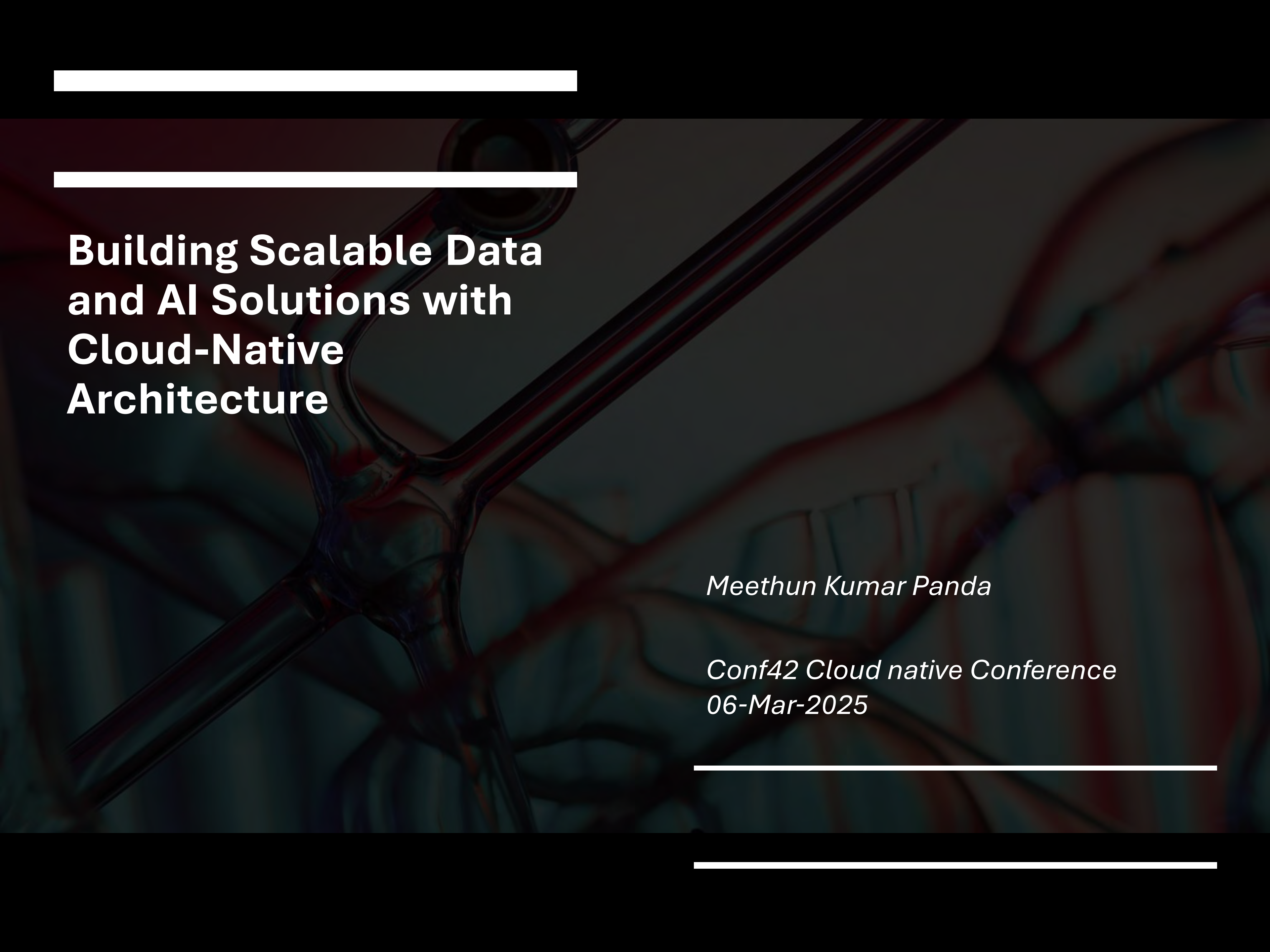

Building Scalable Data and AI Solutions with Cloud-Native Architecture
Meethun Kumar Panda
Conf42 Cloud native Conference
06-Mar-2025

Scalable Data and AI Solutions in the Cloud is key to accelerate business value

Build foundation for analytics

Build D&A platform capabilities incrementally to realize the use cases

Cloud-native architecture

Achieve elasticity by scaling up or down automatically based on workload demands; Achieve fault tolerance through high availability and self-healing mechanisms

Enable rapid value creation

Enable new ways of working for rapid product iteration cycle with cost efficiency through pay-as-you-go approach

Scale up seamlessly

Scale up resources seamlessly to accelerate use cases leveraging optimal infrastructure

Ensure governed accessible data

Ensure curated and governed data is easily accessible from various data domains

Future proof digital advantage

Multi-cloud strategies for workload portability; Able to integrate future data sources conforming to the data governance rules and policies

Understanding Cloud-Native Architecture: Key Principles and Benefits

Scalability

- Cloud-native systems dynamically adjust resources based on demand
- Example – A video streaming platform (e.g., Netflix) scales its infrastructure during peak hours when more users are online and scales down during off-peak hours to save costs

Resilience

- Systems recover from failures automatically without disrupting services
- Example –
  If a server hosting a banking AI chatbot crashes, the system reroutes traffic to another healthy server, ensuring uninterrupted customer service.

Automation

- DevOps and Infrastructure as Code (IaC) automate deployments, updates, and scaling.
- Example – A cloud-based fraud detection AI system continuously updates models in production without manual intervention using CI/CD pipelines.

Flexibility

- Modular API-driven architectures for interoperability across cloud platforms
- Example – A healthcare provider uses AWS for storage, Google Cloud for AI

Cloud-native architecture leverages microservices, containers, orchestration tools, serverless computing, and DevOps CICD to achieve lower costs, faster innovation, and optimized AI performance

Microservices and Containerization: Enabling Scalability and Resilience

Microservices - break down applications into independent services that communicate via APIs

1. **Independent scaling**: Scale AI inference separately from data ingestion.
2. **Fault isolation**: If one service fails, others remain operational.
3. **Faster deployments**: Modify a single AI model service without redeploying the entire application.

Example Use Case:

Spotify uses microservices and Kubernetes to scale its AI-powered music recommendation engine without downtime

Containerization

- Consistency across environments – AI models and data pipelines run identically across dev, testing, and production
- Kubernetes automation enabling scaling, orchestration, and self-healing of AI workloads.
- Service Mesh (Istio, Linkerd) for microservices observability, load balancing and security.
- Sidecar AI model deployment for independent scalability of ML inference services

Serverless Computing: Enhancing AI and Data Workflows with On-Demand Resources

1 **Serverless computing** enables AI applications to execute code without managing infrastructure. (e.g., AWS Lambda, Azure Functions, Google Cloud Functions);

Serverless GPUs (GPU-as-a-Service, Inference-as-a-service) provide cost-efficient AI model inference without the overhead of dedicated infrastructure.

2 **Key Benefits**:

- **Auto-scaling**: Dynamically allocates resources based on usage.
- **Cost efficiency**: Pay only for execution time, reducing idle costs.
- **Faster deployment**: Eliminates provisioning and setup complexities.

Example: A chatbot running on AWS Lambda scales automatically when users interact with it and scales down when idle, cutting infrastructure costs.

Common use cases:

- AI powered chatbots
- 3 AI model inference with event-driven triggers.
- Real-time data processing using serverless ETL pipelines.

Storage and Data Management: Handling Large-Scale AI workloads in the Cloud

Storage and Data Management to handle large scale data processing

- **Data lakes & Data warehouses**:
  - Data lakes (AWS S3, Azure Data Lake, Delta lake) store unstructured raw data for AI/ML models.
  - Data warehouses (BigQuery, Snowflake) store structured, query-optimized data for analytics.
  - Vector Database to store embeddings for GenAI
- **Data processing tools**:
  - Apache Spark, Dataflow, Databricks for large-scale batch and stream processing.
- **Scalability strategies for AI data management**:
  - Distributed databases (NoSQL, NewSQL) for real-time AI workloads.
  - Tiered storage & lifecycle policies to optimize cost and performance.
- **Example**: A self-driving car company stores petabytes of sensor data in a cloud data lake for AI model training.

MLOps to automate and Scale AI workflows

- **MLOps** integrates DevOps best practices into AI model development, deployment, and monitoring.
- **Key components**:
  - **Automated CI/CD pipelines** for AI models.
  - **Feature stores** for scalable feature engineering
  - **Model versioning and governance** to track performance over time.
  - **Monitoring and bias detection** to prevent drift.
- **Example**: A bank uses MLOps to continuously update fraud detection models, preventing new types of cyber fraud in real time.

Security and Compliance in Cloud-Native AI Solutions

Key Security and compliance

Data Privacy

- Implement differential privacy to prevent data leakage while maintaining model utility.
- Use data masking and anonymization for handling PII (Personally Identifiable Information).
- Comply with GDPR, HIPAA, and CCPA for AI-driven applications handling sensitive data

Data Encryption

- Utilize AES-256 encryption for data at rest and TLS 1.3 for data in transit.
- Leverage homomorphic encryption for privacy-preserving AI model training on encrypted data.
- Implement hardware security modules (HSMs) for cryptographic key management.

Zero Trust Architecture

- Enforce least privilege access using RBAC (Role-Based Access Control) and ABAC (Attribute-Based Access Control).
- Implement continuous authentication with AI-driven behavioral analytics for anomaly detection.
- Use secure enclave computing (e.g., AWS Nitro, Intel SGX) to protect AI workloads.

API Vulnerabilities

- Secure AI model endpoints using OAuth2, JWT (JSON Web Tokens), and mTLS (Mutual TLS).
- Implement WAF (Web Application Firewalls) and API gateways (e.g., Kong, Apigee, AWS API Gateway) to protect AI services.
- Regularly conduct penetration testing and runtime API threat monitoring to detect unauthorized access.

Performance Optimization: Ensuring Speed and Efficiency in AI and Data Pipelines

Right sizing infrastructure

How to design an auto-scale AI workloads based on demand?
e.g., Implement auto-scaling mechanisms using Kubernetes, KServe, or AWS Auto Scaling for dynamic workload management.

Model optimization

What are some techniques to improve model training and inference speed?
e.g., Apply quantization (INT8, FP16, BF16) to reduce model size while maintaining accuracy.

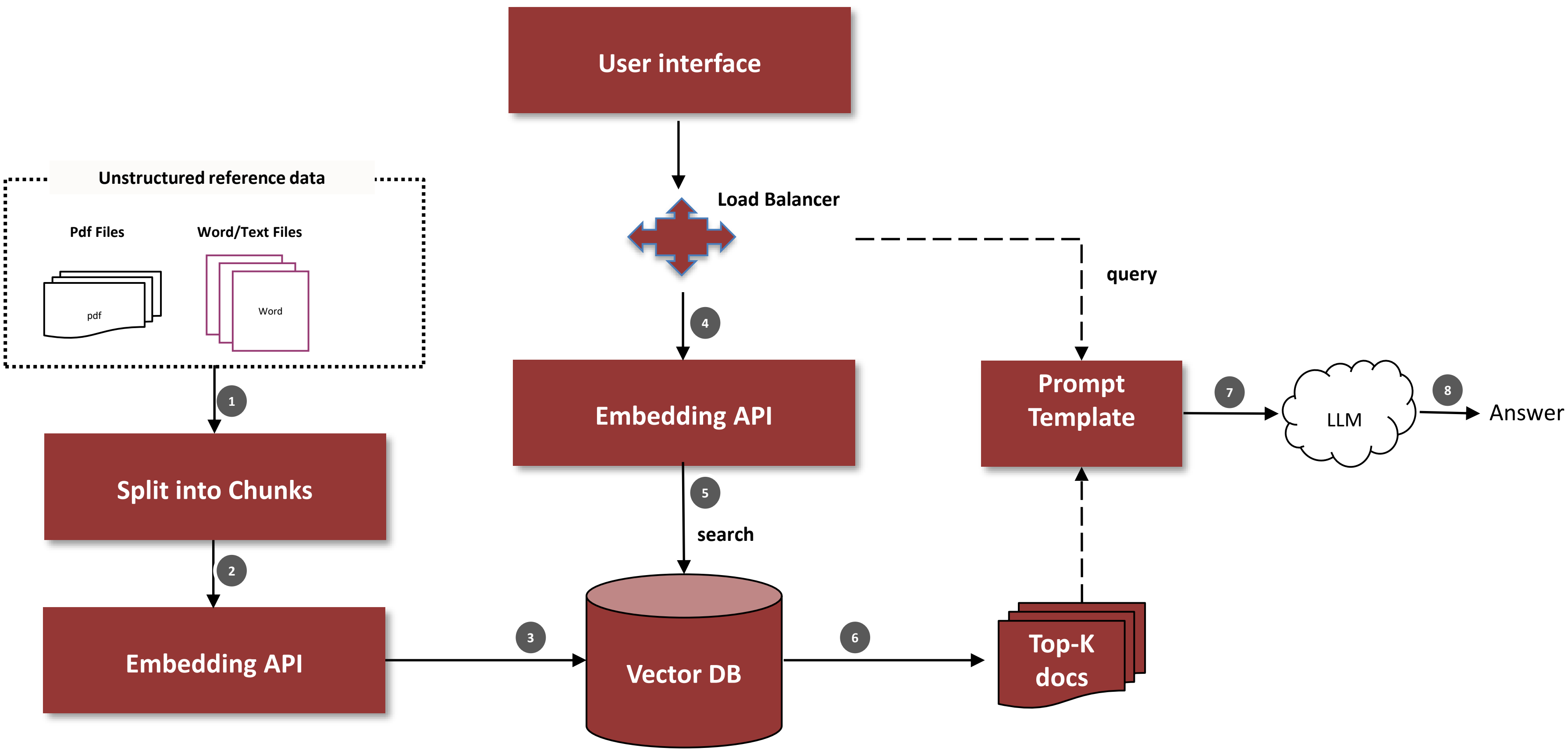
Data locality

Can the data be stored and processed in the same cloud region to reduce latency?
e.g., data sharding and partitioning to optimize distributed AI training and inference workloads.

AI acceleration frameworks

How can we accelerate large language model development, and inference?
e.g., NVIDIA Triton, Hugging Face Optimum for LLM performance tuning, Groq for faster inference

RAG based architecture leveraging cloud native capabilities



- Load balancer to ensure even distribution of requests (e.g., API gateway, NGINX)
- Each front-end, back-end, Vector DB, Prompt manager, LLM services are docker containerized to enable portability
- Container orchestration (K8S) to manage containers with Horizontal Pod Autoscaler (HPA) capability to ensure services scale up/down based on demand.
- Security best practices - Istio for service mesh, OAuth for API authentication, data encryption

- Vector database runs as a stateful K8S service with persistent storage
- LLM is hosted on GPU nodes or optimized inference services (e.g., AWS SageMaker, Azure ML)
- LLM inference can be further optimized by autoscaling inference pods based on request load.
- Prometheus & Grafana monitor API latency, database queries, and resource utilization. Kibana & Elasticsearch track system logs and errors

Future Trends: The Evolving Landscape of Cloud-Native AI and Data Solutions

AI-powered cloud automation: Self-optimizing cloud architectures.

Edge AI and Hybrid Cloud: Processing AI closer to the data source.

LLMOps: Optimization strategies for large-scale generative AI models.

Quantum Computing & AI: Emerging potential of quantum-enhanced AI workloads.