# Action Items & Further Reading

- [ ] Research Ashby's Law

## 2025-06-16 Workshop

A workshop on normalizing AI enterprise architecture.
* **Organizer**: All things AI, allthingsopen.allthingsopen
* **AIE Network**: theaie.net
* **Theme**: Normalizing AI. The "NORMAL" acronym stands for **N**ew AI stack, including **O**bservability, **R**etrieval augmented generation, **M**odel management, **A**gentics, and **L**LM orchestration.

---

## How Granite and Open Source models support modular enterprise AI

*James Busche, IBM open technologies group*

This session focused on IBM's Granite models and their commitment to transparency and open-source principles.

- **IBM Granite**: The key differentiator is the transparency of the data and training methods.
  > The speaker emphasized the importance of true open source in the LLM world: "you have a legit open source license. But you also have... all the stuff that's trained in so that you know where what it was trained on. And I think that's important, especially in an age where deep seek is making a lot of... news because they as a country have a certain firewall policy about what they get into."
- **Granite 3.3**: New capabilities include reasoning, vision understanding, and speech.
- **Accessibility**: Low-parameter models can run on a laptop with low GPU requirements.
- **Features**: Supports reasoning toggling.
- **Open Source Tools**: The ecosystem around Granite includes `ollama`, `open-webui`, `anything-llm`, `continue` (VS Code extension), python venv, and the IBM Granite models themselves.
- **Ollama**: Now provides direct support for Granite models.
- **Workshop**: ibm.github.io offers a workshop on using Granite.

- **Demo**: The speaker demonstrated using `open-webui` to interact with a small Granite model. When asked "Who are the CEOs of IBM?", the model hallucinated, providing incorrect names and dates. To fix this, he used Retrieval-Augmented Generation (RAG) by uploading a simple text file containing the correct list of CEOs. When asked again, the model used the provided document to give the correct answer, demonstrating how RAG can ground a model with factual, up-to-date information.

---

## Rogue Agents, the new AI Reliability Playbook

*Atin Sanyal from Galileo (enterprise AI solutions)*

This talk presented an 8-step playbook for building reliable agentic systems and managing their inherent uncertainty. The core problem is that while LLM capabilities have increased, enterprise confidence in deploying them has decreased due to their complexity and unpredictability.

- **How to quantify the uncertainty?**
  - Multi-agent systems have no single point of evaluation or observation.
- **Agent Components**: Agents are comprised of three main parts:
  1. **Reasoning**: The "brain" that handles planning.
  2. **Memory**: Remembers past interactions to provide context.
  3. **Tools**: Deterministic functions that take action (e.g., data retrievals, API calls, orchestration).
- **Obstacles**: The primary challenges with agents are:
  - Unpredictable interactions
  - High sensitivity to prompt phrasing
  - Retrieval quality issues
  - This leads to a **compounding effect of small errors**, where minor issues early in a chain can lead to wildly incorrect final outputs.
- **How to tame rogue agents?**
  - The solution is **Integrated Observability**, an infinite loop of evaluation and improvement.

**The 8-Step Playbook**

**Play 1: Create metrics for Agents**
* Define agent quality metrics like:
* **Tool Error Rate**: Are the tools executing correctly?
* **Tool Selection Quality**: Is the agent choosing the right tool for the job?
* **Task Completion**: Is the agent making progress towards its goal?
* **Custom architectures need custom metrics**: A one-size-fits-all approach to metrics doesn't work. Evaluation systems must be flexible to allow developers to create their own evals tailored to their specific agent design.

**Play 2: Set up experimental infrastructure**
* Establish a process to:
1. **Identify** issues in production.
2. **Isolate** the specific data segments, LLMs, RAG components, or agent parameters causing the problem.
3. **Root cause** the issue in an "operation theater" or experimental environment.
* This enables effective **A/B testing** of different components (prompts, models, etc.) to find a fix before deploying it back to production.

**Play 3: Optimize prompts**
* Prompt sensitivity is a major cause of bad outcomes. The speaker noted:

> *"I honestly say that 90% of agentic engineering is prompt engineering."*
> ** **Version control is critical** for managing prompts as they evolve.*
> ** A systematic approach is needed to **build, test, and compare** prompts, such as A/B testing different prompt versions over a fixed dataset.*

**Play 4: Set up session & agent analytics for effective root causing**
* Traditional tracing that only looks at individual requests is ineffective for agents.
* **Super Tracing** is needed to debug across entire agentic sessions.
* Gain insights at 3 levels:

* **Trace Level**: A single end-to-end request (e.g., one RAG query).
* **Session/Interaction Level**: A group of traces that accomplish a specific sub-task.
* **Application Level**: The overall performance of the entire agent.
* The speaker noted that an agent can appear healthy at the trace level (all green) but still fail at the session level if a sub-task had a poor outcome.

**Play 5: Add agentic tests to CI/CD**
* Treat agentic systems like traditional software by integrating tests into the CI/CD pipeline.
* These "agentic tests" are a hybrid of code and data, running mock calls to assert outcomes at both the tool level and the end-to-end interaction level.

**Play 6: Build SLMs for real-time monitoring**
* Using large models (LLM-as-judge) for evaluation does not scale due to cost and latency.
* **Small Language Models (SLMs)** between 2-10B parameters offer a wide range of capabilities for detecting harmful or incorrect outcomes.
* Fine-tuning an SLM for **binary outcomes** (e.g., good/bad, hallucination/not) is the most effective way to build an Eval SLM.
* With **<200ms latency**, SLMs open the door to real-time guardrailing and monitoring.

**Play 7: Curate test sets for Agents**
* Develop specific test sets for agents that:
* Assert the format and content of tool outputs (e.g., valid JSON).
* Test intermediate tool inputs.
* Evaluate the quality of tool selection.

**Play 8: Improve metric accuracy w/ human feedback**
* There are several types of human feedback that can be used to improve the performance of evaluation models (Eval Agents):
* **Binary preference signals**: Simple thumbs up/down without verbal feedback.
* **Critique of explanation**: Verbal feedback on the reasoning the eval model used.
* **Critique of value**: Verbal feedback on the final output.
* An interesting finding from their research was that **brief, terse**

**feedback works better than long, detailed feedback.**
* **Question**: How does an SLM give a numerical evaluation (e.g., a score from 0-10) rather than just a binary true/false?
* The speaker explained that the primary focus is on building highly accurate and scalable *binary classifiers* that can operate in real-time. These models are fine-tuned to give a simple pass/fail or "is/is not" judgment. Their performance is measured numerically using metrics like **AUROC scores** on benchmark datasets, but their output in production is typically the binary classification, not a nuanced 0-10 score. The goal is scalable reliability over granular scoring.

---

## RAG, Graphs and Grounding: making AI explainable and trustworthy

*John Willis, author of 13 books*
* Website: aicio.ai
* Background: Infrastructure operations, DevOps Research Assessment (DORA).
* **SWEbench**: A benchmark for evaluating code generation models.
* **Contextual security analysis** beats traditional SAST (Static Application Security Testing).
* Delivery throughput and stability have decreased recently.
* **Ashby's Law** (Law of Requisite Variety): A system must have as much variety in its responses as the environment has in its challenges.
* **OSI Model**: A conceptual framework for understanding network interactions.
* **Core Principles**:
* Version control everything.
* Database management is crucial.
* Implement AI-automated governance.
* **SDLC for AI**: The software development lifecycle needs to be adapted for AI.
* A **prototype is not production**.
* "The easier it is to start, the easier it is to believe you are done."
* **Dimensionality Reduction**: A technique for reducing the number of input variables.
* **Tool**: atlas.nomic.ai for visualizing large datasets.

* **Recommendation**: Provide qualitative descriptions for scores/scales (e.g., "score 1 means X, score 2 means Y...").

---

## Treating AI models as code

*Chester Leung, Opaque & Giuseppe Giordano, Accenture*

This session argued for applying the disciplined practices of software development to the lifecycle of LLM applications to address their inherent lack of security, transparency, and control.

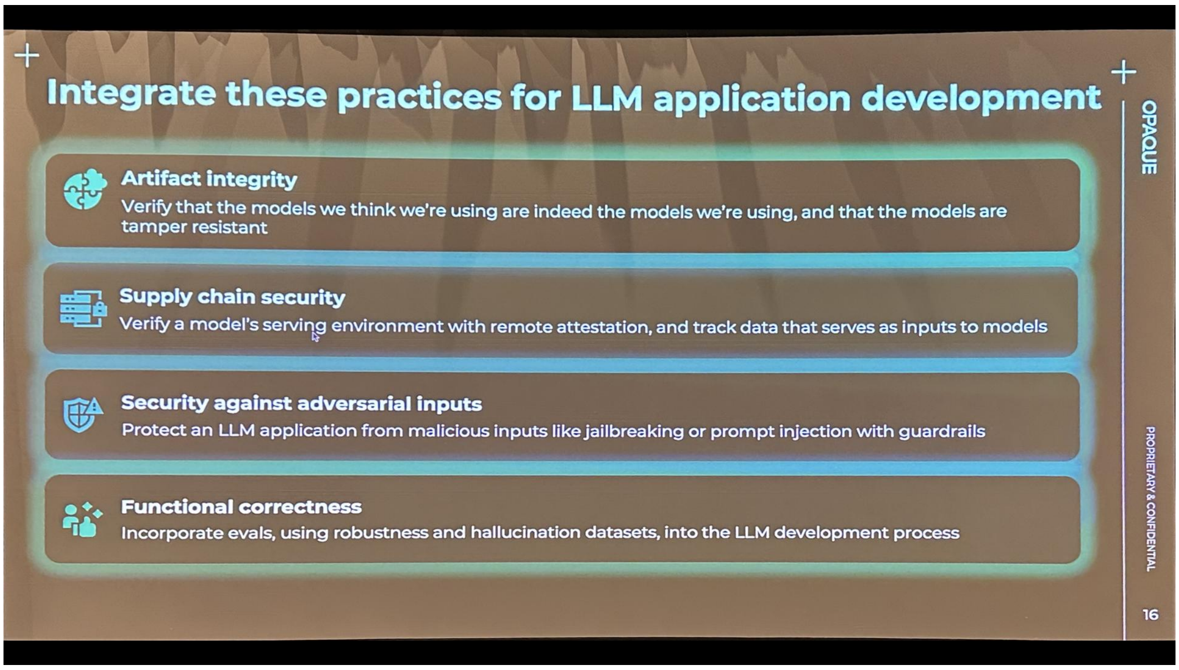
- LLM apps break traditional software rules regarding security, transparency, and control.
- Software development practices give us **reproducibility, accountability, and security**. The goal is to map these to LLM apps.

### Gaps in Current LLM Development

1. **Artifact Integrity**: How do we verify that the model we think we're using is the one we're actually using and that it's tamper-resistant?
2. **Supply Chain Security**: What data and training went into the model? What environment is used to serve the model? We need verifiable answers.
3. **Security Against Adversarial Inputs**: How do we protect the application from prompt injection, jailbreaking, and IP exposure?
4. **Functional Correctness**: How do we define and evaluate correctness for non-deterministic models? This requires incorporating evals using robustness and hallucination datasets into the development process.

### Practices for LLM App Development

The key is to integrate proven security and development practices.



## Integrate these practices for LLM application development

**Artifact integrity**
Verify that the models we think we're using are indeed the models we're using, and that the models are tamper resistant

**Supply chain security**
Verify a model's serving environment with remote attestation, and track data that serves as inputs to models

**Security against adversarial inputs**
Protect an LLM application from malicious inputs like jailbreaking or prompt injection with guardrails

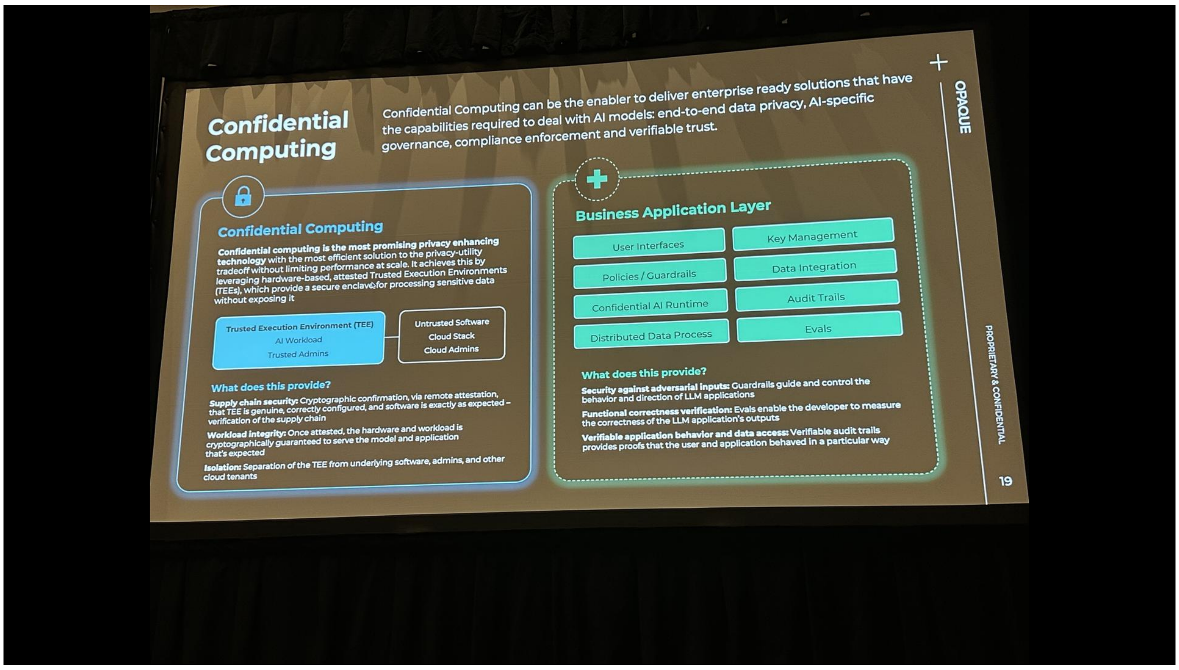
**Functional correctness**
Incorporate evals, using robustness and hallucination datasets, into the LLM development process

OPAQUE

PROPRIETARY & CONFIDENTIAL

16

- **Privacy-Enhancing Technologies (PETs)** **** are critical. The most promising is*** ***Confidential Computing**, which protects data while it's being used in RAM or a processor via a **Trusted Execution Environment (TEE)**.
- A TEE provides a secure enclave for processing sensitive data, isolating the AI workload from the underlying cloud stack and administrators.



## Confidential Computing

Confidential Computing can be the enabler to deliver enterprise ready solutions that have the capabilities required to deal with AI models: end-to-end data privacy, AI-specific governance, compliance enforcement and verifiable trust.

### Confidential Computing

**Confidential computing is the most promising privacy enhancing technology** with the most efficient solution to the privacy-utility tradeoff without limiting performance at scale. It achieves this by leveraging hardware-based, attested Trusted Execution Environments (TEEs), which provide a secure enclave for processing sensitive data without exposing it

**What does this provide?**

**Supply chain security:** Cryptographic confirmation, via remote attestation, that TEE is genuine, correctly configured, and software is exactly as expected – verification of the supply chain

**Workload integrity:** Once attested, the hardware and workload is cryptographically guaranteed to serve the model and application that's expected

**Isolation:** Separation of the TEE from underlying software, admins, and other cloud tenants

### Business Application Layer

**What does this provide?**

**Security against adversarial inputs:** Guardrails guide and control the behavior and direction of LLM applications

**Functional correctness verification:** Evals enable the developer to measure the correctness of the LLM application's outputs

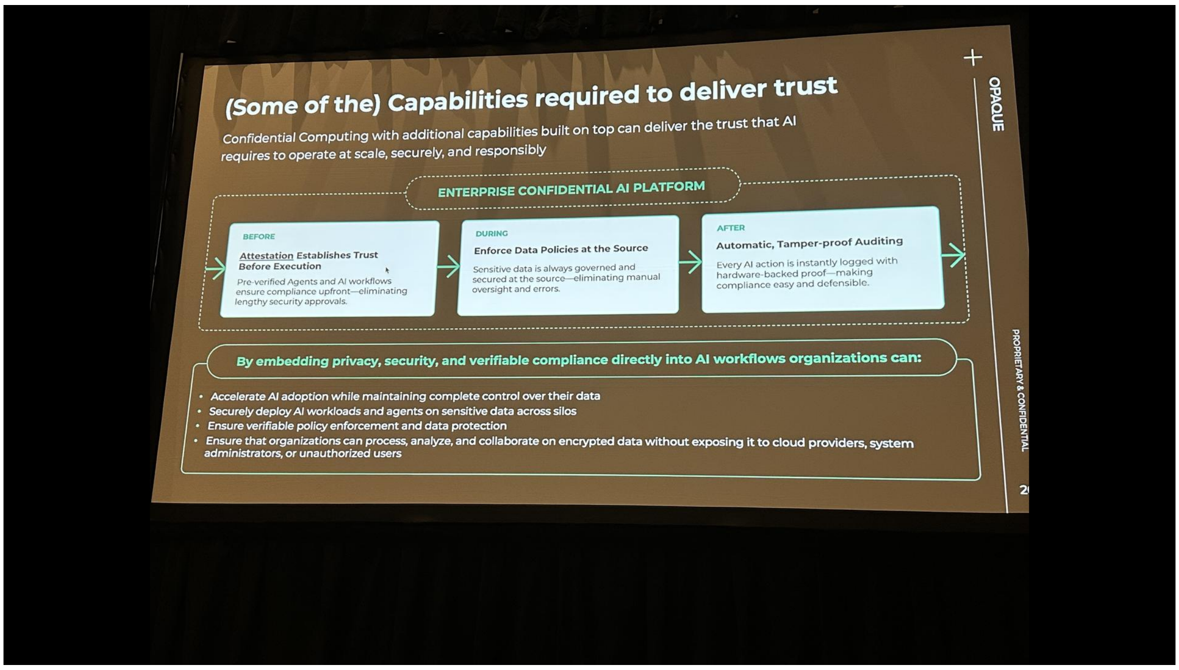
**Verifiable application behavior and data access:** Verifiable audit trails provides proofs that the user and application behaved in a particular way

OPAQUE

PROPRIETARY & CONFIDENTIAL

19

- This approach is encapsulated in an **Enterprise Confidential AI Platform** that establishes trust *before* execution, enforces policies *during*, and provides tamper-proof audits *after*.



## (Some of the) Capabilities required to deliver trust

Confidential Computing with additional capabilities built on top can deliver the trust that AI requires to operate at scale, securely, and responsibly

ENTERPRISE CONFIDENTIAL AI PLATFORM

BEFORE
**Attestation Establishes Trust Before Execution**
Pre-verified Agents and AI workflows ensure compliance upfront—eliminating lengthy security approvals.

DURING
**Enforce Data Policies at the Source**
Sensitive data is always governed and secured at the source—eliminating manual oversight and errors.

AFTER
**Automatic, Tamper-proof Auditing**
Every AI action is instantly logged with hardware-backed proof—making compliance easy and defensible.

**By embedding privacy, security, and verifiable compliance directly into AI workflows organizations can:**

- Accelerate AI adoption while maintaining complete control over their data
- Securely deploy AI workloads and agents on sensitive data across silos
- Ensure verifiable policy enforcement and data protection
- Ensure that organizations can process, analyze, and collaborate on encrypted data without exposing it to cloud providers, system administrators, or unauthorized users

OPAQUE

PROPRIETARY & CONFIDENTIAL

2

- **Demo**: The speakers showed a chatbot designed to answer questions about sales commission rates.
  - The entire pipeline (data ingestion from a sensitive database, LLM)

runs inside a confidential computing environment.
    - Before the chatbot sends a query, it performs **remote attestation** to cryptographically verify that the pipeline is running in genuine hardware and the software is exactly what is expected.
    - When a user asked a malicious question ("how to make a bomb"), the guardrails within the secure pipeline intercepted and blocked the request.
- **Scalability**: This architecture can run on powerful hardware like H100 GPUs.
- **Overhead**: Performance overhead is minimal during runtime and is primarily incurred during the initial setup and attestation of the confidential VMs.

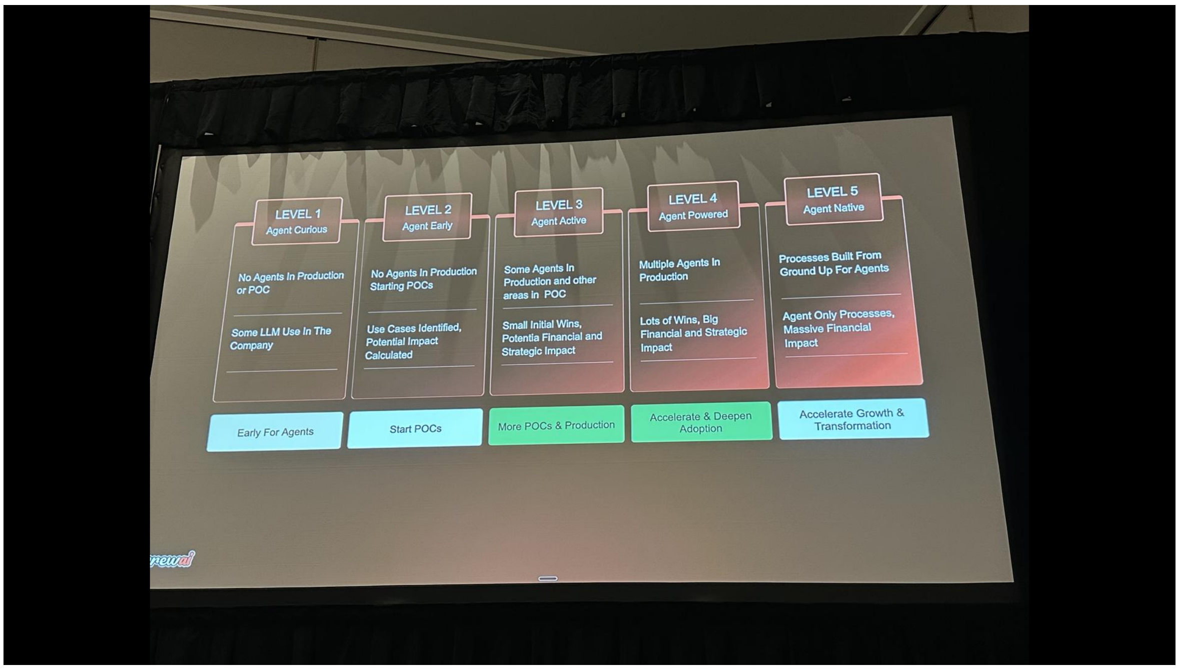
---

## AI Agents at scale: insights and trends from millions of agent interactions

*Joao Moura from crew AI*

This session provided insights from `crew.ai`, a platform that builds and manages ecosystems of AI agents, handling over 60 million agent interactions a month.

- The ultimate vision is an **Agent Native** enterprise, where processes are built from the ground up for agents. The platform maturity model progresses from "Agent Curious" to "Agent Native".



| LEVEL 1 Agent Curious | LEVEL 2 Agent Early | LEVEL 3 Agent Active | LEVEL 4 Agent Powered | LEVEL 5 Agent Native |
|---|---|---|---|---|
| No Agents In Production or POC | No Agents In Production Starting POCs | Some Agents In Production and other areas in POC | Multiple Agents In Production | Processes Built From Ground Up For Agents |
| Some LLM Use In The Company | Use Cases Identified, Potential Impact Calculated | Small Initial Wins, Potentia Financial and Strategic Impact | Lots of Wins, Big Financial and Strategic Impact | Agent Only Processes, Massive Financial Impact |
| Early For Agents | Start POCs | More POCs & Production | Accelerate & Deepen Adoption | Accelerate Growth & Transformation |

* The stack for an agent-native enterprise includes: Data Management -> LLMs -> Agent Orchestration -> Agent Memory -> Authentication & Scoping -> Enterprise Connectors -> Agentic Apps.
* **Challenges**: The biggest hurdles are **interoperability** between different agent systems and ensuring **observability**.
* **Task Dimensions**: Agent tasks can be categorized by two dimensions:
* **Complexity (C)**
***: How many steps or variables are involved?***
** * *Precision (P)**: How exact does the final output need to be?
* **High C, High P**: Filling out a 60-page IRS tax form. This requires high precision.
* **High C, Low P**: Teaching a concept. The process is complex, but there isn't one single "right" answer.
* **Frameworks**:
* **Crew**: Better for tasks requiring more *agency* (autonomy, complex planning).
* **Flows**: Better for tasks requiring finer *precision* and control.
* **Favorite Use Case**: **Back-office automation**.

> *"This is something that is not really sexy but you know what there is a lot of savings in there. Like you automate some of this*

> *thing and they are running like many many times a day. That can be quite quite helpful."*

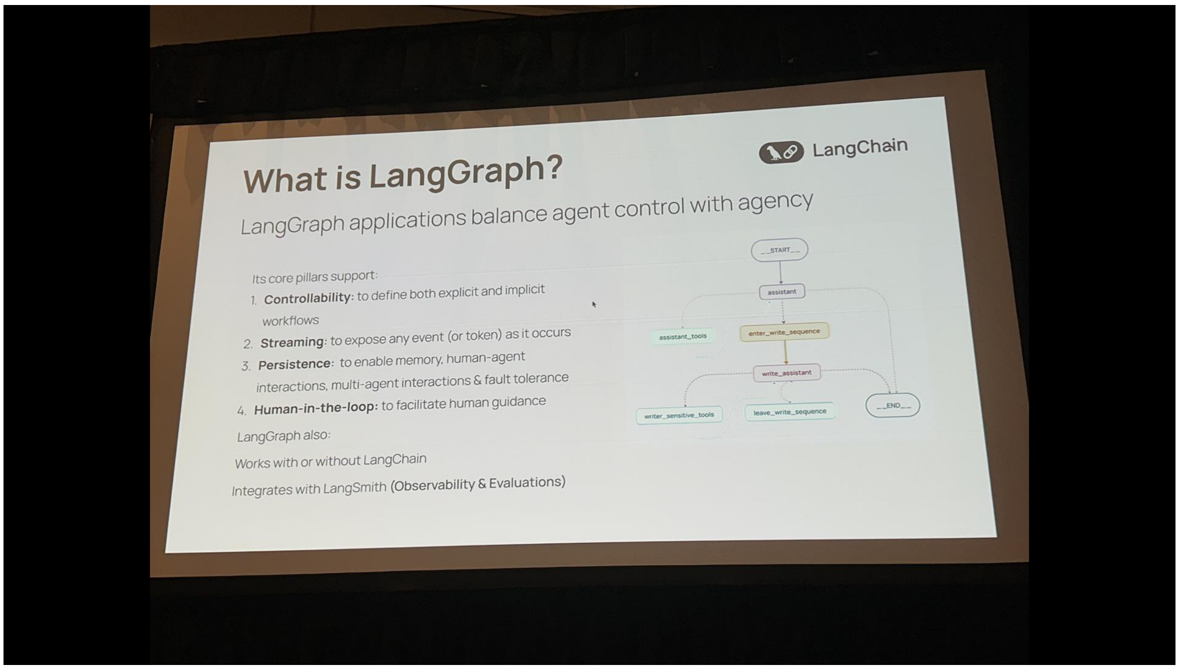
---

## Building Reliable Agents with LangGraph

*Jake Broekhuizen from LangChain*

This session introduced LangGraph as a framework for building controllable and reliable agentic systems.

- **LangChain Ecosystem**:
  - **Open Source**: LangChain (integration framework), LangGraph (agent orchestration).
  - **Proprietary**: LangSmith (observability), LangGraph Platform (enterprise hub).
- **Evolution of AI System Design**:
  1. **Paradigm 1: Chain**: A predetermined, deterministic control flow (e.g., simple RAG). Lacks flexibility.
  2. **Paradigm 2: ReAct Agent**: An LLM running in a loop with tools. This introduced non-determinism but can suffer from poor reliability and get stuck in loops.
  3. **Paradigm 3: LangGraph**: Combines the reliability of deterministic graphs with the flexibility of LLM-based agents.
- **What is LangGraph?**
  - A controllable agent orchestration framework for handling complex tasks reliably. Its core pillars support building production-ready applications.



## What is LangGraph?

LangChain

LangGraph applications balance agent control with agency

Its core pillars support:

1. **Controllability:** to define both explicit and implicit workflows
2. **Streaming:** to expose any event (or token) as it occurs
3. **Persistence:** to enable memory, human-agent interactions, multi-agent interactions & fault tolerance
4. **Human-in-the-loop:** to facilitate human guidance

LangGraph also:

Works with or without LangChain

Integrates with LangSmith **(Observability & Evaluations)**

- **Core Pillars of LangGraph**:
    1. **Controllability**: Express any control flow you need, including fixed paths, conditional edges, and cycles (for self-correction/ reflection)
       . Nodes in the graph update a shared state (like chat history) and can contain arbitrary logic, including calls to other sub-graphs.
    2. **Streaming**: Provides a real-time, interactive experience for end-users by streaming intermediate state updates as they happen.
    3. **Persistence**: Enables both short-term memory (via a checkpointer) and long-term memory (via a persistent store) for stateful applications.
    4. **Human-in-the-loop**: Allows for human intervention at any point in the graph for approval, editing, or providing additional input.
- **Demo**: The speaker used LangGraph Studio (`smith.langchain.com`) to build a multi-agent customer support bot.

Image not found: cropped_slides/IMG_1429.jpeg

* The graph demonstrated a supervisor agent that could route tasks to

other specialized agents, wait for human input, and maintain both short-term (conversation history)
and long-term (user preferences) memory.
* **My question about long-term memory (update, history tracking, conflict resolution)**:

> *The speaker clarified that LangGraph provides a low-level solution and the implementation is up to the user. "our offering is very, uh, low-level in the sense that the long-term memory store... is a key value store... the actual, uh, memory logic of merging, maintaining, updating... is kind of up to whoever is building the application".*

---

## Building LLMs with Nvidia NeMo and Securing them with confidential computing

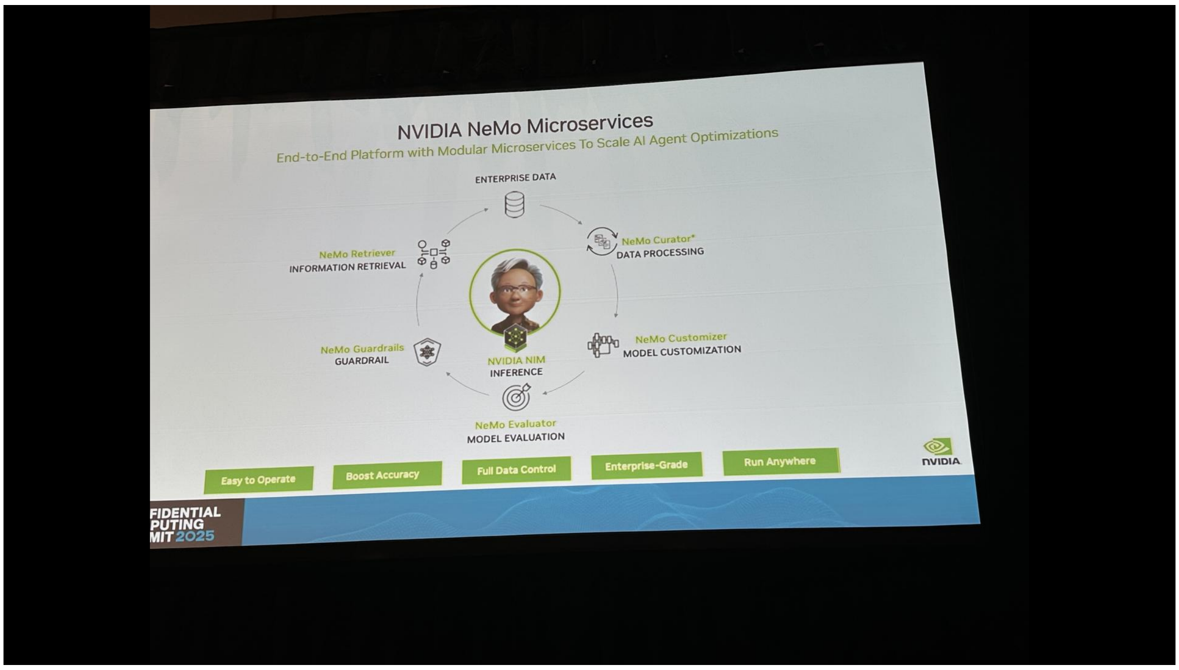
*Chester Chen, Shashank Verma, Sylendran Arunagiri from Nvidia*

This presentation covered two main topics: using Nvidia's NeMo platform to optimize LLMs and using confidential computing to secure them.

### Part 1: AI Agents at Scale with Nvidia NeMo

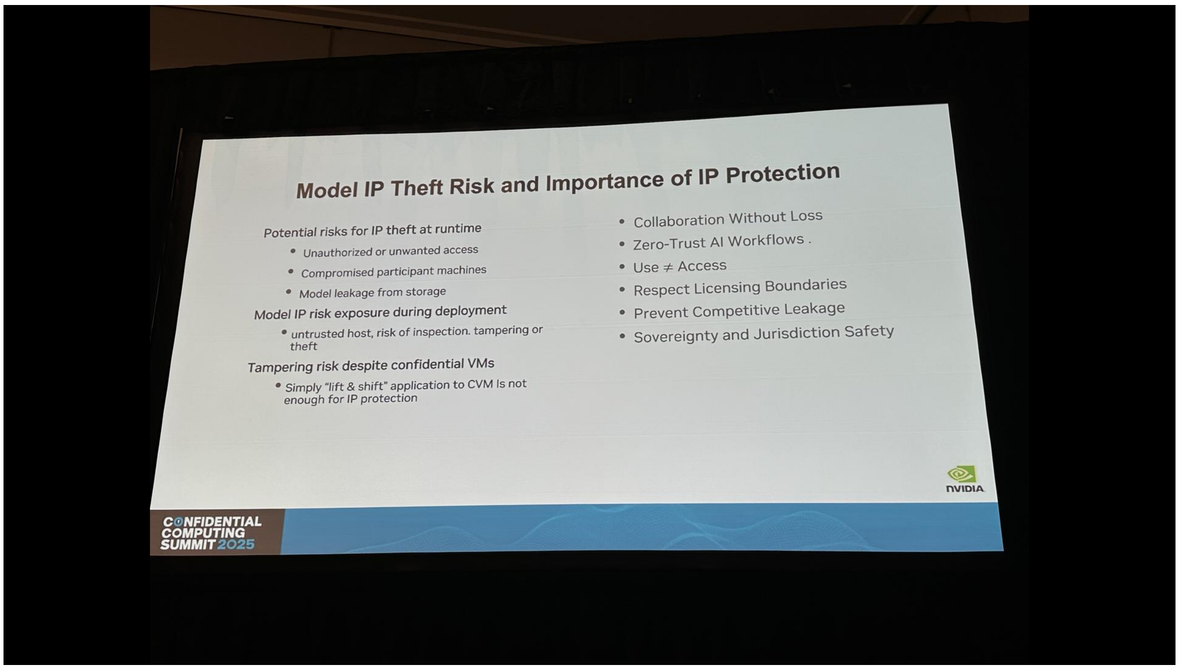
Operating AI agents at scale is challenging due to rapidly aging data, evolving customer needs, rising inference costs, and high accuracy targets.

- **Data Flywheel**: The solution is a "data flywheel" that automates the optimization of AI agents at scale.
- **Nvidia NeMo Microservices**: This is an end-to-end platform with modular microservices that power the data flywheel. It gives enterprises full control over their data and optimization pipelines. The platform includes services for data processing, model customization, evaluation, guardrailing, and retrieval.



**Part 2: How confidential computing secures federated learning and inference**

- **Federated Learning (FLARE)*
  *: Nvidia FLARE is a framework for decentralized model training.
  > "Don't move the data at all. So you do the local training and then you only share the model weights."
  - This is used widely in hospitals and financial services where data privacy is paramount.
- **The Problem**: Even if you only share model weights, how do you protect your model IP from being copied or stolen? The concept is **use ≠ access**.
- **Risks**:
  - **Runtime Risk**: Unauthorized access to a compromised machine or model leakage from storage.
  - **Deployment Risk**: An untrusted host can inspect, tamper with, or steal the model from a checkpoint.



## Model IP Theft Risk and Importance of IP Protection

Potential risks for IP theft at runtime
- Unauthorized or unwanted access
- Compromised participant machines
- Model leakage from storage

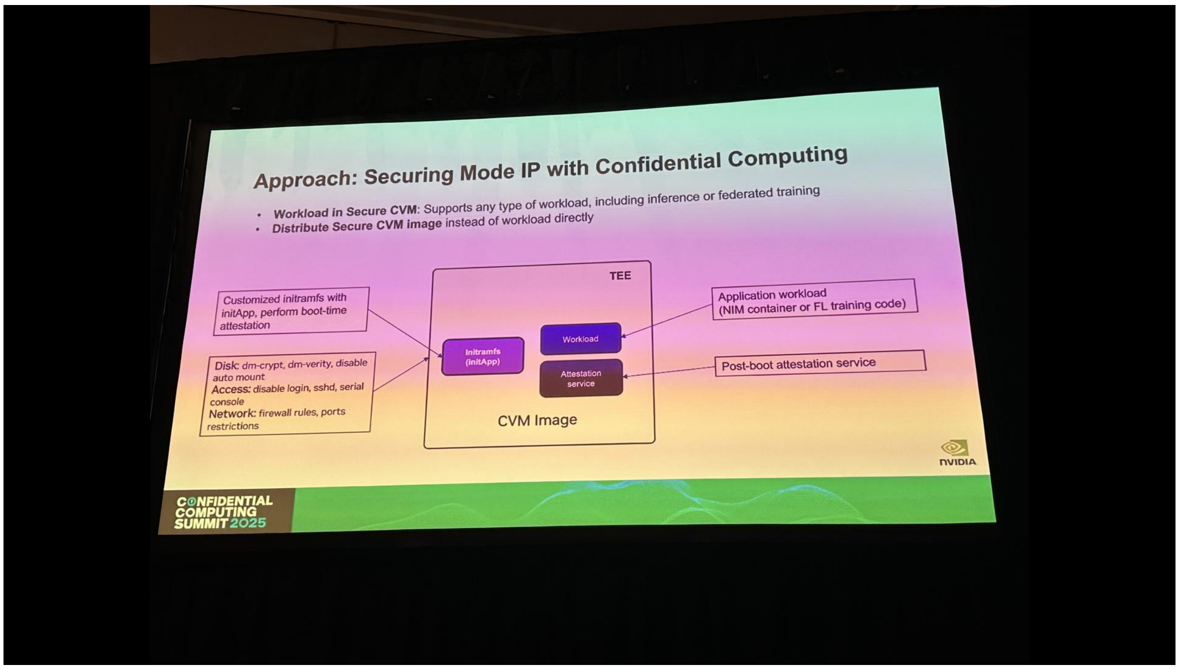
Model IP risk exposure during deployment
- untrusted host, risk of inspection. tampering or theft

Tampering risk despite confidential VMs
- Simply "lift & shift" application to CVM is not enough for IP protection

- Collaboration Without Loss
- Zero-Trust AI Workflows .
- Use ≠ Access
- Respect Licensing Boundaries
- Prevent Competitive Leakage
- Sovereignty and Jurisdiction Safety

- **Approach: Securing Model IP with Confidential Computing**
    - Instead of deploying the model code directly, it is packaged into a **secure, locked-down Confidential Virtual Machine (CVM)**

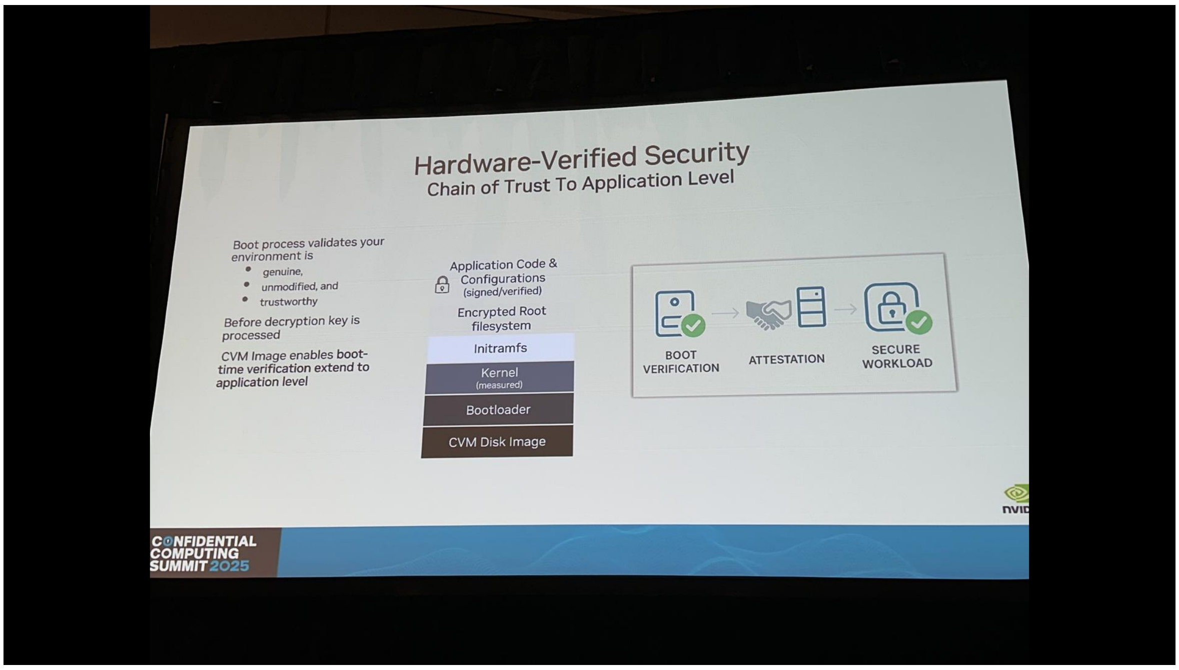
      **image**.
    - This CVM image is distributed to participants. The CVM is designed to be a black box with encrypted disk, restricted network access, and no login/SSH capabilities.


Approach: Securing Mode IP with Confidential Computing

- **Workload in Secure CVM**: Supports any type of workload, including inference or federated training
- **Distribute Secure CVM image** instead of workload directly

TEE
Customized initramfs with initApp, perform boot-time attestation
Disk: dm-crypt, dm-verity, disable auto mount
Access: disable login, sshd, serial console
Network: firewall rules, ports restrictions
Initramfs (initApp)
Workload
Attestation service
CVM Image
Application workload (NIM container or FL training code)
Post-boot attestation service
CONFIDENTIAL COMPUTING SUMMIT 2025

- **Hardware-Verified Security**: Trust is established via a "chain of trust" that extends from the hardware to the application level.
  - The boot process validates that the environment is genuine and unmodified before any decryption keys are processed.
  - **Attestation** occurs at boot time and periodically post-boot to ensure the environment remains secure.


Hardware-Verified Security
Chain of Trust To Application Level

Boot process validates your environment is
- genuine,
- unmodified, and
- trustworthy

Before decryption key is processed

CVM Image enables boot-time verification extend to application level

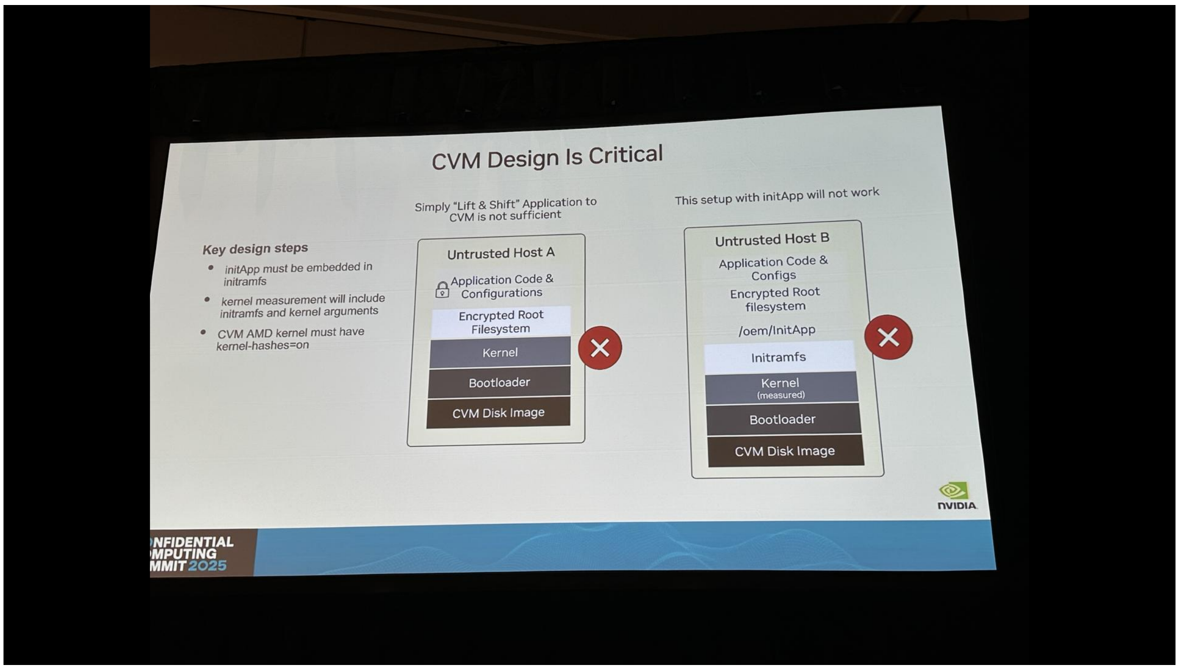
Application Code & Configurations (signed/verified)
Encrypted Root filesystem
Initramfs
Kernel (measured)
Bootloader
CVM Disk Image
BOOT VERIFICATION
ATTESTATION
SECURE WORKLOAD
CONFIDENTIAL COMPUTING SUMMIT 2025

- **CVM Design is Critical**: Simply "lifting & shifting" an application into a CVM is not sufficient.

> The speaker warned, "if you state the competitive VM file to somebody else... you can simply, you know, modify your, you know, boot up code and then... disable all these features."

- The security mechanisms must be embedded deep into the boot process (initramfs)

  to prevent tampering.



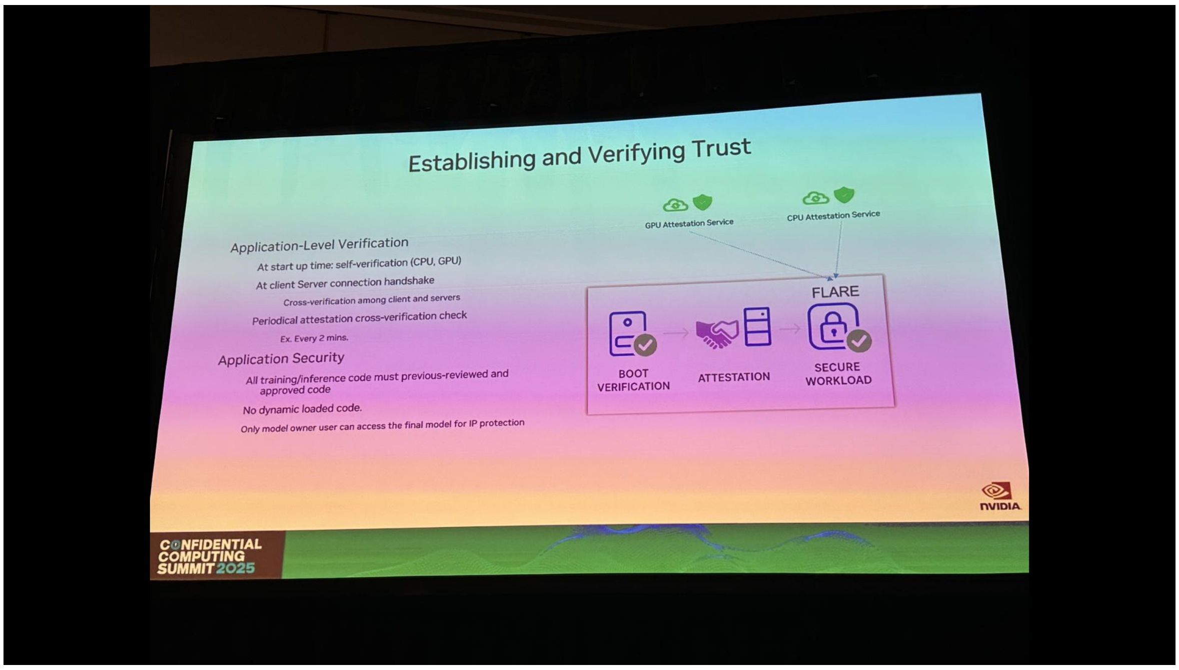
## CVM Design Is Critical

**Key design steps**

- initApp must be embedded in initramfs
- kernel measurement will include initramfs and kernel arguments
- CVM AMD kernel must have kernel-hashes=on

Simply "Lift & Shift" Application to CVM is not sufficient
Untrusted Host A
Application Code & Configurations
Encrypted Root Filesystem
Kernel
Bootloader
CVM Disk Image
This setup with initApp will not work
Untrusted Host B
Application Code & Configs
Encrypted Root filesystem
/oem/InitApp
Initramfs
Kernel (measured)
Bootloader
CVM Disk Image
NVIDIA
NFIDENTIAL MPUTING MMIT 2025

- **Establishing and Verifying Trust**: Trust is verified at multiple stages: at startup, during client-server connections, and periodically through cross-verification between participants and attestation services.



## Establishing and Verifying Trust

Application-Level Verification

- At start up time: self-verification (CPU, GPU)
- At client Server connection handshake
  - Cross-verification among client and servers
- Periodical attestation cross-verification check
  - Ex. Every 2 mins.

Application Security

- All training/inference code must previous-reviewed and approved code
- No dynamic loaded code.
- Only model owner user can access the final model for IP protection