femington

Int. J. Intell. Syst. Data Sci. (2026) 1(1):1–11
Article ID: IJISDS-2026-050

Cloud-Native Distributed AI: Enabling Secure Collaborative Learning on Heterogeneous Infrastructure

Shiva Kumar Bommakanti
sbommakanti2@gmail.com
Independent Researcher

Abstract

This paper presents a foundational reference architecture for a cloud-native platform engineered to facilitate decentralized, privacy-preserving artificial intelligence model development. Rather than proposing a novel framework, this work synthesizes existing federated learning (FL) systems into a cohesive architectural blueprint that addresses the critical need for secure, collaborative intelligence without direct data centralization. The paper's primary technical contribution lies in its layered decomposition of distributed FL components, spanning control planes, execution engines, and secure communication channels, and their integration with cloud-native orchestration principles. The discussion emphasizes scalable operational capabilities across heterogeneous cloud infrastructure, detailing secure provisioning mechanisms and adaptable communication patterns. The described system is poised to transform complex collaborative AI tasks from isolated development into practical, cloud-native deployments, highlighting its significance for enterprise-grade distributed AI solutions.

Keywords

• Federated Learning (FL) • Cloud-Native Architecture • Distributed Artificial Intelligence • Privacy-Preserving Machine Learning • Secure Aggregation and Communication • Heterogeneous Infrastructure Orchestration

1. Introduction to Cloud-Native Federated Learning

The exponential growth of data generated by edge devices, coupled with increasing concerns over data privacy and sovereignty, has necessitated a paradigm shift from centralized machine learning to decentralized collaborative learning techniques. Federated Learning (FL), first formalized in [1], enables multiple clients to collaboratively train a shared model without exposing their private data. This approach has gained traction in regulated industries such as healthcare [2], finance [3], and telecommunications [4], where direct data sharing is either legally restricted or commercially sensitive.

However, traditional FL frameworks are often constrained by assumptions of homogeneous environments and uniform compute capabilities. In contrast, real-world applications increasingly involve heterogeneous infrastructure, spanning mobile devices, on-premise clusters, public clouds, and edge servers. These heterogeneities introduce challenges related to orchestration, scalability, security, and reliability [5].

Cloud-native technologies, including containerization (e.g., Docker), orchestration frameworks (e.g., Kubernetes), and declarative infrastructure-as-code tools, provide a promising solution for overcoming these challenges [6]. By abstracting the deployment and management of FL components, cloud-native

FL systems offer improved portability, resource elasticity, and maintainability across diverse computing environments.

Moreover, secure and authenticated communication between federated entities is vital for maintaining trust in collaborative training settings. Federated platforms often adopt mutual TLS (mTLS), certificate-based identity verification, and role-based access control (RBAC) to prevent unauthorized participation and data leakage [7]. These security layers are typically integrated into the control plane and job execution subsystems.

The statistical heterogeneity of data across clients presents another major challenge. Non-IID data distributions can significantly impact convergence behavior and degrade global model performance. Various optimization techniques such as FedAvg [1], FedProx [5], and SCAFFOLD [8] have been proposed to mitigate such effects by modifying aggregation rules or introducing client-specific control variates.

To support a wide range of learning paradigms, cloud-native FL frameworks often expose extensible plugin architectures. These plugins abstract roles such as model training, privacy filtering, job monitoring, and communication logic. Modular designs such as those found in NVFlare and Flower facilitate integration with domain-specific requirements while maintaining interoperability across clients [9, 10].

In contrast to classical centralized FL topologies, recent designs explore decentralized or partially decentralized approaches. Architectures such as hierarchical FL, peer-to-peer FL, and federated split learning [11] reduce bottlenecks associated with a single aggregator and enhance fault tolerance. Cloud-native designs benefit from dynamic service discovery, automatic job restarts, and stateless communication layers that enable flexible reconfiguration during runtime.

Containerized execution environments are key to achieving uniform behavior across clients. Whether running on a GPU-equipped hospital server or a CPU-limited mobile device, containerized workflows ensure consistent runtime behavior, dependency resolution, and security policies. This also facilitates Continuous Integration/Continuous Deployment (CI/CD) for federated jobs and components.

To accommodate domain-specific use cases, modern FL ecosystems must support integration with diverse ML backends. Frameworks such as PyTorch, TensorFlow, and XGBoost are frequently used in privacy-sensitive domains. Cloud-native FL platforms expose backend-agnostic interfaces to ensure seamless integration of models, regardless of their underlying frameworks.

The subsequent section presents a layered architectural decomposition of such systems. It elaborates on how distributed executors, orchestration components, and secure communication channels coordinate to deliver scalable, fault-tolerant, and privacy-preserving federated learning pipelines across heterogeneous infrastructure.

Contributions: This manuscript distinguishes itself from prior surveys and architectural descriptions by providing a holistic reference architecture that explicitly maps federated learning components to cloud-native primitives. The unique contributions include: (i) a layered decomposition of FL systems that aligns with containerized microservices architecture, (ii) a detailed examination of secure communication protocols within the context of zero-trust networking models, and (iii) an integrated perspective on optimization, privacy, and deployment that bridges theoretical FL research with enterprise-grade implementation practices. This synthesis provides practitioners with a structured roadmap for implementing production-ready federated systems on heterogeneous infrastructure.

2. Architecture of Distributed Federated Systems

A cloud-native federated learning system must balance modularity, extensibility, and secure communication across clients operating in diverse network environments. The architectural backbone typically comprises

a control plane, execution engine, client-server orchestrator, and plugin system, each abstracted for reusability and scalability [7, 10].

The control plane governs the lifecycle of federated jobs. It maintains the job state, handles scheduling, initiates task distribution, and collects execution metadata. Federation tasks are typically organized as multi-phase workflows, each defined by a sequence of actions such as training, evaluation, and model aggregation [12].

A central orchestrator communicates with clients via secure channels, typically over gRPC with mutual TLS authentication. Client endpoints register securely, and their capabilities are discovered dynamically through a service registry. Federated job metadata, such as model configurations, role assignments, and job-specific plugins, is dispatched prior to execution.

Clients host an execution engine capable of interpreting job configurations and managing plugins associated with specific roles (e.g., model trainer, statistics aggregator). These plugins abstract the computation and allow flexibility in defining task behavior. The plugin interface supports Python-based modules and handles initialization, execution, and state transitions [9, 10].

Figure 1 presents a high-level overview of the architecture. The server node executes the controller logic and dispatches tasks to remote clients. The clients report status, metrics, and partial updates back to the server in a synchronous or asynchronous mode, depending on topology.

To ensure modularity, each component is loosely coupled and can be dynamically replaced or upgraded. For instance, the training pipeline can be swapped for another model without altering the orchestration or security stack. This separation of concerns is a key architectural tenet in cloud-native software [6].

The job lifecycle includes several stages: initialization, provisioning, execution, aggregation, and finalization. These are configured through a centralized job YAML or JSON specification that defines task phases and links them to corresponding executors. Each phase may include multiple participants or be restricted to specific roles.

Fault tolerance is achieved via stateless execution agents that checkpoint local state and enable recovery in case of interruption. Clients experiencing delays or dropped connections are either excluded temporarily or assigned retry logic based on runtime policies [7].

Persistent logging and observability tools integrated with the architecture help diagnose failures and measure job health. Logs, model snapshots, and performance metrics are typically pushed to a centralized repository or a third-party monitoring system such as Prometheus or ELK stack.

Figure 1 illustrates the fundamental topology of a federated learning system in a cloud-native context. The diagram highlights three key components: (i) the central controller (aggregator server) responsible for model distribution and update aggregation, (ii) multiple client nodes executing local training on private data, and (iii) secure communication channels established via mutual TLS. The bidirectional arrows represent the two-phase communication pattern, model dissemination from server to clients followed by gradient or weight updates from clients to server. This architecture enables parallel training across distributed data sources while maintaining data locality.

Listing 1 presents an exemplary job specification structure. The YAML configuration defines the training workflow phases, model registry location, client selection policies, and aggregation strategy. Each phase specifies its executor type (e.g., “Trainer”, “Evaluator”), participant roles, and hyperparameters. This declarative approach enables version control, reproducibility, and automated CI/CD integration for federated jobs.

Listing 1: Example federated job specification in YAML

```
apiVersion: federated.ai/v1
kind: TrainingJob
metadata:
  name: medical-imaging-classifier
spec:
  phases:
    - name: initialization
      executor: ModelDistributor
      modelSource: registry:latest/model:v2
    - name: local-training
      executor: PyTorchTrainer
      participants: ["client-*"]
      hyperparameters:
        epochs: 5
        batchSize: 32
        learningRate: 0.01
    - name: secure-aggregation
      executor: FedAvgAggregator
      aggregationStrategy: weighted_by_samples
      minClientsRequired: 3
    - name: validation
      executor: ModelEvaluator
      datasetRef: holdout-set
  privacy:
    differentialPrivacy:
      epsilon: 2.0
      delta: 1e-5
```

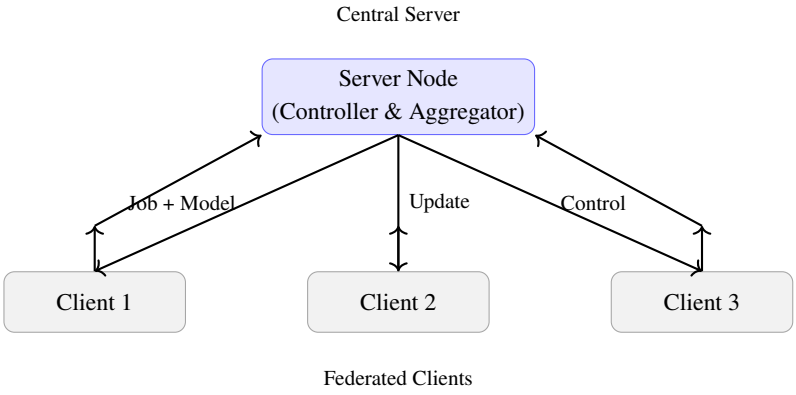

Figure 1: Simplified federated learning topology showing the central aggregator server (top) connected to multiple distributed clients (bottom). The architecture enables parallel local training on private datasets followed by secure model update transmission. This hub-and-spoke pattern represents the foundational communication topology upon which cloud-native FL systems are built.

3. Secure Communication and Operational Resilience

In federated learning (FL) environments, particularly those involving cross-organizational or cross-border collaborations, security and system resilience are paramount. A breach in communication integrity or model confidentiality may expose sensitive information, undermine trust, or even compromise regulatory

compliance. Consequently, secure protocols and fault-tolerant designs are integral to the infrastructure of distributed FL platforms [13, 14].

The communication pipeline between the controller and clients is typically secured using Transport Layer Security (TLS) with mutual certificate-based authentication. This ensures that both entities validate each other's identity before initiating any data exchange. Certificates are rotated periodically to minimize the risk of long-term key compromise [5].

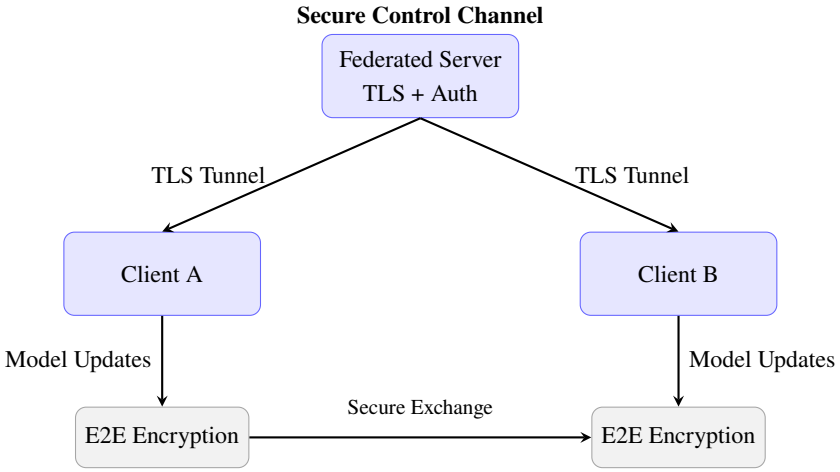

Figure 2: Secure communication architecture in federated learning. Each client establishes a mutually authenticated TLS tunnel. Encrypted payloads are transmitted through secure channels, ensuring end-to-end data protection.

Federated learning systems often leverage authenticated key exchange protocols to ensure perfect forward secrecy (PFS). The ephemeral nature of session keys limits the impact of intercepted traffic even in the event of private key leaks. Additionally, payloads, including model updates and configuration packets, are signed and encrypted end-to-end to prevent tampering during transit. To enhance operational resilience, FL architectures employ asynchronous messaging queues and event-driven schedulers. These systems allow decoupled components to operate independently, thus tolerating network delays or temporary outages. For instance, if a client goes offline mid-training, its updates are skipped or delayed without halting the entire federated round [15].

Replay attacks are mitigated by embedding cryptographic nonces and timestamps in each message packet. These allow servers to verify the freshness of received data and reject duplicate or out-of-order transmissions. Audit trails are maintained at both ends using tamper-evident logs to support post-incident forensic analysis and compliance auditing. Figure 2 presents an abstract overview of the security mechanisms involved in federated learning communication. It includes TLS tunnels, mutual certificate authentication, encrypted payload transmission, and secure aggregation of model updates.

Redundancy and fault isolation are also critical in production deployments. Each client runs within an isolated container or virtual machine that restricts external access. This compartmentalization ensures that a fault or compromise on one node does not cascade across the federation.Federated systems also support adaptive client selection and retry strategies. Based on historical participation metrics or system load, unreliable clients may be excluded from certain training rounds, while others are retried with a backoff strategy. These mechanisms improve convergence stability and reduce the variance in model performance [7]. Monitoring subsystems track message delivery success, round completion latency, and cryptographic validation failures. Alerts are generated in response to anomalies, which may indicate

adversarial behavior, malfunctioning nodes, or misconfigurations. Incident response workflows can then trigger automated rollback or client revocation.

4. Optimization Techniques in Federated Learning

Optimization in federated learning presents unique challenges due to decentralized data, limited communication bandwidth, and system heterogeneity. Unlike centralized settings, federated optimization must operate under partial observability, asynchronous updates, and constrained computation at the edge [4].

A widely adopted approach in this domain is Federated Averaging (FedAvg), which performs multiple local gradient descent steps on each client before aggregating models globally. This reduces communication rounds but may suffer from client-drift in non-IID settings. Modified versions such as FedProx introduce a proximal term to constrain local updates and stabilize training [16].

Gradient compression techniques—including quantization and sparsification—are employed to reduce uplink bandwidth usage. These methods ensure that only the most significant updates are transmitted to the server. Randomized sparsification and top-k selection strategies have demonstrated effectiveness in reducing communication costs without major performance loss [17].

Adaptive optimization algorithms like Adam or Yogi have been adapted for federated setups to accelerate convergence. However, client heterogeneity poses a risk of overfitting local data distributions. Server-side momentum and adaptive learning rate schedulers are used to counterbalance such divergence [18].

In cross-device federated learning, the selection of active clients per round affects optimization significantly. Strategies such as importance sampling, availability-aware sampling, or contribution-based scheduling have been proposed to ensure fairness and convergence [19]. Figure 3 outlines the optimization process. Clients perform local model training using their data shards and return compressed or full gradients to the server, which then aggregates and updates the global model. Federated Multi-task Learning (FMTL) is another optimization paradigm wherein clients learn personalized models with a shared representation backbone. Methods such as MOCHA and FedPer explore this direction by formulating optimization as a convex multi-task objective under privacy constraints.

System-aware optimization has recently gained traction. Here, model architectures and training strategies are dynamically adapted based on client capabilities such as CPU utilization, memory constraints, or network speed. These adaptive policies optimize both model performance and resource efficiency.

Robust aggregation functions, such as median or trimmed mean, are adopted to mitigate the influence of noisy or malicious updates. Byzantine-resilient aggregation ensures convergence even in the presence of adversarial clients [20].

5. Privacy Techniques and Secure Aggregation

Preserving privacy in federated learning (FL) is essential, particularly in sensitive domains such as healthcare, finance, or defense. Unlike traditional centralized learning, FL avoids raw data centralization; however, model updates themselves may still leak private information [21].

Differential privacy (DP) provides a theoretical framework for limiting the risk of information leakage by adding random noise to gradients or model weights before transmission. A privacy budget, denoted by (ε, δ), quantifies the allowable disclosure risk [22]. DP mechanisms are often applied at the client-side prior to encryption, ensuring end-to-end privacy compliance.

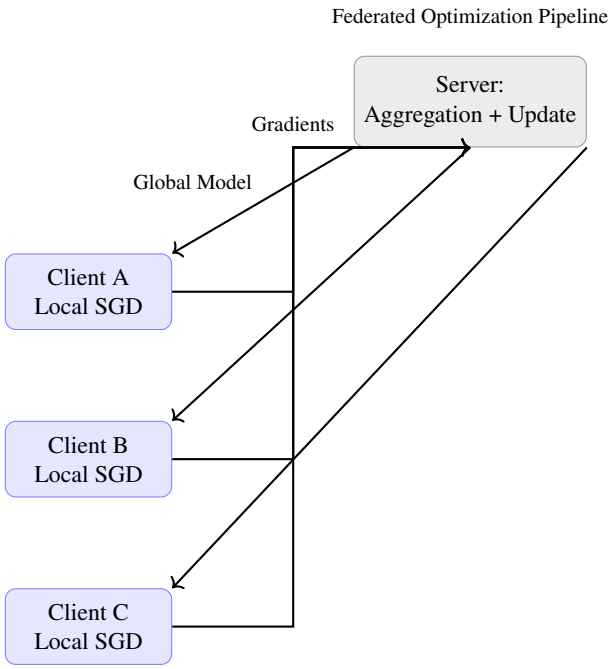

Figure 3: Federated optimization workflow depicting the iterative training process. Clients perform multiple local gradient descent steps on their private data shards, then transmit compressed updates to the central server. The server aggregates these contributions (typically via FedAvg or variants) to produce an improved global model, which is redistributed for subsequent training rounds.

Secure Aggregation (SecAgg) is a cryptographic protocol that enables the server to compute the aggregate of multiple encrypted inputs without learning individual contributions [23]. Clients generate pairwise random masks that cancel out during aggregation, and intermediate values remain unintelligible to any party.

Homomorphic Encryption (HE) allows computation on ciphertexts. Though computationally intensive, it supports privacy-preserving model training in scenarios where parties cannot access plaintext data or weights. Partial HE (e.g., Paillier, BFV) is often used to perform addition or scalar multiplication securely [24].

Another technique is Secure Multi-party Computation (SMPC), where multiple parties jointly compute a function without revealing their private inputs. In FL, SMPC is used to perform aggregation functions securely, especially when a trusted server is unavailable [25].

Figure 4 provides a visual representation of privacy-preserving components in a federated setup. Each client locally applies differential privacy, encrypts model updates, and participates in a secure aggregation process. The server learns only the final aggregated model without accessing individual contributions.

Zero-Knowledge Proofs (ZKPs) are increasingly explored to validate correctness of model updates without revealing sensitive data. These are especially relevant in federated settings where adversarial clients may attempt model poisoning or submission of malformed updates.

Hybrid approaches combining DP, HE, and SMPC are being proposed to achieve practical security under real-world constraints. These mechanisms trade off between computational overhead, communication cost, and privacy guarantees [26].

Auditable privacy logs and privacy budget accounting mechanisms are embedded into modern federated frameworks to monitor and enforce usage limits, particularly under GDPR or HIPAA constraints.

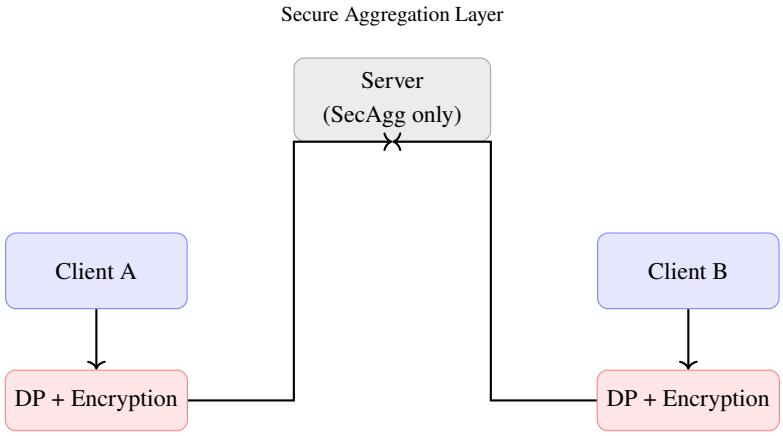

Figure 4: Privacy-preserving federated learning pipeline incorporating multiple protection layers. Each client applies local differential privacy by adding calibrated noise to gradients, then encrypts the noisy updates. The server performs secure aggregation using cryptographic protocols (e.g., SecAgg) that reveal only the final aggregated model while individual client contributions remain private.

6. Deployment and Orchestration of Federated Systems

Deploying federated learning (FL) systems in real-world scenarios requires scalable, resilient, and reproducible infrastructure. The modular nature of FL pipelines makes them ideal candidates for cloud-native design principles based on microservices and containerization [27].

Containerized components encapsulate federated clients, the orchestrator, and auxiliary services. Each unit is packaged with its dependencies, ensuring consistency across heterogeneous hardware. Docker is widely used for packaging FL components, enabling rapid provisioning across cloud or edge devices [3].

Orchestration is handled using Kubernetes (K8s), which offers fault-tolerant scheduling, self-healing, and horizontal scaling. Custom resource definitions (CRDs) are employed to model FL jobs, enabling declarative deployment via YAML specifications. Kubernetes-native tools like Helm or Kustomize manage complex configuration across clusters [28].

Edge-aware scheduling mechanisms are integrated into orchestration frameworks to optimize data locality and resource allocation. These schedulers can dynamically assign jobs to on-premise GPUs, cloud instances, or mobile devices based on latency, availability, and capability constraints.

Communication between pods or nodes is secured using mutual TLS authentication, and network policies restrict inter-service communication to only authorized paths. Secrets and credentials are stored in Kubernetes secrets or encrypted volumes to ensure runtime isolation.

Monitoring and observability are critical for production-grade deployments. Tools such as Prometheus, Grafana, and Fluentd provide real-time metrics, logs, and alerts. These help track job progression, resource usage, and error diagnostics across nodes [29].

A typical deployment pipeline includes continuous integration/continuous delivery (CI/CD) to build, test, and deploy updates. GitOps workflows are increasingly adopted for versioned and auditable infrastructure as code (IaC) practices. Platform engineers can push changes to version control systems, triggering automated deployments and rollbacks.

Figure 5 illustrates the cloud-native deployment of a federated system. Clients and server components are deployed as pods in a Kubernetes cluster, connected through secure channels, with monitoring and scaling mechanisms in place.

Auto-scaling policies adjust the number of client pods based on workload, while node-affinity rules ensure deployment near relevant datasets. Stateful components like model stores or job metadata repositories persist in external volumes or databases.

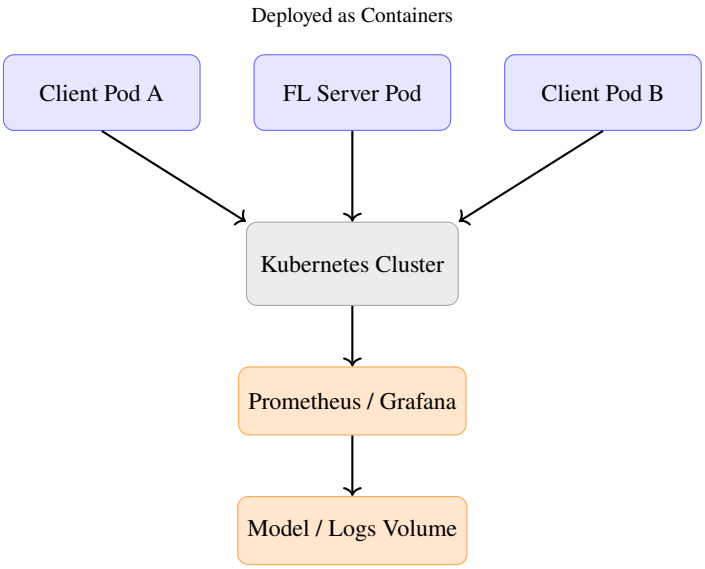

Figure 5: Cloud-native deployment architecture using Kubernetes orchestration. FL components are containerized and deployed as pods: aggregator server pods (center), client pods (left/right), and monitoring infrastructure (Prometheus/Grafana). Persistent volumes store model checkpoints and job metadata. Horizontal Pod Autoscalers dynamically adjust client replicas based on workload demands.

7. Conclusion

This paper has presented a structured reference architecture for cloud-native federated learning systems, synthesizing existing research into a cohesive framework that maps distributed learning components to containerized infrastructure primitives. The core architectural insight is the separation of concerns achieved through layered decomposition: control plane orchestration, secure communication channels, plugin-based execution engines, and privacy-preserving aggregation mechanisms. This abstraction enables federated systems to operate across heterogeneous infrastructure while maintaining security guarantees and statistical convergence properties.

Key findings from this synthesis include: (i) the critical role of containerization and Kubernetes orchestration in managing infrastructural heterogeneity, (ii) the necessity of combining multiple privacy techniques (differential privacy, secure aggregation, homomorphic encryption) to achieve practical confidentiality guarantees, and (iii) the importance of declarative job specifications for reproducible, version-controlled federated workflows. The integration of cloud-native DevOps practices—CI/CD pipelines, GitOps, and observability stacks—emerges as essential for production-grade deployments.

As federated learning matures toward widespread adoption, several research directions warrant continued investigation: resource-aware scheduling algorithms that optimize across edge constraints, formal verification of privacy guarantees in composite protocols, and standardized interoperability interfaces between FL frameworks. The convergence of federated learning with edge intelligence and real-time data pipelines will ultimately enable privacy-preserving AI applications across healthcare, finance, and smart infrastructure domains.

References

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *Proceedings of AISTATS*, 2017.

[2] Nicola Rieke, Jonathan Hancox, Wenqi Li, Fausto Milletari, Holger R Roth, Shadi Albarqouni, Spyridon Bakas, Manuel J Cardoso, Antonio Criminisi, et al. The future of digital health with federated learning. *npj Digital Medicine*, 3(1):1–7, 2020.

[3] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2):1–19, 2019.

[4] Peter Kairouz, H Brendan McMahan, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[5] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of MLSys*, 2020.

[6] Hardik Pancholi and Saptarshi Majumdar. Federated learning and cloud-native computing: synergies, challenges, and opportunities. In *IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, 2021.

[7] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vlad Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of MLSys*, 2019.

[8] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. *Proceedings of ICML*, 2020.

[9] Muhammad Adnan, Sumeet Kalra, and David McDonald. Federated learning with privacy and fairness in healthcare. *Artificial Intelligence in Medicine*, 2022.

[10] Samuel Liu, Wenqi Li, Dong Xu, et al. Nvflare: An open-source federated learning framework for secure, cross-silo training. In *Proceedings of NeurIPS FL Workshop*, 2022.

[11] Praneeth Vepakomma, Omid Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. In *Proceedings of NeurIPS Workshop on Privacy-Preserving ML*, 2018.

[12] Micah J Sheller, Brian Edwards, Gary A Reina, Jason Martin, and Spyridon Bakas. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. In *Scientific Reports*, volume 10, pages 1–12, 2020.

[13] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. In *NIPS Workshop on Machine Learning on the Phone and other Consumer Devices*, 2017.

[14] Stephen Hardy, Wilko Henecka, Jason Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677*, 2017.

[15] Yu Chen, Yuxuan Qin, and Xin Jin. Asynchronous online federated learning for edge devices with non-iid data. *IEEE Transactions on Network Science and Engineering*, 2020.

[16] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *MLSys*, 2018.

[17] Felix Sattler, Klaus-Robert Wiedemann, Klaus-Robert Muller, and Wojciech Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[18] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Brendan McMahan. Adaptive federated optimization. In *ICLR*, 2021.

[19] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC*, 2019.

[20] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *NeurIPS*, 2017.

[21] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. *IEEE S&P*, 2019.

[22] Martin Abadi, Andy Chu, Ian Goodfellow, et al. Deep learning with differential privacy. In *ACM CCS*, 2016.

[23] Keith Bonawitz, Vladimir Ivanov, Benjamin Kreuter, et al. Practical secure aggregation for privacy-preserving machine learning. In *ACM CCS*, 2017.

[24] Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shigeo Moriai. Privacy-preserving deep learning via additively homomorphic encryption. In *IEEE TDSC*, 2017.

[25] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *IEEE S&P*, 2017.

[26] Stacey Truex, Ling Liu, Kamalika Chow, et al. A hybrid approach to privacy-preserving federated learning. *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, 2019.

[27] Andreas Berl, Daniel Breitenbacher, et al. Deployment of federated learning systems using kubernetes and helm. *Journal of Grid Computing*, 2021.

[28] Yue Zhao, Meng Li, Liang Lai, Naveen Suda, David Civin, and Vikas Chandra. Federated learning with non-iid data via local structure preservation. In *ICML*, 2020.

[29] Fan Li, Esha Chou, Qixuan Ye, et al. Openfl: An open-source framework for federated learning. https://github.com/intel/openfl, 2020.