning open-source Large Language Models (LLMs) such as Meta LLaMA 3 using PyTorch, with a particular emphasis on cloud-native training and deployment. The course covers everything from the fundamentals to advanced concepts, ensuring students acquire both theoretical knowledge and practical skills.

  The journey begins with an introduction to LLMs, focusing on their architecture, capabilities, and the specific features of Meta LLaMA 3. Next, the course dives into PyTorch fundamentals, teaching students how to perform basic operations with tensors and build simple neural networks. This foundation is crucial for understanding the mechanics behind LLMs. Data preparation is a crucial aspect of training models. The course covers comprehensive data collection and preprocessing techniques, such as tokenization and text normalisation. These steps are essential for preparing datasets suitable for fine-tuning LLMs like Meta LLaMA 3. Through practical exercises, students learn how to handle and preprocess various types of text data, ensuring they can prepare their datasets for optimal model performance.

  Fine-tuning Meta LLaMA 3.2 with PyTorch forms a significant part of the course. Students will delve into the architecture of Meta LLaMA 3, learn how to load pre-trained models, and apply fine-tuning techniques. The course covers advanced topics such as regularisation and optimization strategies to enhance model performance. Practical sessions guide students through the entire fine-tuning process on custom datasets, emphasising best practices and troubleshooting techniques.

  A critical aspect of this course is its focus on cloud-native training and deployment using Nvidia NIM. Furthermore, students learn how to deploy models using Docke/Rancherr and Kubernetes, set up monitoring and maintenance tools, and ensure their models are scalable and efficient.

  To round off the learning experience, the course includes an in-depth segment on exporting models for inference and building robust inference pipelines. Students will deploy models on cloud platforms, focusing on practical aspects of setting up monitoring tools to maintain model performance and reliability.

  The course culminates in a capstone project, where students apply all the skills they have learned to fine-tune and deploy Meta LLaMA 3 on a chosen platform. This project allows students to demonstrate their understanding and

proficiency in the entire process, from data preparation to cloud-native deployment.

Learning Repo:

https://github.com/panaversity/learn-fine-tuning-llms

Prerequisite: AI-101, AI-461, AI-501

- **AI-651: Advanced Cloud Native and Distributed AI Computing**

  Master Kubernetes, Ray, Terraform, and GitHub Actions to deploy your AI pipelines, APIs, microservices, and open source models in the cloud. We will cover distributed system design involving creating AI systems that are distributed across multiple nodes, focusing on scalability, fault tolerance, consistency, availability, and partition tolerance.

  Certifications:

  - Certified Kubernetes Application Developer (CKAD)
  - HashiCorp Certified: Terraform Associate

  Learning Repo: https://github.com/panaversity/learn-kubernetes

  Prerequisite: AI-101, AI-301, AI-461

III. Vertical Specialization Level (Optional)

Students will have the option of selecting one of the following specialisations, details available at the end of FAQs:

1. **AI-701: Healthcare and Medical Agentic AI**
2. **AI-702: Web3, Blockchain, and Agentic AIIntegration**
3. **AI-703: Metaverse, 3D, and Agentic AI Integration**
4. **AI-704: Agentic AI for Accounting, Finance, and Banking**
5. **AI-705: Agentic AI for Engineers**
6. **AI-707: Agentic AI for Sales and Marketing Specialization**
7. **AI-708: Agentic AI for Automation and Internet of Things (IoT)**
8. **AI-709: Agentic AI for Cyber Security**

Zia Khan, CEO Panaversity
MSE, MBA, MAC, MA, CPA, CMA
https://www.linkedin.com/in/ziaukhan/

Common Questions (FAQs) with Detailed Answers

1. **What is Agentic AI? and how does it differ from Generative AI and classical Predictive AI.**

   Agentic AI refers to systems that act as autonomous “agents”—they don’t just generate outputs or predict outcomes; they actively make decisions, plan, and take actions in their environment to achieve specific goals.

   ### Key Differences:

   - **Agentic AI:**
     - **Autonomy & Action:** Operates independently, dynamically planning and executing tasks (think of a robot deciding how to navigate a warehouse).
     - **Interaction:** Can interact with other agents or humans, adapting to changing conditions in real time.

    - **Goal-Oriented:** Driven by objectives that require both decision-making and iterative feedback.
- **Generative AI:**
    - **Content Creation:** Specializes in producing new content (like text, images, or music) by learning patterns from large datasets (e.g., ChatGPT or DALL-E).
    - **Reactive Generation:** Generates output based on a given prompt without inherent planning or long-term goal management.
- **Classical Predictive AI:**
    - **Forecasting:** Focuses on predicting future events or trends from historical data (e.g., forecasting sales or weather).
    - **Static Models:** Often use statistical or machine learning models that produce probabilities or forecasts, but they don't make independent decisions or act upon the predictions.

In summary, while generative and predictive AIs are excellent at creating content and forecasting outcomes, agentic AI goes a step further by autonomously interacting with and influencing its environment.

Key Differences Summarized:

Feature	Agentic AI	Generative AI	Classical Predictive AI
Primary Goal	Achieve specific goals through autonomous action	Create new, original content	Predict future outcomes or classify data
Autonomy Level	High	Low to Moderate	Low
Environment Interaction	Direct and active	Limited or indirect (focused on data creation)	Limited or indirect (focused on data analysis)
Action Taking	Yes	Potentially, but not the primary focus	Typically No
Goal Setting	Can set and pursue goals	Typically requires external prompting	Typically does not set goals
Adaptation	Adapts to changing environments in real-time	Can adapt its outputs to new inputs in real-time	Adapts based on new data during retraining

2. **What is a Certified Agentic and Robotic AI Engineer?**
   A Certified Agentic and Robotic AI Engineer is a professional skilled in the development of autonomous, AI-driven systems that can act and make

decisions independently (agentic AI) and physical systems that interact with the physical world (robotic or physical AI). This certification program, offered by Panaversity, equips participants with expertise in building and deploying autonomous software agents, humanoid robots, and fine-tuning large language models (LLMs) for specific applications.

The program covers skills across several domains:

- **Agentic AI**: Focus on AI systems that can autonomously learn, perceive, reason, and act, including multi-agent AI systems, AI-powered SaaS solutions, and knowledge graphs.
- **Humanoid Robotics and Physical AI**: Training on designing, simulating, and deploying robots capable of interacting with humans naturally using platforms like ROS 2 and NVIDIA Isaac.
- **Cloud Native and Distributed Computing**: Includes building scalable AI-powered microservices, leveraging Docker/Rancher, Kubernetes, and Ray for cloud-native and distributed applications.
- **LLM Fine-Tuning**: Students learn to customize LLMs, using tools like PyTorch and FastAI, for specific applications.
- **AI Ethics and Governance**: To prepare participants for real-world deployment, the program includes principles and practices for responsible AI use.

This comprehensive curriculum enables students to become leaders in the $100 trillion AI-driven industrial revolution, blending technical prowess with ethical awareness and a future-focused approach to the development and application of AI in both virtual and physical domains￼.

3. **How valuable can the Certified Agentic and Robotic AI Engineer Program be in the new age of Agentic AI?**

   In brief, a **Certified Agentic and Robotic AI Engineer Program** can be **highly valuable** in the new age of Agentic AI. This is because:

   - **Specialized Skills:** Agentic AI requires a unique skillset encompassing AI model understanding, autonomous system design, and potentially robotics integration. The program likely provides this specialized knowledge.
   - **High Demand:** As Agentic AI becomes more prevalent, the demand for professionals with these specific skills will increase significantly, making certified individuals highly sought after by employers.
   - **Validation of Expertise:** Certification serves as tangible proof of acquired skills and expertise, enhancing credibility and marketability in a rapidly evolving field.

- **Career Advancement:** Such a program can open doors to exciting career opportunities in autonomous systems, robotics, AI solutions architecture, and more.
- **Competitive Edge:** Possessing this certification would give engineers a significant competitive advantage in the job market, positioning them as leaders in the Agentic AI space.

4. **What is the potential for Certified Agentic and Robotic AI Engineers to start their own companies and become successful startup founders?**

   The potential is enormous. Certified Agentic and Robotic AI Engineers — armed with verified expertise in next-gen automation, AI integration, and robotics—are uniquely positioned to launch disruptive startups. With markets in robotics and AI expanding rapidly and venture capital flowing into innovative technologies, these engineers can leverage open-source frameworks, cloud resources, and their hands-on experience to quickly validate ideas, reduce costs, and scale solutions. In short, they have all the tools needed to become successful startup founders in a booming, transformative industry.

5. **Why don't we use TypeScript (Node.js) to develop AI Agents instead of using Python?**
   We will not use Typescript in AI powered API development because Python is a priority with the AI community when working with AI and if any updates come in libraries they will first come for Python. Python is always a better choice when dealing with AI and API.

   - **Python is the de facto standard for AI Development**.
   - TypeScript is a more modern language that is gaining popularity for Web Development, but Python is more widely used and has a larger ecosystem of libraries and frameworks available, especially for AI.
   - TypeScript is used for web user interfaces, while Python is used for developing AI Agents and APIs.
   - Python is a more commonly used language for AI and Agent development, and it has a larger ecosystem of libraries and frameworks available for these purposes.
   - TypeScript is a modern language that is becoming increasingly popular for API development also, but it is still not as widely used as Python, especially for AI applications and development.

6. **What is OpenAI Agents SDK?**

   The OpenAI Agents SDK is a framework designed for developers to build, deploy, and manage autonomous agents that leverage OpenAI's language models. It provides a set of tools and APIs to create intelligent, conversational agents that can handle multi-turn interactions, manage context, and perform complex tasks. The SDK is built with flexibility in mind, allowing for extensive customization and integration into broader applications, making it suitable for both rapid prototyping and production-grade solutions.

   In brief, here are its main features:

   - **Seamless Integration:** Offers robust APIs to integrate OpenAI's models directly into your applications.
   - **Customizability:** Allows developers to tailor agents for specific tasks and fine-tune behavior based on application needs.
   - **Multi-Turn Conversation:** Supports complex dialogues by maintaining context over multiple interactions.
   - **Scalability:** Designed to support everything from prototyping to enterprise-grade applications.
   - **Rich Ecosystem:** Backed by extensive documentation and community support, facilitating easier troubleshooting and innovation.

   These capabilities make it a powerful toolkit for building advanced, context-aware AI solutions.

7. **Why do we use OpenAI Agents SDK and Microsoft AutoGen for developing AI Agents as compared to other frameworks like AutoGen, CrewAI and LangGraph?**

   We're not avoiding AutoGen, CrewAI and LangGraph; we're prioritizing our progressive learning approach and our Agentia World vision. We believe that OpenAI Agents SDK and Cloud Native Technologies are more aligned with our vision of AI-to-AI communication on a global scale. For progressive learning they are also a better combination.

   While AutoGen, CrewAI, and LangGraph each have their strengths, for a vision as expansive as Agentia—with ubiquitous intelligence, complex dialogues, and real-world physical integration— OpenAI Agents SDK offers the robust, scalable, and flexible foundation needed to bring this world to life.

It provides a perfect stepping stone in our progressive learning journey toward Agentia.

8. **What is Model Context Protocol (MCP)?**

   Model Context Protocol (MCP) is a standardized framework designed to manage and transmit contextual information between AI models in multi-agent systems. In brief, MCP:

   - **Defines Context Structure:** Establishes a common format for context data—such as prior interactions, metadata, and task instructions — ensuring consistency across models.
   - **Enhances Coherence:** Enables different agents to maintain awareness of conversation history and operational parameters, which improves the quality and coherence of multi-turn interactions.
   - **Facilitates Collaboration:** Supports smooth communication among various models or agents, allowing them to work together effectively in complex workflows.

   Overall, MCP helps in creating a seamless, coordinated AI ecosystem where models share relevant context to produce more informed and accurate responses.

9. **Why don't we use Flask or Django for API development instead of FastAPI?**

   - **FastAPI is a newer and more modern framework than Flask or Django.** It is designed to be fast, efficient, and easy to use. FastAPI is also more scalable than Flask or Django, making it a better choice for large-scale projects.
   - **FastAPI is also more feature-rich than Flask or Django.** It includes several built-in features that make it easy to develop APIs, such as routing, validation, and documentation.
   - **Overall, FastAPI is a better choice for API development than Flask or Django.** It is faster, more scalable, and more feature-rich.

10. **Why do we need to learn Cloud Native technologies in a Agentic AI program?**

    Cloud Native technologies are essential for developing and deploying Agentic AI applications because they provide a scalable and reliable platform for hosting and managing complex workloads.

- Cloud Native computing offers a vast pool of resources that can be provisioned on demand, which is ideal for Agentic AI applications that can be computationally intensive.
- Cloud Native approach providers offer a wide range of services that can be used to support Agentic AI applications, including storage, computing, networking, and workflows.
- Cloud Native services are typically more cost-effective than on-premises infrastructure, which can be a significant advantage for Agentic AI applications that are often used for large-scale projects.

The Certified Agentic and Robotic AI Engineering Program teaches you how to use cloud native services, including containers, Dapr, and Kubernetes, to deploy your applications to the cloud. You will also learn how to use **Docker/Rancher containers** to package and deploy your applications, and how to manage your cloud infrastructure.

By the end of the program, you will be able to:

- Use Docker/Rancher containers to package and deploy your applications
- Develop and deploy Agentic AI applications to the cloud
- Manage your cloud infrastructure

11. What is the purpose of Docker/Rancher Containers and what are the benefits of deploying them with Docker Compose/Rancher Desktop, and Kubernetes?

- **Docker Containers** are a way to package software into a single unit that can be run on any machine, regardless of its operating system. It is used to create a Dockerfile, which is a text file that describes how to build a Docker image. The image is then used to create a container, which is a running instance of the image. This makes them ideal for deploying applications on a variety of platforms, including cloud-based services.
- **Docker Compose/Rancher Desktop** is a tool provided by Docker that allows you to define and manage multi-container Docker applications locally. It enables you to use a YAML file to configure the services, networks, and volumes needed for your application's setup. With Docker Compose/Rancher Desktop, you can describe the services your application requires, their configurations, dependencies, and how they should interact with each other, all in a single file. This makes it easier to orchestrate complex applications locally composed of multiple interconnected containers.

- **Kubernetes** is a container orchestration system that automates the deployment, scaling, and management of containerized applications. It allows you to run multiple containers on a single machine or across multiple machines. It is an open source and can be deployed in your data centre or the cloud.

12. What is the purpose of learning to develop APIs in an Agentic AI program?

APIs (Application Programming Interfaces) are used to connect different software applications and services together. They are the building blocks of the internet and are essential for the exchange of data between different systems.

In the Certified Agentic and Robotic AI Engineering Program, students will learn to develop APIs not just as a backend but also as a **product** itself. In this model, the API is at the core of the business's value.

- APIs are used to make it possible for different software applications to communicate with each other.
- APIs are used to access data from a remote server.
- APIs are used to create new services or applications that are integrated with existing systems.
- APIs are used to improve the security of applications by providing a way to control access to data.
- By learning to develop APIs, students will gain the skills necessary to create powerful and efficient software applications that can be used to solve a variety of business problems.

13. What is the purpose of using Python-based FastAPI and related technologies?

In this Program, students will learn how to use Python-based FastAPI as a core library for API development.

- FastAPI is a high-performance, lightweight, and easy-to-use framework for building APIs.
- It is designed to be fast, scalable, and secure.
- FastAPI is compatible with a wide range of programming languages and frameworks, making it a good choice for developers with different skill sets.
- Students will also learn about the following related technologies:

- **Pydantic:** Pydantic is a Python library that helps to improve the quality of your code by checking for errors and potential problems.
- **GQL:** The Graph Query Language (GQL) is an international standard published by the International Organization for Standardization (ISO) as ISO/IEC 39075:2024. GQL is designed for querying property graphs and is the first database query language ISO has published since SQL in 1987. It defines data structures and operations for creating, accessing, querying, maintaining, and controlling property graphs, providing a standardised way to manage graph data across different implementations.
- **Neo4j:** Neo4j is a powerful, open-source graph database designed to store and manage highly connected data. Unlike traditional relational databases, Neo4j uses a graph model, where data is represented as nodes, relationships, and properties. This structure is ideal for applications where relationships between data points are as important as the data itself.

By the end of the program, students will be able to use Python-based FastAPI to develop APIs that are fast, scalable, and secure.

14. What are the benefits of using Docker Containers for development, testing, and deployment?

Docker Containers are a fundamental building block for development, testing, and deployment because they provide a consistent environment that can be used across different systems. This eliminates the need to worry about dependencies or compatibility issues, and it can help to improve the efficiency of the development process. Additionally, Docker Containers can be used to isolate applications, which can help to improve security and make it easier to manage deployments.

15. What is the advantage of using open Docker, Dapr, and Kubernetes technologies instead of using AWS, Azure, or Google Cloud technologies?

Using open-source technologies like Docker, Dapr, and Kubernetes offers several advantages over relying solely on proprietary cloud services from AWS, Azure, or Google Cloud. Here's a detailed comparison:

Advantages of Using Docker, Dapr, and Kubernetes (Open Technologies)

1. Portability and Flexibility:
   - Vendor Agnostic: These tools are cloud-agnostic, meaning you can run your applications on any cloud provider or on-premises infrastructure without being locked into a specific vendor.
   - Ease of Migration: Applications packaged in Docker containers can easily be moved across different environments, and Kubernetes provides a consistent orchestration layer, ensuring seamless transitions.

2. Cost Efficiency:
   - Avoid Vendor Lock-In: Being locked into a single cloud provider can lead to higher costs over time. Using open technologies allows you to leverage competitive pricing from multiple providers or even use on-premises resources.
   - Optimised Resource Utilisation: Kubernetes helps in efficiently managing resources through automated scaling and load balancing, potentially reducing costs.

3. Community and Ecosystem:
   - Open Source: These tools are backed by strong, active open-source communities that continuously improve the software, provide support, and share best practices.
   - Ecosystem: A rich ecosystem of tools and integrations is available, providing flexibility to choose the best components that fit your specific needs.

4. Standardisation and Consistency:
   - Unified Platform: Using Docker for containerization, Dapr for service invocation, storage, and workflows, and Kubernetes for orchestration, and provides a standardised way to deploy, manage, and scale applications across different environments.
   - Consistency Across Environments: These tools ensure that your development, staging, and production environments are consistent, reducing bugs and deployment issues.

5. Customization and Control:
   - Full Control: Open-source tools give you complete control over your infrastructure and deployment pipelines. You can customise and extend the functionality to suit specific requirements.
   - Transparency: Access to the source code means you can audit and modify the software to meet your security and compliance needs.

Advantages of Using AWS, Azure, or Google Cloud Technologies

1. Managed Services:

- **Ease of Use:** Cloud providers offer a wide range of managed services that abstract away the complexity of setting up and managing infrastructure. This can save time and reduce operational overhead.
- Integrated Solutions: These platforms provide integrated services and tools, such as databases, machine learning, analytics, and monitoring, which can be easily combined to build complex applications.

2. Scalability and Reliability:
- Global Infrastructure: Cloud providers have extensive global infrastructure, ensuring high availability, redundancy, and low latency.
- Auto-Scaling: Advanced auto-scaling capabilities can dynamically adjust resources to meet changing demands, ensuring optimal performance.

3. Security and Compliance:
- Built-In Security: Cloud providers offer robust security features, including identity and access management, encryption, and compliance certifications, helping to protect your data and meet regulatory requirements.
- Automatic Updates: Managed services often include automatic updates and patches, reducing the risk of security vulnerabilities.

4. Innovation and Support:
- Cutting-Edge Technology: Major cloud providers continuously innovate and introduce new services and features, allowing you to leverage the latest technologies without significant investment.
- Support and SLA: Comprehensive support services and Service Level Agreements (SLAs) ensure that you have access to expert help and guaranteed uptime.

Conclusion

Choosing between open-source technologies like Docker, Kubernetes, and Dapr versus proprietary cloud services from AWS, Azure, or Google Cloud depends on your specific needs and priorities.

- Open Technologies: Offer portability, cost efficiency, customization, and control, making them ideal for multi-cloud strategies, avoiding vendor lock-in, and having more control over your infrastructure.
- Cloud Providers: Provide ease of use, managed services, scalability, security, and access to cutting-edge technology, which can be advantageous for rapid development, scaling, and leveraging advanced services.

In many cases, a hybrid approach that combines the strengths of both open-source tools and cloud provider services can provide the best of both worlds, allowing you to optimise for cost, flexibility, and innovation.

16. **Why in this program are we not learning to build LLMs ourselves? How difficult is it to develop an LLM like ChatGPT or Google's Gemini?**

Developing an LLM like ChatGPT or Google Gemini is extremely difficult and requires a complex combination of resources, expertise, and infrastructure. Here's a breakdown of the key challenges:

Technical hurdles:

Massive data requirements: Training these models requires an immense amount of high-quality data, often exceeding petabytes. Compiling, cleaning, and structuring this data is a monumental task.
Computational power: Training LLMs demands incredible computational resources, like high-performance GPUs and specialised AI hardware. Access to these resources and the ability to optimise training processes are crucial.
Model architecture: Designing the LLM's architecture involves complex decisions about parameters, layers, and attention mechanisms. Optimising this architecture for performance and efficiency is critical.
Evaluation and bias: Evaluating the performance of LLMs involves diverse benchmarks and careful monitoring for biases and harmful outputs. Mitigating these biases is an ongoing research challenge.

Resource and expertise:

Team effort: Developing an LLM like ChatGPT or Google Gemini requires a large team of experts across various disciplines, including AI researchers, machine learning engineers, data scientists, and software developers.
Financial investment: The financial resources needed are substantial, covering costs for data acquisition, hardware, software, and talent. Access to sustained funding is critical.

Additionally:

Ethical considerations: LLMs raise ethical concerns like potential misuse, misinformation, and societal impacts. Responsible development and deployment are crucial.
Rapidly evolving field: The LLM landscape is constantly evolving, with new research, models, and benchmarks emerging. Staying abreast of these advancements is essential.

Therefore, while ChatGPT 4 and Google Gemini have made impressive strides, developing similar LLMs remains a daunting task accessible only to a handful of organisations with the necessary resources and expertise.

In simpler terms, it's like building a skyscraper of knowledge and intelligence. You need the right materials (data), the right tools (hardware and software), the right architects (experts), and a lot of hard work and attention to detail to make it stand tall and function flawlessly.

Developing similar models would be a daunting task for individual developers or smaller teams due to the enormous scale of resources and expertise needed. However, as technology progresses and research findings become more accessible, it might become incrementally more feasible for a broader range of organisations or researchers to work on similar models, albeit at a smaller scale or with fewer resources. At that time we might also start to focus on developing LLMs ourselves.

To sum up, the focus of the program is not on LLM model development but on applied Cloud GenAI Engineering (GenEng), application development, and fine-tuning of foundational models. The program covers a wide range of topics including Python, GenAI, Microserices, API, Database, Cloud Development, and DevOps, which will give students a comprehensive understanding of generative AI and prepare them for careers in this field.

17. **Business wise does it make more sense to develop LLMs ourselves from scratch or use LLMs developed by others and build applications using these tools by using APIs and/or fine-tuning them?**

Whether it makes more business sense to develop LLMs from scratch or leverage existing ones through APIs and fine-tuning depends on several factors specific to your situation. Here's a breakdown of the pros and cons to help you decide:

Developing LLMs from scratch:

Pros:

Customization: You can tailor the LLM to your specific needs and data, potentially achieving higher performance on relevant tasks.
Intellectual property: Owning the LLM allows you to claim intellectual property rights and potentially monetize it through licensing or other means.
Control: You have full control over the training data, algorithms, and biases, ensuring alignment with your ethical and business values.

Cons:

High cost: Building and training LLMs require significant technical expertise, computational resources, and data, translating to high financial investment.
Time commitment: Developing an LLM is a time-consuming process, potentially delaying your go-to-market with your application.
Technical expertise: You need a team of highly skilled AI specialists to design, train, and maintain the LLM.

Using existing LLMs:

Pros:

Lower cost: Leveraging existing LLMs through APIs or fine-tuning is significantly cheaper than building them from scratch.
Faster time to market: You can quickly integrate existing LLMs into your applications, accelerating your launch timeline.
Reduced technical burden: You don't need a large team of AI specialists to maintain the LLM itself

Cons:

Less customization: Existing LLMs are not specifically designed for your needs, potentially leading to lower performance on some tasks.
Limited control: You rely on the data and biases of the existing LLM, which might not align with your specific requirements.
Dependency on external parties: You are dependent on the availability and maintenance of the LLM by its developers.

Here are some additional factors to consider:

The complexity of your application: Simpler applications might benefit more from existing LLMs, while highly complex ones might require the customization of a dedicated LLM.
Your available resources: If you have the financial and technical resources, developing your own LLM might be feasible. Otherwise, existing options might be more practical.
Your competitive landscape: If your competitors are using LLMs, you might need to follow suit to remain competitive.
Ultimately, the best decision depends on your specific needs, resources, and business goals. Carefully evaluating the pros and cons of each approach will help you choose the strategy that best aligns with your success.

18. What is Agentic Memory?

Agentic Memory refers to the memory systems used by agentic AI — autonomous systems that reason, act, and adapt—to store and retrieve information needed for tasks. It enables agents to maintain context, learn from past interactions, and make informed decisions, much like human memory supports daily activities.

Difference Between Short-Term and Long-Term Memory

Short-Term Memory:

- **What It Is**: Temporary storage for immediate context, like the current conversation or task.
- **How It Works**: Held in the agent's prompt or context window (e.g., 128k tokens for advanced LLMs), reset after each session.
- **Example**: An agent remembers your last question ("What's the weather?") to suggest clothing now.
- **Limit**: Constrained by context window size—too much data, and older info gets cut off.

Long-Term Memory:

- **What It Is**: Persistent storage for information across sessions, like facts or user preferences.
- **How It Works**: Saved in external systems (e.g., vector databases like Pinecone, graph DBs like Neo4j) and retrieved as needed via tools or embeddings.
- **Example**: An agent recalls you like spicy food from a chat last month to plan a meal.
- **Advantage**: Not limited by session duration or context size—scales indefinitely.

Brief Comparison

- **Duration**: Short-term lasts for one interaction; long-term spans days, months, or years.
- **Storage**: Short-term is in-memory (prompt); long-term is external (databases).

- **Use**: Short-term for quick context (e.g., chat flow); long-term for deep knowledge (e.g., user history).

In agentic AI, short-term memory keeps things running smoothly now, while long-term memory builds intelligence over time—both critical for autonomy and effectiveness.

19. What is Agentic RAG?

RAG (Retrieval-Augmented Generation) is a technique that enhances AI language models by combining **retrieval** of relevant external information with **generation** of responses.

- **How It Works**: When asked a question, the AI first retrieves related data (e.g., from a database or documents using embeddings) and then uses that info to generate a more accurate, context-rich answer.
- **Example**: Asked "What's the capital of Brazil?", it retrieves "Brazil's capital is Brasília" from a source and generates a response based on that.
- **Why It Matters**: It reduces reliance on pre-trained knowledge, making answers more up-to-date and factual.

In brief, RAG is like giving an AI a library to check before it speaks, boosting accuracy and relevance.

Agentic RAG is an advanced form of RAG where an AI agent doesn't just passively retrieve and generate responses but actively reasons, plans, and uses tools to fetch or process information from external sources. It combines retrieval (pulling relevant data) with generation (creating answers) in a proactive, autonomous way.

- **How It Works**: An agent queries a knowledge base (e.g., vector database), retrieves context (e.g., via embeddings), and uses reasoning to refine or act on it—often via tool calls (e.g., web search).
- **Example**: Asked "What's the best laptop in 2025?", it searches recent articles, evaluates options, and generates a tailored recommendation.
- **Difference from Basic RAG**: Traditional RAG retrieves and responds; Agentic RAG adds decision-making and action (e.g., "Should I search more?").

In brief, Agentic RAG makes AI smarter by letting it think and explore, not just fetch and talk, enhancing its ability for complex, knowledge-driven tasks.

20. Do we need to use Design Thinking and BDD for designing AI Agents?

Design Thinking and Behavior-Driven Development (BDD) are methodologies that can greatly enhance the process of designing AI Agents, though they are not strictly necessary. Here's how each can be beneficial:

Design Thinking
Design Thinking is a user-centred approach to innovation and problem-solving that involves understanding the user, challenging assumptions, redefining problems, and creating innovative solutions through iterative prototyping and testing.

Benefits for AI Agents:
1. User-Centric Focus: Ensures that the AI solutions are tailored to the actual needs and pain points of users.
2. Empathy: Helps in understanding the context and environment in which the AI will be used, leading to more relevant and effective solutions.
3. Iterative Development: Encourages continuous testing and refinement of ideas, leading to more robust and user-friendly AI models.
4. Collaboration: Promotes cross-disciplinary collaboration, which can bring diverse perspectives and expertise to the design process.

Behaviour-Driven Development (BDD)
BDD is a software development methodology that encourages collaboration between developers, QA, and non-technical stakeholders through the use of natural language descriptions of the desired behaviour of the software.

Benefits for AI Agents:
1. Clear Requirements: Ensures that the requirements are clearly understood and agreed upon by all stakeholders.
2. Testable Scenarios: Facilitates the creation of testable scenarios that can validate the AI's behaviour against the expected outcomes.
3. Documentation: Provides clear and comprehensive documentation of the AI's intended behaviour, which is useful for future maintenance and enhancements.
4. Alignment: Ensures that the development stays aligned with business goals and user expectations.

Application in Designing AI Agents

- Design Thinking:

- Map out the user journey and identify critical interaction points where the AI Agent will provide value.
- Prototype and test the agent's interactions in various environments to ensure robustness and usability.
- Use empathy maps and personas to better understand and anticipate user needs and behaviours.

- BDD:
- Write behaviour scenarios that describe how the AI Agent should react in different situations.
- Develop tests that simulate these scenarios to verify the agent's decision-making and learning processes.
- Continuously refine the agent's behaviour based on test results and user feedback.

While not strictly necessary, Design Thinking and BDD can significantly enhance the design and development process of AI Agents by ensuring a user-centred approach, clear requirements, and continuous improvement through iterative testing and feedback. These methodologies help in creating more effective, reliable, and user-friendly AI solutions.

21. What is Ray, how do we use it in developing AI?

Ray is an open-source distributed computing framework specifically designed to simplify the development and scaling of AI applications and other data-intensive tasks. It provides a flexible platform for building distributed applications that can scale from a single machine to a large cluster, making it ideal for AI workloads that demand substantial computational power, such as machine learning training, reinforcement learning, data processing, and model serving.

Here's how Ray is used in AI development and why it's advantageous:

a. Distributed Computing for AI Workloads

Ray's primary feature is its ability to distribute computation across multiple CPUs and GPUs, allowing developers to run tasks in parallel and manage complex workflows efficiently. This is particularly beneficial in AI development, where training large models or processing vast datasets can be time-consuming and resource-intensive.

- Parallelization: Ray enables developers to parallelize tasks, such as data preprocessing or model training, across multiple nodes, reducing runtime and maximising resource usage.
- Scaling: AI tasks can be distributed across clusters, allowing them to scale up to cloud-based environments or large on-premises setups with minimal code changes.

b. Simplified Development with Python API

Ray is designed to work seamlessly with Python, which is the most popular language in AI and machine learning. Its intuitive API makes it easy for developers to turn existing Python functions into distributed tasks, which reduces the complexity of parallel programming.

- Ease of Use: Developers can use familiar Python code without deep knowledge of distributed systems, thanks to Ray's ability to automatically handle task scheduling, data sharing, and fault tolerance.
- Flexible Abstractions: Ray provides high-level abstractions, such as tasks and actors, to manage distributed computations efficiently. This flexibility allows developers to easily implement complex parallel workflows.

c. Frameworks Built on Ray for Specialized AI Tasks

Ray includes several libraries tailored for specific AI use cases, simplifying complex tasks and streamlining workflow integration:

- Ray Tune: Used for hyperparameter tuning, which is crucial in optimizing AI models. Ray Tune distributes the tuning process, allowing multiple hyperparameter configurations to be evaluated in parallel, which speeds up model optimization significantly.
- Ray RLlib: A library for reinforcement learning (RL) that simplifies the implementation and scaling of RL algorithms. It supports a range of algorithms out of the box and can run on single machines or distributed clusters, making it suitable for large-scale RL projects.
- Ray Serve: Designed for model serving, it provides a scalable, flexible way to deploy trained AI models as services that can handle multiple requests in real-time, making it ideal for production applications.

d. Use Cases in AI Development

Ray is applicable across various stages of AI development, including:

- Data Processing and Preprocessing: Ray can distribute data loading, transformation, and cleaning tasks, which are often bottlenecks in AI workflows, across a cluster. This accelerates the process and allows for more efficient handling of large datasets.
- Model Training: With Ray, model training can be distributed across multiple GPUs or nodes, reducing training time and enabling the handling of larger models or datasets.

- Hyperparameter Optimization: Ray Tune simplifies running multiple experiments to find the best model configurations, automating the tuning process and improving model performance.
- Reinforcement Learning: Ray RLlib supports scaling RL algorithms, which are often computationally intensive and require significant parallelization to achieve optimal performance.
- Model Deployment: Ray Serve allows developers to deploy models for real-time predictions, handling requests with high throughput and low latency.

5. Advantages of Using Ray in AI
   - Scalability: Ray's distributed architecture makes it easy to scale AI workloads, whether they are running on a laptop, a multi-GPU setup, or a cloud cluster.
   - Efficiency: By enabling distributed processing and parallelism, Ray reduces the computational time needed for data processing, model training, and serving, leading to faster iteration cycles.
   - Unified Framework: Ray integrates various stages of the AI pipeline (training, tuning, deployment) under a single platform, making it easier for teams to manage and scale AI workflows consistently.
   - Cost Optimization: By optimising resource usage across clusters, Ray helps minimise infrastructure costs, especially in cloud environments where resources are billed based on usage.

22. When Fine-Tuning Open-Source Large Language Models, how is Ray and PyTorch important and what role do these libraries play?

Fine-tuning open-source Large Language Models (LLMs) like Meta's LLaMA or other foundational models is a complex and resource-intensive task that involves adapting a pre-trained model to perform better on specific tasks or domains. Ray and PyTorch play essential roles in this process by providing tools to efficiently handle the heavy computational requirements of fine-tuning, making it scalable, flexible, and streamlined.

Here's how each of these libraries contributes to fine-tuning LLMs:

a. Role of PyTorch in Fine-Tuning LLMs

PyTorch is a deep learning framework widely used for developing and fine-tuning neural networks, including LLMs. It provides the foundational tools for setting up, training, and optimising neural network models.

- **Flexible Model Building**: PyTorch's dynamic computation graph, also known as define-by-run, allows researchers and developers to easily

customise the LLM architecture or adjust training configurations to fit specific tasks. This is especially useful when experimenting with different model architectures or training techniques.

- **Pre-Trained Model Integration**: PyTorch integrates seamlessly with the Hugging Face Transformers library, which provides access to many pre-trained LLMs. This integration simplifies the fine-tuning process, allowing developers to load pre-trained models, prepare them for task-specific data, and modify them as needed.
- **GPU Acceleration**: PyTorch supports GPU and distributed training, crucial for handling the immense computational requirements of fine-tuning LLMs. Large language models have millions to billions of parameters, so GPU acceleration dramatically reduces training time.
- **Tools for Optimization**: PyTorch offers numerous built-in optimizers and training utilities, such as AdamW (an optimizer widely used in NLP), learning rate schedulers, and gradient clipping. These tools help ensure that fine-tuning is efficient, effective, and stable.

b. Role of Ray in Scaling and Distributing the Fine-Tuning Process

Ray is critical for scaling the fine-tuning process across multiple GPUs or even across distributed clusters, which is essential for handling the large-scale computations required for LLMs.

- **Distributed Training**: Ray makes it easy to distribute PyTorch workloads across multiple CPUs or GPUs by turning Python functions into remote functions that can be executed concurrently. This is useful for model parallelism, where different parts of the model are processed on different devices, or for data parallelism, where the same model is trained on different data batches across multiple devices.
- **Hyperparameter Tuning with Ray Tune**: Fine-tuning often involves adjusting hyperparameters like learning rate, batch size, or dropout rates. Ray Tune, a library within Ray, automates this process by running multiple configurations simultaneously, speeding up the search for the optimal settings.
- **Efficient Resource Utilisation**: Ray helps manage computational resources efficiently by automatically handling task scheduling, load balancing, and fault tolerance. This allows developers to focus on model training without manually managing cluster infrastructure.
- **Integration with PyTorch Distributed**: Ray seamlessly integrates with PyTorch's native distributed framework (such as torch.distributed). This allows developers to leverage Ray's parallelism tools along with PyTorch's model training capabilities to create efficient distributed training pipelines.

c. Combined Workflow Using PyTorch and Ray for Fine-Tuning

The ideal workflow for fine-tuning LLMs combines the strengths of both PyTorch and Ray:

- **Model Loading and Preparation**: Load the pre-trained model in PyTorch and set up training parameters.
- **Data Parallelism with Ray**: Distribute data across multiple GPUs or nodes using Ray to increase training speed.
- **Hyperparameter Tuning with Ray Tune**: Use Ray Tune to test various configurations, automatically tracking results and selecting the optimal combination.
- **Training Management and Optimization**: Use PyTorch for efficient training with distributed backpropagation and GPU utilisation, while Ray manages the orchestration across clusters.

d. Advantages of Using Ray and PyTorch Together in Fine-Tuning

- **Scalability**: Ray allows PyTorch models to scale beyond a single device or server, which is crucial for handling large-scale models and datasets.
- **Speed and Efficiency**: Distributing data processing and training across multiple devices can significantly reduce fine-tuning time, making iterative experimentation faster.
- **Flexibility**: Ray's integration with Python and PyTorch allows developers to build complex distributed training pipelines without needing to learn complex distributed systems programming.
- **Cost Efficiency**: By efficiently managing hardware resources, Ray helps reduce infrastructure costs, particularly in cloud environments where resources are billed based on usage.

Summary

Ray and PyTorch together provide a robust platform for fine-tuning large language models. PyTorch offers the deep learning capabilities required for LLM training, while Ray adds scalability and resource management, making distributed fine-tuning accessible, efficient, and effective. This combination enables developers to fine-tune powerful models on large datasets across distributed environments, paving the way for faster, scalable AI development.

23. In this course PyTorch plays a major role in the fine-tuning of open-source LLMs, why don't we use TensorFlow instead?

While TensorFlow is a powerful and widely-used deep learning framework, PyTorch offers several advantages that make them particularly well-suited for fine-tuning open-source Large Language Models (LLMs). Here's a detailed comparison highlighting why PyTorch might be preferred over TensorFlow for this specific task:

PyTorch vs. TensorFlow

a. Dynamic Computation Graphs:
- PyTorch: Uses dynamic computation graphs (define-by-run), which allow for greater flexibility and ease of debugging. This is especially useful when experimenting with new models and training strategies.
- TensorFlow: Initially used static computation graphs (define-and-run). Although TensorFlow 2.0 introduced eager execution to support dynamic graphs, PyTorch's implementation is often considered more intuitive and easier to work with for dynamic tasks.

b. Ease of Use:
- PyTorch: Known for its simplicity and clear, Pythonic code, which makes it easier to learn and use, especially for research and prototyping.
- TensorFlow: While TensorFlow 2.0 improved usability, it is still considered more complex compared to PyTorch, particularly for newcomers.

c. Community and Ecosystem:
- PyTorch: Has seen rapid adoption in the research community, leading to a rich ecosystem of tools, libraries, and community support. Libraries like Hugging Face's Transformers are built primarily for PyTorch, offering extensive support for LLMs.
- TensorFlow: Has a strong industrial presence and is widely used in production environments. However, the research community has increasingly favoured PyTorch.

d. Integration with Hugging Face:
- PyTorch: Hugging Face's Transformers library, which is a go-to for working with LLMs, is deeply integrated with PyTorch. This library provides pre-trained models, tokenizers, and utilities that simplify the process of fine-tuning LLMs.
- TensorFlow: Although Hugging Face provides TensorFlow support, the integration is not as seamless or feature-rich as it is with PyTorch.

While TensorFlow remains a powerful framework, particularly in production environments, PyTorch and provides a combination of flexibility, ease of use, and community support that make them particularly well-suited for the fine-tuning of open-source LLMs.

24. What is Physical AI?

Physical AI refers to the integration of artificial intelligence with physical entities, such as robots, that can operate and interact in the real world. This concept involves AI systems that not only process data and make decisions but also perform physical actions and understand the laws of physics.

Key Characteristics:

1. Real-World Interaction:
   - Physical AI systems can perceive their environment through sensors, process this information, and take appropriate actions using actuators.

2. Embodiment:
   - Unlike purely digital AI, Physical AI involves AI embedded in physical bodies, like humanoid robots, which can navigate and manipulate the physical world.

3. Understanding Physics:
   - These AI systems are designed to comprehend and adhere to the physical laws that govern real-world interactions, such as gravity, friction, and object dynamics.

4. Human-like Functionality:
   - Humanoid robots are a prime example of Physical AI, as they are built to perform tasks in environments designed for humans, utilising a form factor that mirrors human anatomy.

5. Data-Driven Training:
   - Physical AI leverages vast amounts of real-world data to train AI models, enabling robots to improve their performance through machine learning and interaction experiences.

Applications:
- **Healthcare**:
  - Assistive robots that help with patient care, rehabilitation, and surgery.
- **Service Industry**:
  - Robots that perform tasks such as cleaning, delivery, and customer service.
- **Manufacturing**:

- Industrial robots that assemble products, manage inventory, and ensure quality control.
- **Exploration**:
  - Robots designed for exploration in environments like space, underwater, or disaster zones.

Physical AI represents a significant shift from traditional AI applications confined to virtual environments. It aims to bridge the gap between digital intelligence and physical capability, creating systems that can understand and interact with the world in a human-like manner. This evolution has the potential to revolutionise various industries by enhancing automation, improving efficiency, and enabling new forms of human-machine collaboration.

25. What are the different specialisations offered at the end of the program and what are their benefits?

At the end of the certification program we offer eight specialisations in different fields:

AI-701: Healthcare and Medical Agentic AI: This specialisation will teach students how to use agentic and generative AI to improve healthcare and medical research. This is relevant to fields such as drug discovery, personalised medicine, and surgery planning.
Benefits:

- Learn how to use generative and agentic AI to identify diseases, develop new drugs, and personalise treatment plans.
- Gain a deeper understanding of the ethical implications of using generative AI in healthcare.
- Prepare for a career in a growing field with high demand for skilled professionals.

AI-702: Web3, Blockchain, and Agentic AI Integration
Integrating Web3, blockchain, and agentic AI technologies creates a powerful ecosystem where autonomy, security, transparency, and user control are at the forefront. These technologies combined enable:

- Secure, decentralised AI applications that protect user privacy,
- Transparent AI interactions that foster trust,
- Autonomous, scalable AI agents that operate reliably in peer-to-peer environments,
- Tokenized incentives that drive community participation, and

- Innovative ownership models for digital assets, enabling new economic opportunities.

AI-703: Metaverse, 3D, and Agentic AI Integration: This specialisation will teach students how to create and use 3D models and other immersive content manually and with generative AI. This is relevant to fields such as gaming, marketing, and architecture.
Benefits:

- Learn how to use generative AI to create realistic and immersive 3D models.
- Develop the skills necessary to work in the growing field of virtual reality (VR) and augmented reality (AR).
- Apply generative AI to solve real-world problems in areas such as product design, marketing, and education.

AI-704: Agentic AI for Accounting, Finance, and Banking: This specialisation will teach students how to integrate generative AI with Web3 and blockchain technologies. This is relevant to fields such as finance, healthcare, and supply chain management.
Benefits:

- Learn how to create smart contracts and decentralised applications (dApps).
- Gain a deeper understanding of the potential of blockchain technology and how it can be used to improve business processes.
- Develop the skills necessary to work in a rapidly growing field with high demand for skilled professionals.

AI-705: Agentic AI for Engineers: This specialisation will teach students how to use generative AI to improve engineering design and problem-solving. This is relevant to fields such as manufacturing, construction, and product development.
Benefits:

- Learn how to use generative AI to create simulations, optimize designs, and predict failures.
- Gain a deeper understanding of the engineering design process and how generative AI can be used to improve it.
- Prepare for a career in a growing field with high demand for skilled professionals.

AI-706: Agentic AI for Sales and Marketing: This specialisation will teach students how to use generative AI to improve sales and marketing campaigns. This is relevant to fields such as advertising, public relations, and customer service.
Benefits:

- Learn how to use generative AI to create personalised marketing messages, generate leads, and track campaign performance.
- Gain a deeper understanding of the latest marketing trends and how generative AI can be used to improve them.
- Prepare for a career in a growing field with high demand for skilled professionals.

AI-707: Agentic AI for Automation and Internet of Things (IoT) :

- **Provide Multi-Modal User Interface for the IoT systems:** Multimodal interaction exploits the synergic use of different modalities to optimise the interactive tasks accomplished by the users. This allows a user to use several input modes such as speech, touch, and visual to interact with IoT systems.
- **Improve efficiency and accuracy of industrial processes:** By implementing GenAI in automation and IoT systems, industries can optimise their processes, reduce manual labour, and increase productivity while ensuring higher accuracy and consistency.
- **Enhance decision-making:** GenAI can analyse vast amounts of data collected by IoT sensors to derive valuable insights, enabling businesses to make informed decisions regarding operations, maintenance, and resource allocation.
- **Personalise user experiences:** GenAI can leverage IoT data to understand user preferences and behaviours, enabling the creation of personalised experiences across smart devices and IoT-enabled systems.

AI-708: Agentic AI for Cyber Security:

- **Strengthen threat detection and response:** GenAI can be used to rapidly detect and respond to cyber threats by analysing large volumes of security data in real time, identifying anomalies, and suggesting appropriate countermeasures.
- **Enhance security monitoring and analysis:** GenAI can assist security analysts in monitoring and analysing security logs, automating threat detection, and providing insights into security risks and vulnerabilities.
- **Improve threat intelligence:** GenAI can be used to gather and analyse threat intelligence from various sources, enabling organisations to stay informed about the latest threats and trends and proactively strengthen their security posture.