

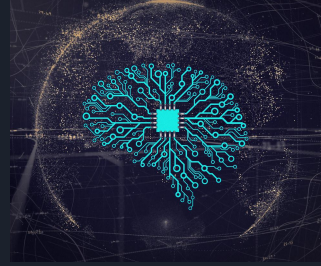
Cloud Native AI Introduction, Challenges, and Path Forward



Introduction

- What is Cloud Native?
- What is AI?
- What is Cloud Native AI?

What is Cloud Native ?

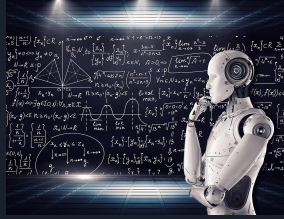


Cloud Native is a modern way of building and running software that fully utilizes cloud environments like AWS, Azure, or private and hybrid clouds. This approach embraces flexibility and scalability, allowing applications to adapt to changing needs. Key technologies include

- containers - which are lightweight software packages enabling easy application movement between different cloud environments.
- Service meshes - manage communication between different parts of an application.
- Microservices - break down applications into smaller, independent parts for easier updates.
-

In short, cloud-native is about building flexible, efficient software that works well in cloud environments!

What is AI?



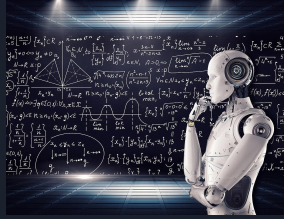
AI, or Artificial Intelligence, refers to the capability of machines to simulate human intelligence, enabling them to perform tasks that typically require human-like understanding, reasoning, learning, and problem-solving.

There are two primary classifications of AI: discriminative and generative.

Discriminative AI:

- Focuses on learning to make decisions or classifications based on existing data.
- Creates a "model" based on known outcomes to predict new data.
 - Examples include classifying emails as spam or not, identifying images of cats versus dogs.
- Used for tasks where the desired output is clear and known (like supervised learning).
- Excels in tasks like sequence prediction, such as predicting the next word in a sentence.

What is AI?



Generative AI:

- Learns underlying patterns or representations in data.
- Uses these patterns to generate new, original data.
- Can create stories, music, or visual art based on learned structures.
- Ideal for tasks with unclear or undefined outcomes.
- Represents AI's ability to be creative and produce unique content.

In summary, discriminative AI focuses on making decisions based on known information, while generative AI generates new content based on learned patterns, demonstrating AI's ability to be innovative and artistic.

What is Cloud Native AI?



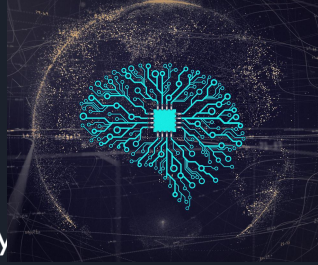
Cloud Native Artificial Intelligence (CNAI) is an emerging concept that integrates the principles of Cloud Native computing with Artificial Intelligence (AI) applications. It aims to facilitate the development and deployment of AI solutions in a scalable and efficient manner by leveraging Cloud Native practices.

CNAI solutions aim to address the challenges encountered by AI application scientists, developers, and deployers throughout the lifecycle of *Developing, Deploying, Running, Scaling*, and monitoring AI workloads on cloud infrastructure. Let see:

- **Developing:** CNAI provides tools and frameworks that facilitate the development of AI applications using Cloud Native principles. This includes leveraging containerization and declarative APIs to streamline the development process and ensure portability and consistency across different environments.



What is Cloud Native AI?



- **Deploying:** CNAI simplifies the deployment of AI models and applications by automating the deployment process and handling dependencies effectively. Orchestrators like Kubernetes enable seamless deployment and management of AI workloads across distributed clusters.
- **Running:** CNAI ensures that AI workloads run efficiently and reliably on cloud infrastructure by optimizing resource allocation and utilizing auto-scaling capabilities to accommodate changing demands.
- **Scaling:** One of the key features of CNAI is scalability. It enables AI workloads to scale up or down dynamically based on workload fluctuations, ensuring optimal performance and cost-efficiency.
- **Monitoring:** CNAI solutions incorporate robust monitoring and logging capabilities to track the performance, health, and usage of AI applications. This allows AI practitioners to proactively identify issues, optimize resource utilization, and ensure the overall stability of the system.

It accelerates AI application performance and reduces costs.




Cloud Native AI: Bridging Cloud Infrastructure and Artificial Intelligence

Cloud Native AI – combines the flexibility of cloud infrastructure with the power of artificial intelligence. In the cloud, businesses and developers can prototype quickly, scale services, and share resources efficiently.

AI – faces challenges like accessing storage, networking, and computing resources.

By integrating AI into cloud-native systems, we enhance productivity for operators and end-users.

An example is the open-source project K8sGPT, which assists Kubernetes (K8s) operators using AI capabilities. This symbiosis of cloud-native and AI opens up new opportunities, making complex systems more accessible even to less technical users.




Exploring AI, Machine Learning, and Data Science Applications: Predictive vs. Generative Approaches

In this section, we'll break down the concepts of AI, machine learning (ML), and data science into simpler terms and discuss how they are applied in two main categories: Predictive AI and Generative AI.

Understanding AI, ML, and Data Science

- AI (Artificial Intelligence): Creating systems that can perform tasks similar to humans.
- ML (Machine Learning): Using algorithms to learn from data and make decisions without explicit programming.
- Data Science: Applying statistics, mathematics, and computer science to analyze and interpret data, often using ML techniques.



Exploring AI, Machine Learning, and Data Science Applications: Predictive vs. Generative Approaches

Predictive AI

- Focuses on predicting and analyzing existing patterns or outcomes.
- Examples: Classification (identifying categories), clustering (grouping similar data points), regression (predicting numeric values), object detection (identifying objects in images).

Generative AI

- Aimed at creating new and original content.
- Examples: Large Language Models (LLMs) that generate text, RAG17 which requestData

Predictive vs. Generative Approaches

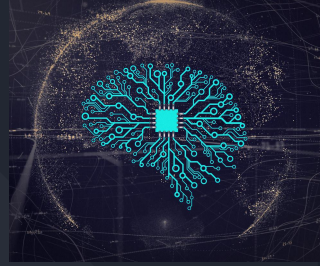
Challenges/Need	Generative AI	Predictive AI
Computational Power	Extremely high. Requires specialized hardware.	Moderate to high. General-purpose hardware can suffice.
Data Volume and Diversity	Massive, diverse datasets for training.	Specific historical data for prediction.
Model Training and Fine-tuning	Complex, iterative training with specialized compute	Moderate training.
Scalability and Elasticity	Highly scalable and elastic infrastructure (variable and intensive computational demands)	Scalability is necessary but lower elasticity demands. Batch processing or event-driven tasks.

Predictive vs. Generative Approaches

Challenges/Need	Generative AI	Predictive AI
Storage and Throughput	High-performance storage with excellent throughput. Diverse data types. Requires high throughput and lowlatency access to data.	Efficient storage with moderate throughput. It focuses more on data analysis and less on data generation; data is mostly structured.
Networking	High bandwidth and low latency for data transfer and model synchronization (e.g., during distributed training).	Consistent and reliable connectivity for data access.

Understanding Core Topics

- **Orchestration**
- **Kubernetes**
- **Convolutional Neural Networks (CNNs)**
- **Transformers**





Orchestration in context of CNAI

In the context of Cloud Native Artificial Intelligence (CNAI), orchestration refers to the process of managing and coordinating AI workloads, resources, and services across distributed and dynamic cloud environments using automated tools and platforms. Orchestration plays a crucial role in optimizing the deployment, scaling, and operation of AI applications within Cloud Native architectures.

Key aspect of orchestration in CNAI include:

Resource Allocation and Scaling:

- Orchestration tool like Kubernetes manage the allocation of computational resources (such as CPU and memory) to AI workloads based on demand.
- Auto-scaling features dynamically adjust resource allocation to accommodate fluctuations in AI workload requirements, ensuring optimal performance and cost efficiency.



Orchestration tool – Kubernetes?

Kubernetes (often abbreviated as “K8s”) is a system that manages containers—those lightweight virtual machines that hold your applications.

It is a software tool that helps manage and organize applications that are packaged in containers. It automates tasks like starting, stopping, and scaling these containers across multiple computers, making it easier to run applications reliably in large-scale environments.

Kubernetes handles many complex tasks behind the scenes, like ensuring applications stay running even if part of the system fails, and it helps balance the workload across different parts of the system to keep everything running smoothly.

It's like having a smart manager for your applications in a big computer network, making sure they work efficiently and without interruption.



Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a type of advanced technology for processing images and other types of data.

They were initially created back in the 1980s but didn't gain widespread use until around the early 2000s. In recent times, they've become very popular because they're really good at learning from large collections of images and can do many different things with images, like recognizing objects, figuring out what's in a picture, and drawing lines around different objects.



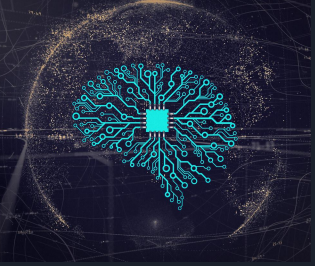
Transformers

Transformers are a type of technology created by researchers from the University of Toronto and Google in 2017.

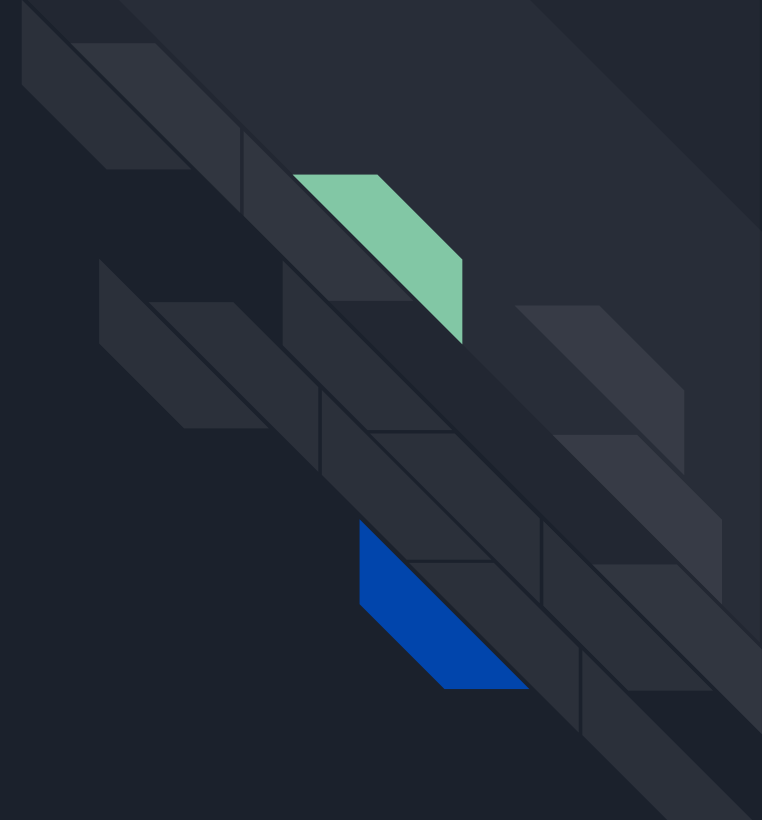
They work differently from older methods and use a clever system called "scaled dot-product attention," which helps them remember things they've learned before.

Transformers are particularly good at understanding and working with human language.

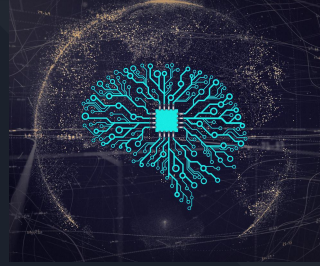
They're used for tasks like answering questions, making summaries of text, and translating languages. Because they work so well with language, they're a crucial part of many advanced language models, including GPT, which is the model that runs services like ChatGPT that you're using now.



CNAI CHALLENGES



CNAI Challenges



- Data Preparation
- Model Training
- CI/CD, Model Registry
- Model Serving
- Education in Kids



CNAI Challenges

Cloud Native Artificial Intelligence (CNAI) faces different challenges depending on who is using it. While Cloud Native platforms are flexible and scalable, they still need to address the scale and speed requirements of AI applications, which can reveal gaps in Cloud Native technologies but also offer chances for improvement.

These challenges are explored in the context of a complete machine learning (ML) process, also known as MLOps.

In simpler terms, when dealing with AI tasks, challenges arise related to managing time & space, parallel processing and keeping different parts of a task coordinated. These challenges highlight gaps in how easy it is to use AI systems effectively.

The typical ML pipeline is comprised of:

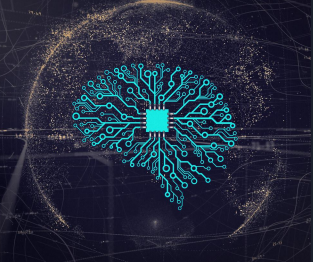
- Data Preparation
- Model Training
- CI/CD, Model Registry
- Model Serving
- Observability



Data Preparation

Data preparation is a crucial first step in AI and machine learning (ML) pipelines, but it comes with various challenges:

- **Data Size:** The demand for data to train AI models is growing rapidly, outpacing hardware capabilities. Cloud Native computing and efficient data handling are essential to manage large datasets effectively.
- **Data Synchronization:** Data often comes from different sources and needs to be synchronized across development and production environments. This complexity increases with distributed computing, requiring consistent data partitioning and synchronization.
- **Data Governance:** Ensuring privacy, security, ownership, and lineage of data is vital for responsible AI development:
 - **Privacy and Security:** Complying with data privacy regulations (like GDPR and CCPA) and implementing robust security measures (encryption, access controls) are critical.
 - **Ownership and Lineage:** Defining data ownership and tracking data lineage throughout the AI lifecycle ensures transparency and accountability.
 - **Mitigating Bias:** Monitoring and addressing biases in data and algorithms is crucial for ethical AI. Using diverse datasets and fairness metrics helps ensure fair outcomes.



Model Training Challenges

- Rising Processing Demands
- Cost Efficiency
- Sustainability
- Scalability
- Orchestration/Scheduling
- Custom Dependencies



Model Training

Model training in AI faces several challenges and complexities as Rising Processing Demands, Cost Efficiency, Sustainability, Scalability, Orchestration/Scheduling, Custom Dependencies.

- **Rising Processing Demands:** Increasing data volumes require distributed processing and accelerators like GPUs, TPUs, and FPGAs.

Virtualization and efficient management of diverse compute resources are essential but complex, requiring CN scheduler enhancements.

- **Cost Efficiency:** Cloud Native environments offer elasticity, allowing dynamic resource provisioning based on workload demands.

Fractionalizing GPUs and utilizing auto scaling frameworks like KServe help optimize resource usage and reduce costs.



Model Training

- **Sustainability:** Auto-scaling serving frameworks and techniques like model compression reduce carbon footprint during model deployment.

Tools like mlco2 and codecarbon aid in predicting and managing the environmental impact of ML models.

- **Scalability:** AI/ML workflows involve complex coordination due to large data volumes and multiple training rounds.

Orchestration tools like Kubernetes simplify scaling by decomposing workflows into modular components.

- **Orchestration/Scheduling:** Cloud Native tools streamline AI workload orchestration and scheduling using containerization and microservices.

Optimizing GPU sharing and scheduling is crucial for efficient AI development, with tools like Yunikorn and Volcano evolving to address advanced scheduling needs.



Model Training

- **Custom Dependencies:** AI applications often require specific frameworks and GPU libraries, posing challenges for compatibility and reproducibility.

Versioned dependencies and reproducible builds are necessary to avoid runtime issues and ensure performance consistency.

In summary, model training in AI involves managing increasing data demands, optimizing resource utilization for cost efficiency and sustainability, and addressing orchestration and scheduling complexities. Cloud Native technologies offer solutions to streamline these processes but require careful management and collaboration between AI and Cloud Native engineering teams for effective integration and scalability.

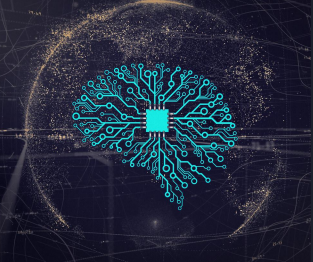


CI/CD, Model Registry

CI/CD (Continuous Integration/Continuous Deployment) and Model Registry are essential components in machine learning (ML) pipelines and software development processes:

- **CI/CD (Continuous Integration/Continuous Deployment):**
 - **Continuous Integration (CI):** The practice of automatically integrating code changes from multiple contributors into a shared repository. CI involves running automated tests to detect integration errors early in the development cycle.
 - **Continuous Deployment (CD):** The automated process of deploying code changes to production environments after passing the CI tests. CD ensures that changes are delivered to users quickly and reliably.
- **Model Registry:** A Model Registry is a centralized repository or catalog where trained ML models are stored, versioned, and managed.
 - Key functionalities of a Model Registry include version control, model metadata storage, model lineage tracking, and collaboration support.
 - Model Registry facilitates the deployment and management of ML models in production environments by providing a reliable source of truth for model artifacts and configurations.

In summary, CI/CD streamlines software development workflows, ensuring that changes are tested and deployed automatically. The Model Registry enables efficient management and deployment of ML models by organizing and versioning model artifacts.



Model Serving Challenges

- Load Variability and Latency
- Microservice Architecture Challenges
- Unified ML Infrastructure
- Model Placement and Resource Allocation
- User Experience and Ease of Use
- Cross-Cutting Concerns
 - Reference Implementation
 - Source Right-Sizing
 - Right-sizing Resource Provisioning
 - Cost Control
 - Observability
 - Disaster Recovery and Business Continuity
 - Security and Compliance Audits
 - Sustainability



Model Serving

Model serving is about deploying and using AI models in a way that's different from how we train or process data with these models.

- **Load Variability and Latency**

The main challenges with model serving are dealing with varying levels of demand (load variability) and meeting specific time limits (latency requirements). There's also a need to ensure that the service stays available even if there are problems, and to share resources efficiently to save money.

AI models can be very different depending on what they're used for. For example, traditional machine learning (ML), deep learning (DL), generative AI (GAI), large language models (LLMs), and new multi-modal approaches (like converting text to video) all have unique characteristics. Because of this diversity, they require different kinds of support from the infrastructure that runs them.

Before large language models (LLMs) became popular, model serving usually only needed one graphics processing unit (GPU). Some people even used regular processors (CPUs) for less urgent tasks. But with LLMs, the biggest challenge is often not having enough memory for the model to work quickly, rather than not having enough processing power.

In simple terms, model serving is about using AI models effectively and efficiently, considering the specific demands and characteristics of each type of model.



Model Serving

- **Microservice Architecture and Developer Experience**

Cloud Native (CN) often uses microservice architecture, which can be challenging for AI applications. In this setup, each stage of the machine learning (ML) process becomes a separate microservice. This can make it hard to manage and synchronize outputs and transitions between these stages. Even experimenting with these solutions on a laptop may require creating many separate parts (called Pods), which adds complexity and limits flexibility for different ML tasks.

Using microservices for ML can also create a fragmented experience for AI practitioners. Instead of just focusing on writing ML Python scripts, they might need to deal with building container images, writing custom resource files, using workflow orchestrators, and more. This extra complexity can be difficult to learn and manage, especially if it's outside their usual expertise.

Additionally, integrating different stages of ML models across various systems increases costs significantly. For example, the Samsara engineering blog mentioned challenges with ML production pipelines spread across multiple microservices, each handling data processing, model inference, and business logic separately. This setup made it complex to synchronize resources and slowed down development and model releases.



Model Serving

To address these issues, a unified ML infrastructure is needed, using a general-purpose distributed computation engine like Ray.

Ray can work alongside existing Cloud Native tools, focusing on computation tasks while letting the Cloud Native ecosystem handle deployment and delivery. The Ray/KubeRay community has collaborated extensively with other Cloud Native communities like Kubeflow, Kueue, Google GKE, and OpenShift to improve ML pipeline performance and reduce costs by sharing resources and streamlining processes.

- **Unified ML Infrastructure:** A unified ML infrastructure refers to a cohesive system or platform that integrates various components of machine learning workflows, such as data processing, model training, and model serving, into a unified framework. This integration simplifies management, enhances performance, and reduces costs by leveraging shared resources and standardized tools across the entire machine learning lifecycle.



Model Serving

A unified infrastructure using distributed computation engines like Ray can enhance ML pipeline performance and reduce costs through resource sharing.

- **Model Placement and Resource Allocation:** Users seek to deploy multiple models with varied requirements (e.g., latency, resiliency) while optimizing resource usage. Efficient GPU fractionalization and dynamic resource allocation (DRA) are crucial for optimal performance and cost control.
- **User Experience and Ease of Use:** Improving user experience and ease of use involves making it easier to deploy AI applications on Kubernetes using simple software development kits (SDKs) and integrating with tools like JupyterLab. This enhances the experience for developers. Additionally, addressing challenges related to multi-tenancy, observability, and debugging requires better tools and support, especially for users who are not system administrators.



Model Serving

- Model Placement

When deploying AI models for making predictions (inference), users often want to run multiple models together in the same cluster, even if these models are not related to each other. They do this to save costs and ensure each model runs independently without affecting others.

To make sure their models are reliable and available even if part of the system fails, users also want to have duplicate copies (replicas) of their models spread across different failure zones.

Kubernetes, a popular system for managing containerized applications, offers ways to control where workloads (like AI model deployments) run within a cluster. It can place them in different parts of the infrastructure, such as different zones or nodes, to improve reliability.

However, users often find it challenging to take full advantage of these features. Improving the usability of Kubernetes can help users easily configure their model deployments to run where they need them for cost savings, reliability, and performance. This way, they can efficiently manage and scale their AI models within a cluster.



Model Serving

- **Resource Allocation for Model Serving**

When serving AI models, one of the main considerations is handling the model parameters, which are the core components of the model used for making predictions. The size of these parameters determines how much memory is needed. In most cases, except for extremely large models like trillion-parameter large language models (LLMs), only a portion of a GPU is needed to run the models efficiently. It's important to be able to use GPUs efficiently by allowing them to be shared among different tasks. Projects like DRA are working on making GPU scheduling more flexible to achieve this.

Response latency, or how quickly the model can provide results, varies depending on the application. For example, in autonomous driving, detecting objects on the road requires very low latency compared to tasks like generating images or poetry. To handle high-load situations with low latency requirements, additional serving instances may need to be launched on different computing resources like CPUs or GPUs. Kubernetes is evolving to support this kind of dynamic scheduling across different resources.

Event-driven hosting is an efficient way to use resources without wasting them, which helps keep costs down. Projects like Kubernetes Event Driven Autoscaling (KEDA) are designed for this purpose, scaling resources based on events or triggers. However, this approach needs to balance model loading latency with end-to-end service latency to ensure a seamless user experience.



Model Serving

- **Resource Allocation for Model Serving**

To improve model sharing and deployment, models can be delivered in a standardized format like the Open Container Initiative (OCI) format, which makes it easier to share and deploy models across different environments.

Another strategy is to use AI to predict resource usage and automatically adjust the number of serving instances to handle expected loads efficiently. This proactive approach optimizes resource allocation for model serving based on predicted demand.



Model Serving

- **Improving User Experience in Cloud Native AI**

Cloud Native (CN) technologies like containers and Kubernetes offer benefits like portability and scalability for deploying AI workloads. However, transitioning from traditional systems to these technologies can have a steep learning curve.

AI practitioners face challenges when configuring runtime environments for training models, especially with custom libraries. Default settings may not give optimal performance, and optimizing for one platform doesn't guarantee the same performance on another.

To provide a better user experience, tools and SDKs written in Python can hide the complexity of Kubernetes. Users want to build machine learning (ML) models with familiar frameworks like PyTorch and TensorFlow and easily deploy them on Kubernetes without dealing with Docker images or complex scaling tools.

Integrating tools like JupyterLab, which combines an IDE-like experience with ML APIs like Kubeflow Katib, helps ML practitioners iterate quickly without switching between different interfaces.



Model Serving

Bridging the gap between Big Data and ML ecosystems is crucial. Modern Generative AI models require large datasets, but tools for loading and preparing data need improvement. Solutions like RayData or data caching services can reduce overhead during training.

Deploying ML tools on Kubernetes can be complex, especially with GPUs. Simplifying the upgrade process and providing clear guidelines for resilience during upgrades is important.

For multi-tenancy, non-admin users need better visibility into available resources, ideally through tools like Grafana dashboards. Debugging is challenging in distributed environments, so providing clearer error messages and support for identifying and resolving failures is essential for AI practitioners.

In summary, improving the user experience in Cloud Native AI involves simplifying deployment, integration with familiar tools, enhancing data handling, and providing better support for debugging and resilience. This makes it easier for AI practitioners to focus on building and deploying models effectively.



Model Serving

Cross-Cutting Concerns in AI Pipeline Development

In addition to specific challenges at each stage of the AI pipeline, there are common issues that affect all stages and software applications. These include reference implementations, observability (monitoring), security, and more.

- **Reference Implementation:**

Cloud and AI are complex subjects, and integrating them effectively can be challenging due to the multitude of tools and projects available. Improving adoption requires a reference implementation that covers most basic use cases. For example, tools like Kind for Kubernetes and Jupyter Notebook have simplified development for cloud and AI/ML projects respectively. Similarly, we need a reference implementation for AI/ML pipelines running in the cloud.

- **Right-Sizing Resource Provisioning:**

AI/ML workloads are demanding on resources, especially with large models like large language models (LLMs) containing billions or trillions of parameters. GPUs, which are commonly used for acceleration, are costly and limited in availability. It's crucial to allocate resources efficiently to save costs and maximize utilization. This involves not only timeslicing GPUs but also partitioning them into smaller units and allocating them as needed for different workloads.



Model Serving

To address this challenge, Kubernetes introduced the Dynamic Resource Allocation (DRA) API in version 1.26.

This API provides more flexibility in managing specialized hardware resources, allowing for:

- Network-attached resources
- Customizable parameters for resource requests
- Custom setup and cleanup actions for resources
- Matching specific resource requests with available resources, including optional requests

The DRA API offers advantages such as allowing custom hardware to be added through drivers without modifying Kubernetes core code, defining resource parameters by vendors, and enabling resource sharing between containers and pods.



Model Serving

- **Cost Control**

AI/ML projects can be expensive, so it's crucial to automate resource allocation and scaling to optimize costs in the cloud. Using microservices allows scaling specific parts of your system as needed, and Kubernetes autoscaling can adjust the number of active instances based on demand, which helps reduce infrastructure costs. Leveraging Spot Instances with well-defined policies can also help balance cost savings with meeting Service Level Agreements (SLAs).

- **Observability**

Observability is key throughout the AI/ML pipeline. Cloud Native (CN) tools like OpenTelemetry and Prometheus can monitor various metrics such as system load, response latency, and access patterns. It's important to monitor the performance and health of AI models in production to detect issues like model drift, where a model's accuracy declines over time due to changing conditions (e.g., people wearing masks affecting facial recognition accuracy). Continuous monitoring helps identify performance issues and ensures models remain reliable.



Model Serving

Monitoring infrastructure is also critical, especially for long-running AI training jobs. Anomalies like GPU memory errors or network issues can occur and impact job performance. Deep diagnostics may be necessary to identify underlying issues not immediately apparent from standard metrics.

- **Disaster Recovery and Business Continuity**

All production services, including AI services, must be resilient to failures. Developing a comprehensive disaster recovery plan is essential to ensure business continuity. This plan may include strategies like data backup, running services across multiple availability zones, and using redundant instances to minimize downtime and mitigate risks. Implementing policies and procedures for disaster recovery helps protect against reputational damage and revenue loss caused by service disruptions.

Model Serving

- **Security and Compliance Audits**

For outward-facing AI/ML services like Model Serving instances, it's crucial to implement security measures such as firewall protection and access controls to protect against unauthorized access. AI/ML workloads should follow security best practices including penetration testing, vulnerability scanning, and compliance checks, especially for domains like healthcare or finance.

Tools like Gripe and Trivy can scan containerized workloads for vulnerabilities. Services like Kyverno enforce policies to ensure that containerized workloads run with minimal privileges and required capabilities.

Additional security layers can be implemented using technologies like confidential computing or Trusted Execution Environments (TEE), which provide encrypted memory and data integrity protection. These technologies safeguard sensitive data and AI models from other infrastructure users while in use.



Model Serving

- **Sustainability**

AI/ML model training consumes significant resources, particularly with large models like GPT-3, which can have a carbon footprint comparable to multiple transcontinental flights. The trend towards larger models for market dominance contributes to inefficiencies and increased energy consumption.

Efforts are underway to increase transparency and standardization in reporting the environmental impact of AI models. Innovations like DeepMind's BCOOLER and more efficient models like DistilBERT aim to reduce energy consumption in AI/ML.

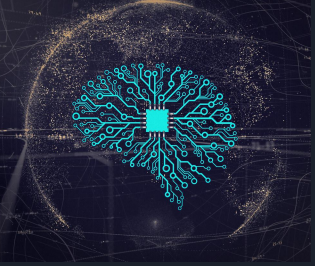
Adopting best practices such as efficient ML architectures, optimized processors, and locating cloud infrastructure in energy-efficient locations can help reduce the carbon footprint of ML training.



Education for Kids

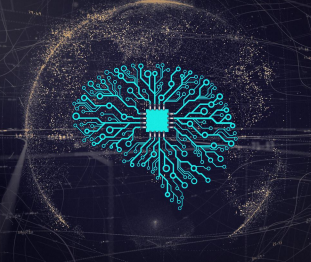
Current technology education often lacks focus on AI and modern development tools. Students may not have exposure to advanced IDEs or AI coding assistance technologies like GitHub's Copilot, which are becoming essential in modern development.

There's a misconception about AI technologies like ChatGPT and Copilot in schools, where they are often discouraged due to concerns about cheating. This prevents students from learning how to effectively use AI technologies to augment their work and develop important skills.



PATH FORWARD WITH CLOUD NATIVE ARTIFICIAL INTELLIGENCE





PATH FORWARD

- Recommendations (or Actions)
 - Flexibility
 - Sustainability
 - Custom Platform Dependencies
 - Reference Implementation
 - Industry Acceptance of Terminology
- Existing Solutions (CNAI Software)
 - Orchestration - Kubeflow
 - Context - Vector Databases
 - Observability - OpenLLMetry
- Opportunities for Further Development
 - CNCF Project Landscape
 - CNAI for Kids and Students
 - Participation
 - Trust and Safety / Safety By Design
 - The Emergence of a New Engineering Discipline



Recommendations for Cloud Native AI

- **Flexibility:**

Embrace the existing tools and techniques that are popular and effective in the realm of AI.

Utilize REST interfaces to interact with cloud-based resources and services, leveraging the flexibility of cloud-native technologies.

- **Sustainability:**

Address the environmental impact of AI workloads, especially in cloud-native environments.

Support projects and methodologies that assess and improve the ecological sustainability of AI.

Integrate cloud-native technologies for optimized workload scheduling, autoscaling, and tuning.

Advocate for standardized methodologies to assess the environmental impact of AI.

Develop energy-efficient AI models and promote transparency in model development and usage using cloud-native platforms like Kubeflow.

Emphasize purposeful and efficient AI usage to reduce unnecessary computational loads.



Recommendations for Cloud Native AI

- **Custom Platform Dependencies**

Recommendation: Ensure that your Cloud Native environment supports the necessary GPU drivers and GPU acceleration for AI workloads.

Importance: AI applications often rely on specific frameworks and library versions that may not be readily available or compatible with standard container images due to different vendors and GPU architectures.

- **Reference Implementation**

Recommendation: Consider creating a user-friendly reference implementation using Cloud Native technologies like OpenTofu.

Purpose: This implementation combines various open-source tools (e.g., JupyterLab, Kubeflow, PyTorch, Spark/Ray/Trino, Iceberg, Feast, MLFlow, Yunikorn, EKS/GKE, S3/GCS) to provide a product-like experience for teams worldwide to start AI/ML projects quickly in the Cloud.

Benefits: By integrating these tools, teams can efficiently scale up their machine learning projects using Cloud-based technologies, advancing open and responsible AI/ML development.



Recommendations for Cloud Native AI

- **Industry Acceptance of Terminology**

Evolution of Language: As AI becomes more widespread, terminology around AI is evolving to make discussions and communication easier for businesses.

Simplification and Standardization: This evolution includes simplifying technical terms and introducing new business-friendly terms (e.g., "repurpose" for reusing content).

Examples of Technical Terms: Technical terms like RAG (Reason, Action, and Generalization) are part of this evolving language used in AI discussions.



Existing Solutions (CNAI Software)

Evolving Solutions for AI/ML:

New tools and technologies are emerging to support AI and Cloud Native Artificial Intelligence (CNAI). One such tool is

- **Orchestration - Kubeflow**

Kubeflow, designed specifically for ML Operations (MLOps). Kubeflow leverages Kubernetes, stateless architectures, and distributed systems to help AI and ML communities adopt Cloud Native tools effectively. It integrates well with Kubernetes, applying machine learning concepts to elastic infrastructures. Kubeflow follows Kubernetes best practices like declarative APIs and composability, implementing individual microservices for each stage of the ML lifecycle. For instance, it offers the Kubeflow Training Operator for distributed training, Katib for hyperparameter tuning, and KServe for model serving, providing flexibility to integrate these components into existing ML infrastructure or use Kubeflow as a comprehensive ML platform.



Existing Solutions (CNAI Software)

- **Context - Vector Databases**

Large Language Models (LLMs) are trained using vast amounts of publicly available data and are interacted with via prompts. To enhance the value of responses without requiring longer or multiple prompts, vector databases play a crucial role. These databases store indexed vectors, which are numerical representations of data. Embeddings, specific vector representations of data, capture relationships and similarities between represented data. When a user provides an LLM prompt, it's transformed using the same embedding used by the vector database. The resulting vector is used to find similar vectors in the database, providing additional context before feeding into the LLM to generate a response. Multi-modal Generative AI (GenAI) systems can handle prompts in various formats like text, images, or audio, with embedding capabilities for diverse inputs.

Vector databases can be purpose-built or traditional databases extended to handle vectors more effectively. They differ in indexing schemes, distance metrics for similarity computation, and data compression techniques. Some notable offerings include Redis, Milvus, Faiss, and Weaviate



Existing Solutions (CNAI Software)

- Observability - OpenLLMetry

OpenLLMetry is a project that helps monitor and understand Large Language Models (LLMs) using a tool called Open Telemetry. Since it's hard to debug Generative AI models like LLMs by stepping through code, developers use observability tools to gather insights over time. The data collected from observing LLMs is crucial for evaluating and improving their performance. OpenLLMetry extends OpenTelemetry to provide specialized monitoring and insights for Generative AI systems.



Opportunities

- **CNCF Project Landscape**

The CNCF (Cloud Native Computing Foundation) and other Linux Foundation groups like LF AI & Data, along with partners such as the AI Alliance, offer a central space for AI projects that both AI and cloud engineers can leverage. Tools like the Cloud Native Landscape provide an overview of the Cloud Native ecosystem, showcasing various projects categorized by their specific functions and areas of focus. This landscape serves as a resource hub for exploring existing and emerging projects related to AI and cloud technologies

The following figure lists established and evolving projects grouped by their functional area.

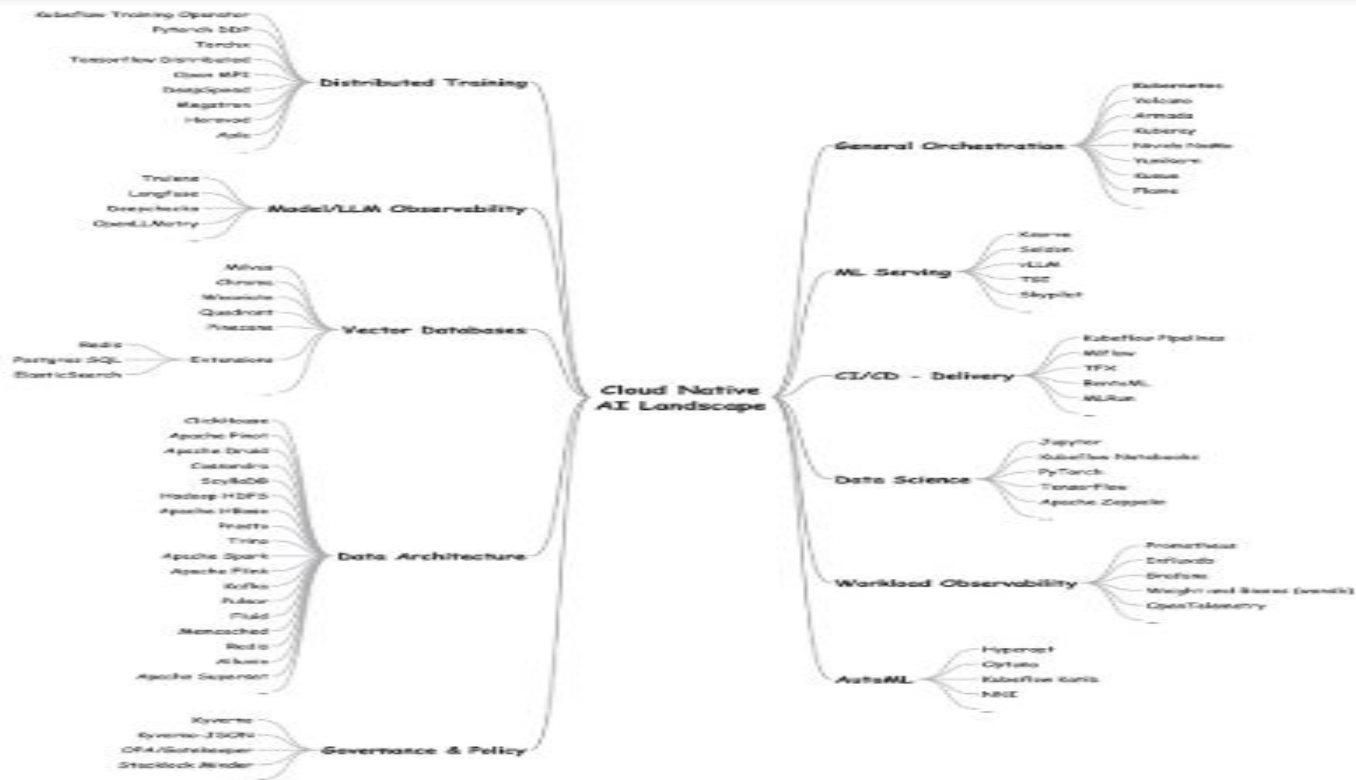


Figure 4
ML Tool to Task Mind Map



Opportunities

- **CNAI for Kids and Students**

Children already use AI tools like ChatGPT daily without fully understanding how they work. The inner workings of modern AI, including different types of AI algorithms, are like a mystery to kids and even to tech-savvy parents. It's hard to get interested in something when you don't really understand it. Instead of just accepting AI tools like ChatGPT as magic, students should learn the basics of neural networks and machine learning algorithms. This helps them understand how AI works and how they can use it better in the future.

The Cloud Native community hosts educational events like CNCF Kids Day at KubeCon, which teach kids about Cloud Native and AI technologies. Getting kids interested in AI early can also help address issues of diversity, equity, and inclusion in computer science. AI is a technology that anyone can use and contribute to, regardless of their background.

AI technology is becoming as important as the internet was during the dot-com era. Just like how everyday people learned to use web technology to improve their work, AI is now becoming part of our daily lives. It's crucial that students keep up with these advancements in AI and Cloud Native technologies.



Opportunities

- **Participation**

As AI continues to grow, there are more opportunities for education and involvement. There are specific roles for AI specialists (like machine learning researchers and data scientists) and for AI generalists (like operators and regular users). Educational programs and certifications focus on teaching AI tools and techniques. Professional societies and meetups provide chances to learn and discuss AI challenges. Industry groups like CNCF and Linux Foundation AI coordinate big AI projects and protocols.



Opportunities

Trust and Safety / Safety By Design

Trust and Safety:

As we create AI and Cloud Native technology, there's a real risk of unintended problems and harmful effects. Sometimes, this happens because of design issues that weren't meant to be harmful but end up impacting vulnerable groups. For example, algorithms that accidentally promote hate speech or violence. Other times, people deliberately use these technologies for harm, like using AI to spread fake news or creating inappropriate content.

AI and Cloud Native tools are also key to making online spaces safer. Trust and Safety teams use technology to manage content, scan for risks, and protect users. However, if these systems aren't carefully designed, they can cause more harm than good.

Responsible technology means reducing harm from technology, making the tech world more diverse, and ensuring that tech serves the public interest. As we develop AI and Cloud Native technology, we need to think about ethical impacts and human rights. This includes things like privacy, freedom of speech, and the right to be safe online.



Opportunities

Safety By Design:

means putting user safety and rights first when creating online products and services. It's about preventing problems before they happen and making sure that organizations prioritize safety and accountability. This approach aims to make online experiences more positive and respectful for everyone.

There are experts and organizations focused on these best practices, like the Global Internet Forum to Counter Terrorism and The Tech Coalition. They help tech companies build safer products and services. OpenAI, which created ChatGPT, was initially started as a non-profit to ensure safety and fairness in AI.

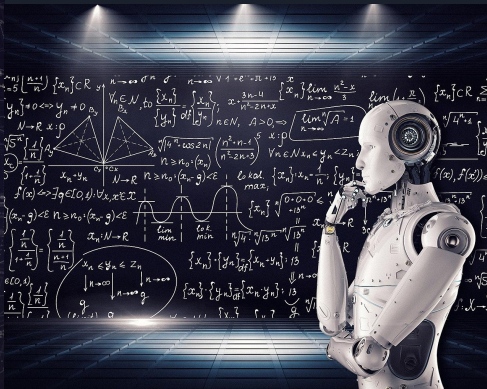
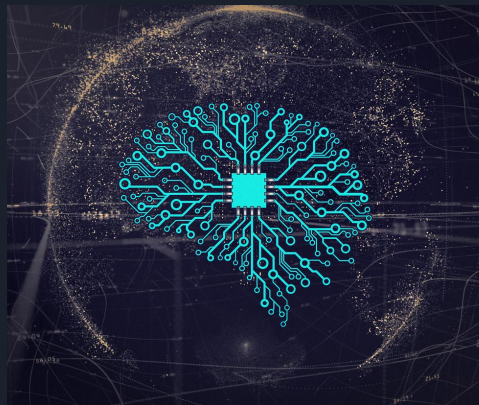


Opportunities

The Rise of New Engineering Roles:

Over the last two decades, the tech industry has seen new engineering roles emerge rapidly. We've seen roles like DevOps Engineer and Infrastructure Engineer become crucial. In the near future, we expect roles like MLDevOps or AI Engineer to bridge Data Science, Infrastructure, and Development. This industry area is evolving, and job titles may change over time. These roles will focus more on AI technology, infrastructure, and deploying AI solutions.

CONCLUSION





Conclusion

Combining Artificial Intelligence (AI) with Cloud Native (CN) technologies presents a unique opportunity for organizations to enhance their capabilities. Cloud Native infrastructure offers scalability, resilience, and ease of use, allowing for more efficient training and deployment of AI models at a larger scale. This paper has explored the intersection of AI and CN, discussing current challenges, opportunities, and potential solutions for organizations to leverage this powerful combination.

Although challenges persist, such as managing resource demands for complex AI workloads and ensuring AI model reproducibility and interpretability, the Cloud Native ecosystem continues to evolve to address these issues. Projects like Kubeflow, Ray, and KubeRay are paving the way for a more unified and user-friendly experience in running AI workloads in the cloud. Ongoing research into GPU scheduling, vector databases, and sustainability also holds promise for overcoming limitations.

As AI and Cloud Native technologies advance, organizations that embrace this synergy will gain significant competitive advantages. The potential applications are vast, ranging from automating tasks and analyzing large datasets to generating creative content and personalizing user experiences. By investing in talent, tools, and infrastructure, organizations can harness AI and Cloud Native technologies to drive innovation, optimize operations, and deliver impactful outcomes.