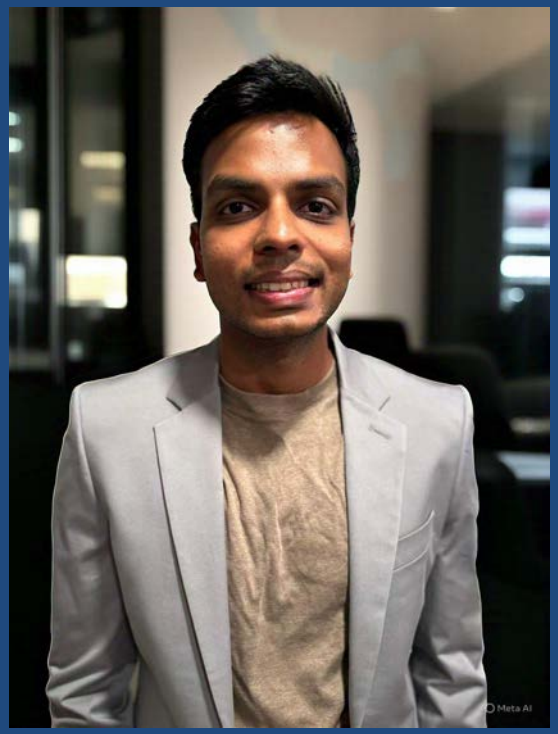
Observability in Privacy Aware Infrastructure

Presenter -

Krishna Ganeriwal
Senior Software Engineer,
Meta Platforms Inc.



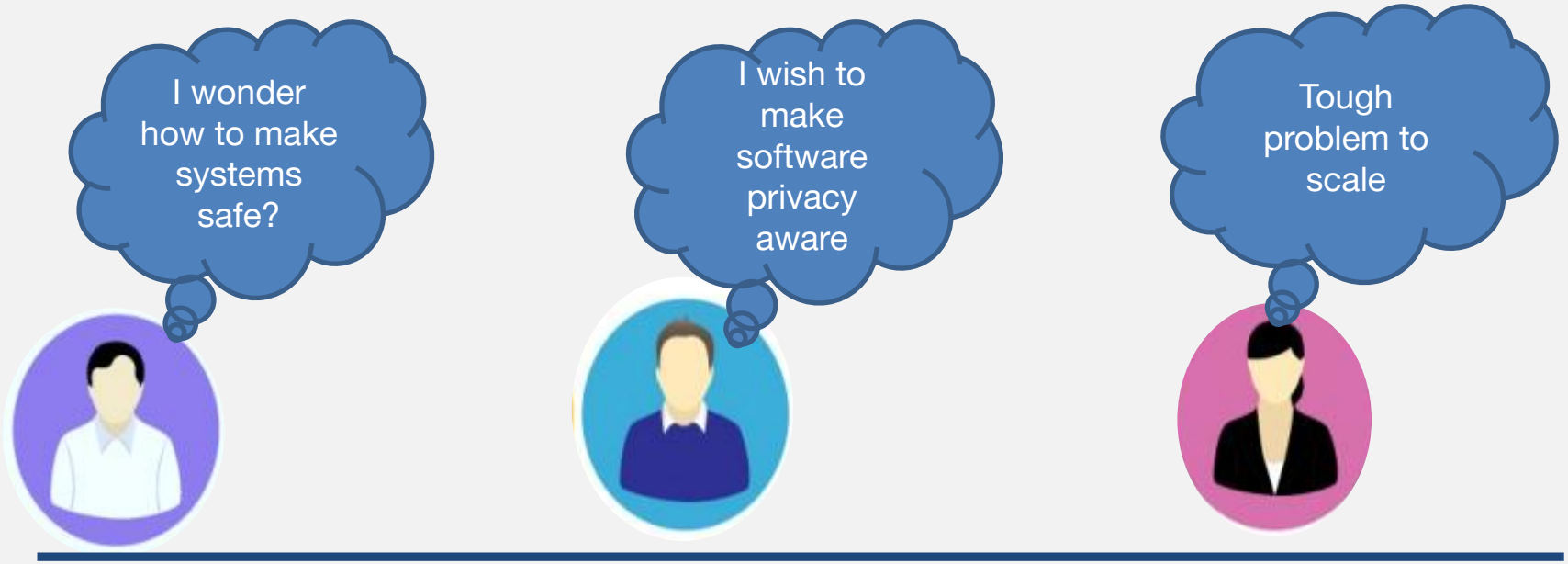
Enabling Trust and Compliance in Modern Software Systems

AGENDA

- What are Privacy Aware Systems?
- What is Observability?
- Why Privacy Aware Systems need Observability?
- Observability for Privacy - Key Use Cases
- Challenges
- Architectural Patterns
- Case Study
- Benefits
- Future Directions
- Conclusion

Privacy Aware Systems

- Privacy-aware systems are designed to ensure that user data is handled in a way that respects privacy regulations and user expectations.
- These systems are part of a broader initiative known as Privacy Aware Infrastructure, which aims to embed privacy deeply into the technical fabric of the company's infrastructure. Here are some key aspects of privacy-aware systems



Observability

Observability is a critical concept in software development and operations that enables teams to understand the internal state of a system by examining its external outputs. In other words, it's about being able to infer what's happening inside a system without shipping additional code.

- **Logs:** Events in the system (e.g., a user accesses data)
- **Metrics:** Numeric time-series data (e.g., how often users opt out of data-sharing)
- **Traces:** Request flow across systems (e.g., user click event traversing services)
- **Goal:** Understand system state from the outside (even without direct access to the internal code)

Why Privacy Infra Needs Observability?

- **Increasing global data regulations:** GDPR, CCPA, DPDP
- **Need for real-time visibility:** into how user data is accessed and shared.
- **Proactive approach:** to identifying violations rather than waiting for audits
- **Regulatory fines:** Companies face penalties up to €20 million or 4% of global turnover (GDPR) for non-compliance

KEY USE CASES

- DATA LINEAGE
- CONSENT PROPAGATION VERIFICATION
- PURPOSE LIMITATION ENFORCEMENT
- ANOMALY DETECTION
- POLICY DRIFT MONITORING

System Design Challenges

- **Volume:** Billions of events; petabytes of lineage data
- **Speed:** Sub-second consent revocation
- **Granularity:** Per-user traceability
- **Security:** Observability systems must not leak PII

Architectural Patterns

- **Event-driven Lineage:** Using tools like Kafka to track data flows in real-time.
- **Consent-aware Tagging:** Every piece of data carries consent status across services.
- **Embedded Policy Engines:** Enforcing privacy policies directly within the codebase.
- **Cryptographic Audit Trails:** Using Merkle trees to maintain verifiable and immutable audit trails.Billions of events; petabytes of lineage data

Case Study: Lineage at Scale

- 1000+ data sources
- Real-time data graphing
- Consent-aware data enforcement
- Automated policy compliance checks

Benefits of Privacy Observability

- 40% faster audits
- Reduced regulatory fine risk
- Increased internal trust
- Safer ML and analytics usage

FUTURE WORK

- Can ML detect privacy issues in traces?
- Observability as Policy Enforcement
- Cross-org Observability Standards

Conclusion

- Observability is critical to privacy enforcement
- Build accountable, auditable systems
- Must be integrated at infra layer

THANK YOU