Infrastruktur Big Data:

Membangun Pondasi untuk Era Informasi

Mengulas tentang teknologi dan arsitektur yang mendbkung pengeloiaan big data secara efektif

Oleh: Rudy C Tarumingkeng

Prof Ir Rudy C Tarumingkeng, PhD
Guru Besar Manajemen NUP: 9903252922
Rektor, Universitas Cenderawasih, Papua (1978-1988)
Rektor, Universitas Kristen Krida Wacana, Jakarta (1991-2000)
Ketua Dewan Guru Besar, IPB-University, Bogor (2005-2006)
Data Analyst, dan Ketua Senat Akademik, IBM-ASMI, Jakarta

Infrastruktur Big Data: Membangun Pondasi untuk Era Informasi

(Teknologi, arsitektur, dan praktik pengelolaan big data secara efektif—dengan narasi kasus dan konteks organisasi)

1) Mengapa "infrastruktur big data" menjadi isu strategis, bukan sekadar teknis

Di banyak organisasi modern—bank, telko, ritel, manufaktur, layanan publik, pendidikan, kesehatan—data tidak lagi hadir sebagai "lampiran laporan," melainkan sebagai **sumber daya inti** yang menggerakkan keputusan, inovasi, dan keunggulan bersaing. Namun big data memiliki karakter yang membuat pendekatan TI tradisional (sekadar menambah server atau membeli aplikasi pelaporan) sering gagal.

Secara konseptual, big data ditandai oleh kombinasi:

- **Volume** (ukuran data besar),
- **Velocity** (kecepatan data masuk/berubah),
- **Variety** (ragam format: tabel, teks, citra, audio, log, IoT),
- **Veracity** (kualitas/ketidakpastian),
- **Value** (nilai bisnis yang harus diekstraksi).

Di titik ini, "infrastruktur big data" berarti **pondasi end-to-end**: mulai dari cara data dikumpulkan, dipindahkan, disimpan, diproses, dianalisis, diamankan, diaudit, sampai disajikan untuk pemakai bisnis dan model AI. Infrastruktur bukan hanya perangkat, melainkan **arsitektur + proses operasi + tata kelola + kompetensi SDM**.

---

2) Prinsip dasar arsitektur big data: dari pipa data ke ekosistem data

Sebelum membahas teknologi, ada prinsip desain yang menentukan apakah sistem big data akan menjadi "mesin nilai" atau "kuburan data".

(a) Pisahkan *data plane* dan *control plane*

- **Data plane**: jalur teknis pemrosesan (ingestion, storage, compute).
- **Control plane**: hal-hal yang mengatur: metadata, katalog, kualitas, akses, lineage, kebijakan retensi, audit.

Tanpa control plane yang kuat, organisasi sering mengalami: "data ada, tapi tidak bisa dipercaya," atau "data banyak, tapi tidak bisa ditemukan."

(b) Skalabilitas dan elastisitas

Big data menuntut kemampuan meningkat/menurun sesuai beban. Dalam praktik modern, ini sering berarti **cloud/hybrid** atau **cluster terorkestrasi** (misal Kubernetes) sehingga komputasi dapat elastis.

(c) *Schema-on-write* vs *schema-on-read*

- **Data warehouse klasik** cenderung *schema-on-write*: data harus rapih dulu sebelum masuk.
- **Data lake** cenderung *schema-on-read*: data boleh "mentah," struktur ditentukan saat dibaca/diolah.

Tren saat ini mengarah ke kompromi yang lebih matang: **lakehouse**, agar fleksibilitas data lake bertemu disiplin warehouse.

(d) Dekatkan data ke kebutuhan nilai

Data yang bagus bukan data yang paling lengkap, tetapi data yang **terhubung dengan use case**: deteksi fraud, prediksi churn, optimasi stok, personalisasi layanan, monitoring kualitas, analisis kebijakan, dsb. Infrastruktur yang sukses biasanya dibangun **berdasarkan prioritas use case**, bukan "kumpulkan semua dulu."

---

3) Lapisan-lapisan infrastruktur big data (end-to-end blueprint)

Bayangkan infrastruktur big data sebagai "kota": ada jalan raya (pipeline), gudang (storage), pabrik (compute), kantor pemerintahan (governance), dan layanan publik (akses).

3.1 Sumber data (Data Sources Layer)

Sumber big data umumnya meliputi:

1. **Sistem transaksi**: ERP, CRM, Core Banking, POS, HRIS.
2. **Log & event**: clickstream aplikasi, log server, audit trail.
3. **IoT & sensor**: mesin pabrik, kendaraan, smart meter energi.
4. **Dokumen & teks**: email, tiket layanan, kontrak, berita internal.
5. **Data eksternal**: pasar, cuaca, demografi, vendor, media sosial (sesuai kepatuhan).

Tantangan utama: heterogenitas format, kepemilikan data lintas unit, dan kualitas data di hulu.

3.2 Ingestion & integration (Mengalirkan data)

Ini adalah "jalan raya" yang mengalirkan data dari sumber menuju platform data.

Dua pola besar ingestion:

1. **Batch ingestion**
   Cocok untuk data periodik (harian/mingguan): laporan transaksi, rekap stok, data master.
   - Alat umum: ETL/ELT tools, scheduler, orkestrasi workflow.
2. **Streaming / real-time ingestion**
   Cocok untuk data cepat: event aplikasi, transaksi kartu, telemetry IoT.
   - Konsep kunci: *message broker* dan *event streaming*.

Arsitektur praktis ingestion sering memadukan:

- **CDC (Change Data Capture)** untuk menangkap perubahan dari database transaksi tanpa membebani sistem inti.
- **Event bus** untuk menyatukan event dari banyak aplikasi.
- **API gateway** untuk integrasi layanan modern.

Kegagalan klasik ingestion biasanya bukan pada alatnya, tetapi pada:

- tidak ada standar definisi event,
- tidak ada kontrak data (data contract),
- tidak ada monitoring kualitas dan keterlambatan (data freshness).

3.3 Storage layer: Data lake, warehouse, lakehouse

Storage adalah "gudang" yang menentukan daya tahan, biaya, performa, dan fleksibilitas.

(a) Data warehouse

Kuat untuk analitik terstruktur dan pelaporan resmi: KPI, dashboard eksekutif, pelaporan regulator.

- Keunggulan: konsistensi, performa query, governance lebih mudah.
- Kelemahan: kurang fleksibel untuk data semi-terstruktur atau eksperimen data science.

(b) Data lake

Menampung data mentah (raw) dalam skala besar, sering di **object storage**.

- Keunggulan: fleksibel, biaya relatif efisien untuk skala besar.
- Kelemahan: raw lake tanpa governance → menjadi "data swamp" (raw tapi tak berguna).

(c) Lakehouse (tren penting)

Lakehouse berusaha menggabungkan:

- fleksibilitas data lake
- manajemen tabel, transaksi, dan performa ala warehouse

Dengan lakehouse, organisasi dapat menyimpan data besar di object storage tetapi tetap memiliki kemampuan: versioning, ACID-like transactions, indexing, dan governance yang lebih baik.

Prinsip penyimpanan yang sehat:

- **Zona data**: raw → cleaned → curated (gold) → feature store (untuk ML).
- **Format kolumnar** (misal Parquet/ORC) untuk efisiensi query analitik.
- **Partisi dan clustering** untuk percepatan query.
- **Retensi dan arsip**: data lama tidak selalu harus berada di storage mahal.

3.4 Compute & processing: mesin pengolah nilai

Jika storage adalah gudang, compute adalah "pabrik" yang mengubah bahan mentah menjadi produk bernilai.

Dua pola pemrosesan utama:

1. **Batch processing**
    - ETL/ELT, agregasi KPI, training model periodik.
2. **Stream processing**
    - deteksi fraud real-time, monitoring mesin, alerting operasional.

Pertimbangan desain compute:

- **Pemrosesan terdistribusi** untuk skala besar (parallelism).
- **Pemilihan engine** berdasarkan kebutuhan: SQL analytics vs data engineering vs ML.
- **Resource management**: siapa boleh memakai compute besar, kapan, dan dengan prioritas apa.

Masalah yang sering muncul:

- *cost blow-up* (biaya cloud melonjak) karena query tidak efisien, data tidak dipartisi, pipeline redundan.
- *pipeline sprawl*: terlalu banyak job tanpa standar, sulit dipelihara.

Solusinya bukan "ganti alat," melainkan **standarisasi pola pipeline**, observability, dan desain data model.

3.5 Query, serving, dan akses data (konsumsi)

Data hanya bernilai jika bisa dipakai.

Lapisan konsumsi mencakup:

- **BI & dashboard**: untuk manajemen dan unit bisnis.
- **Ad-hoc analytics**: eksplorasi cepat oleh analis.
- **Data API**: data disajikan sebagai layanan untuk aplikasi operasional (misal personalisasi).
- **Search & text analytics**: untuk dokumen, tiket layanan, knowledge base.
- **Feature store & model serving**: untuk AI/ML yang dipakai dalam proses bisnis.

Prinsip penting: **berbeda use case, berbeda "serving pattern."** Dashboard eksekutif butuh konsistensi dan definisi KPI tunggal; aplikasi rekomendasi butuh latensi rendah dan data event real-time.

3.6 Governance, security, dan compliance (fondasi kepercayaan)

Di era big data, tantangan utamanya bukan hanya "menyimpan dan menghitung," tetapi **kepercayaan**: siapa boleh akses apa, data mana yang sah, bagaimana audit dilakukan.

Komponen governance yang matang:

1. **Data catalog**: "peta perpustakaan data" (apa isinya, milik siapa, definisinya apa).
2. **Metadata & lineage**: jejak asal-usul data, transformasi, sampai output.
3. **Data quality management**: aturan validasi, monitoring anomali, SLA data.
4. **Access control**: RBAC/ABAC, least privilege.

5. **Privacy & masking**: pseudonymization, tokenization, redaction untuk data sensitif.
6. **Retention policy**: berapa lama data disimpan dan kapan dihapus/diarsip.

Tanpa governance, organisasi akan menghadapi:

- konflik definisi KPI (“angka penjualan kok beda?”),
- risiko kebocoran data,
- model AI bias karena data kotor,
- audit/regulator issue.

3.7 DataOps, MLOps, dan observability (operasi yang berkelanjutan)

Banyak program big data gagal setelah pilot karena tidak siap operasional.

DataOps menekankan:

- pipeline sebagai produk (versioning, testing, CI/CD),
- monitoring end-to-end,
- manajemen perubahan yang disiplin.

MLOps menambahkan:

- tracking eksperimen model,
- monitoring drift (data drift/model drift),
- deployment aman dan terukur,
- governance model (model risk management).

Observability: bukan hanya “server hidup,” tetapi:

- data masuk tepat waktu,
- kualitas data stabil,
- job tidak gagal diam-diam,
- dashboard tidak menampilkan data kadaluarsa.

---

4) Pola arsitektur populer: memilih "gaya bangunan" yang tepat

4.1 Lambda Architecture (batch + speed layer)

Menggabungkan batch (akurasi historis) dan speed layer (real-time).

- Kuat untuk kebutuhan real-time + histori.
- Tantangan: kompleksitas dua jalur (dua pipeline, dua logika).

4.2 Kappa Architecture (streaming sebagai pusat)

Berfokus pada streaming; batch sering dianggap replay stream.

- Lebih sederhana daripada lambda untuk organisasi yang "event-driven."
- Tantangan: butuh kematangan stream processing dan manajemen event.

4.3 Lakehouse Architecture

Sangat populer untuk mengurangi fragmentasi lake vs warehouse.

- Cocok jika organisasi ingin satu fondasi untuk BI + data science + AI.
- Tantangan: governance dan desain zona data harus disiplin.

4.4 Data Mesh (pendekatan organisasi)

Ini bukan alat, melainkan *operating model*: data sebagai produk domain, bukan milik satu tim pusat.

- Cocok untuk perusahaan besar multi-unit yang sering bottleneck di tim data pusat.
- Tantangan: butuh standar platform, kontrak data, dan budaya akuntabilitas domain.

4.5 Data Fabric (integrasi & metadata-driven)

Menekankan integrasi lintas sumber melalui metadata, automasi, dan konektivitas.

- Cocok untuk organisasi dengan landscape data yang sangat tersebar.
- Tantangan: implementasi rumit jika governance lemah.

---

5) Narasi kasus: membangun pondasi big data di organisasi (simulasi realistis)

Bayangkan sebuah organisasi ritel nasional dengan kanal toko fisik + e-commerce. Masalah utama: stok sering tidak sinkron, promosi kurang tepat sasaran, dan pelanggan loyal mulai pindah ke kompetitor.

Tahap 1: Menetapkan use case prioritas (bukan mulai dari alat)

Dewan direksi menetapkan 3 sasaran:

1. **Forecast permintaan & optimasi stok** per wilayah,
2. **Customer 360** (profil pelanggan lintas kanal),
3. **Deteksi anomali transaksi** (fraud/abuse promo).

Tahap 2: Mendesain arsitektur minimal yang bisa tumbuh

- Ingestion batch dari ERP/POS harian + CDC untuk transaksi penting.

- Streaming event dari aplikasi e-commerce (clickstream, cart, payment events).
- Storage dengan zona raw → cleaned → curated.
- Compute: batch untuk forecast harian, streaming untuk alert anomali.
- Catalog + definisi KPI tunggal: "penjualan bersih" didefinisikan jelas.

Tahap 3: Menjadikan governance sebagai akselerator, bukan penghambat

Alih-alih melarang semua akses, organisasi menerapkan:

- klasifikasi data (publik/internal/rahasia),
- masking untuk data sensitif,
- role-based access dengan audit trail.

Hasil 6–9 bulan:

- akurasi forecast membaik (stok lebih efisien),
- promosi lebih tepat (berdasarkan segmentasi perilaku),
- fraud promo turun karena alert real-time.

Catatan penting: keberhasilan bukan karena "platform mahal," tetapi karena **keterpaduan arsitektur + disiplin data + ownership lintas unit.**

---

6) Komponen teknologi: peta fungsi (agar tidak "terjebak merek")

Daripada menghafal nama produk, lebih sehat memahami **fungsi**. Satu fungsi bisa dipenuhi oleh berbagai teknologi.

1. **Ingestion/streaming**: message broker, event streaming, CDC

2. **Orkestrasi**: workflow scheduling, dependency management
3. **Storage**: object storage, distributed file system, warehouse
4. **Processing**: distributed compute (batch/stream), SQL engine
5. **Serving**: OLAP, search, API layer, caching
6. **Governance**: catalog, lineage, policy enforcement
7. **Security**: IAM, encryption, key management, secrets vault
8. **Observability**: logging, metrics, tracing, data quality monitors

Cara berpikir ini mencegah organisasi membeli alat "karena tren," tetapi tidak terpakai karena fungsi dan prosesnya tidak siap.

---

7) Tantangan desain yang paling sering menjatuhkan proyek big data

(1) "Kita bikin data lake dulu, use case nanti"

Akhirnya jadi data swamp. Solusi: **use case-driven roadmap** + zona data + quality gate.

(2) Data kualitas buruk di hulu

Big data memperbesar masalah kecil menjadi besar. Solusi: data contracts, validasi, master data management, dan perbaikan proses sumber.

(3) Fragmentasi platform

Banyak tool yang tidak terintegrasi → biaya tinggi, governance sulit. Solusi: platform reference architecture + standar.

(4) Biaya tidak terkendali

Cloud memudahkan eksperimen, tetapi tanpa guardrail biaya membengkak. Solusi: tagging biaya, quota, optimasi partisi, lifecycle storage.

(5) Kekurangan talenta dan kesenjangan budaya

Data engineering, analytics, governance, security, dan domain knowledge harus bersatu. Solusi: *cross-functional data product teams*.

---

8) Roadmap implementasi: dari pondasi ke skala organisasi

Berikut tahapan yang lazim dan realistis:

Tahap A — Fondasi (0–3 bulan)

- prioritas 2–3 use case,
- rancang arsitektur target + standar data,
- bangun ingestion minimal + storage zona raw/cleaned,
- siapkan katalog awal + akses dasar.

Tahap B — Produk data pertama (3–6 bulan)

- pipeline stabil untuk use case prioritas,
- dashboard/KPI yang disepakati,
- data quality monitoring,
- SOP operasi dan incident management.

Tahap C — Skalasi (6–12 bulan)

- tambah domain data,
- mulai data mesh/data product,
- tambah streaming untuk kasus real-time,
- integrasi MLOps untuk model yang masuk produksi.

Tahap D — Optimasi & inovasi (12 bulan ke atas)

- otomasi governance lebih kuat,
- lakehouse maturity,

- feature store, real-time personalization,
- continuous improvement + cost optimization.

---

9) Dimensi manajerial: infrastruktur big data sebagai kapabilitas organisasi

Agar big data tidak berhenti sebagai proyek TI, organisasi perlu melihatnya sebagai **kapabilitas** yang memiliki empat pilar:

1. **Strategy**: use case, prioritas, nilai bisnis, KPI keberhasilan.
2. **Structure**: peran (data owner, steward, engineer, analyst), model organisasi (centralized vs federated).
3. **Systems**: platform, standar, governance, operasi.
4. **Skills & culture**: literasi data, cara mengambil keputusan berbasis bukti, etika.

Kuncinya: infrastruktur big data yang matang akan memampukan organisasi bergerak dari:
Data → Informasi → Insight → Keputusan → Aksi → Pembelajaran organisasi.

---

10) Penutup: "pondasi" yang menentukan masa depan analitik dan AI

Era AI membuat infrastruktur big data semakin krusial. Model AI yang canggih tidak akan menyelamatkan organisasi bila: data tidak berkualitas, lineage tidak jelas, akses tidak aman, atau pipeline rapuh. Sebaliknya, organisasi yang membangun fondasi big data dengan arsitektur yang tepat, governance yang kuat, serta operasi DataOps/MLOps yang disiplin akan memiliki "mesin pembelajaran" yang terus meningkatkan kualitas keputusan dan layanan.

Refleksi dan Diskusi

Membangun **infrastruktur big data** pada dasarnya adalah proses *mendirikan "pabrik informasi"*—bukan sekadar menginstal Hadoop/Spark atau menambah kapasitas storage. Di dalam "pabrik" ini, data mentah (log transaksi, sensor IoT, rekam layanan, klik pengguna, dokumen, citra, audio) harus **masuk** (ingestion), **diatur** (governance & metadata), **dibersihkan dan diolah** (transformasi), **disajikan** (serving/BI/AI), serta **diawasi kualitas dan risikonya** (observability, security, privacy). Ketika salah satu bagian rapuh, organisasi akan merasakan gejalanya: dashboard tidak konsisten, model AI cepat "melenceng", biaya cloud membengkak, atau—yang paling serius—insiden kebocoran data dan krisis kepercayaan publik.

Secara manajerial, ada beberapa "ketegangan desain" yang selalu muncul:

1. **Skala vs keterkendalian (control)**
   Big data menuntut sistem yang elastis dan terdistribusi; namun semakin terdistribusi, semakin sulit menjaga standar definisi data, kualitas, dan tanggung jawab. Inilah alasan mengapa kerangka tata kelola data (mis. DAMA-DMBOK) menempatkan *data governance* sebagai pusat—agar pertumbuhan data tidak berubah menjadi "hutan belantara" yang tidak bisa dikelola. (dama.org)
2. **Kecepatan inovasi vs kepatuhan regulasi**
   Tim produk ingin eksperimen cepat (A/B testing, personalisasi,

rekomendasi), sedangkan regulasi perlindungan data pribadi menuntut prinsip-prinsip pemrosesan yang sah, aman, dan akuntabel. Di Indonesia, kebutuhan ini menguat karena adanya **UU No. 27 Tahun 2022 tentang Pelindungan Data Pribadi** yang menegaskan kewajiban pengendali/pemroses data serta perlindungan hak subjek data. (JDIH Kemenko Marves)

3. **Konsistensi "single source of truth" vs realitas multi-sumber**
   Dalam praktik, organisasi jarang punya satu sumber data yang rapi. Yang ada adalah banyak sistem: ERP, CRM, aplikasi cabang, log aplikasi, data vendor, media sosial, dan sebagainya. Karena itu, infrastruktur modern menekankan **metadata, katalog data, lineage**, serta mekanisme integrasi batch/streaming untuk menjaga makna data tetap konsisten saat berpindah lintas sistem.

4. **Sentralisasi vs desentralisasi (data mesh)**
   Pendekatan sentralistik (satu tim platform mengatur semua) sering tersendat karena bottleneck. Pendekatan **data mesh** mendorong domain bisnis menjadi "pemilik produk data"—namun membutuhkan disiplin standar, interoperabilitas, dan tata kelola yang matang agar tidak kembali menjadi silo. (Google Books)

Mini-kasus (narasi ringkas)

Bayangkan sebuah pemerintah daerah membangun *data lake* untuk menyatukan data bansos, kesehatan, pendidikan, dan kependudukan. Pada tahun pertama, proyek tampak sukses: laporan terpadu dan peta penerima bantuan terbentuk. Namun memasuki tahun kedua, masalah muncul: definisi "keluarga miskin" berbeda antar dinas, data ganda meningkat, dan audit menemukan akses data terlalu longgar. Solusinya bukan hanya menambah server, melainkan membangun **lapisan tata kelola**: katalog data, standar definisi, kontrol akses berbasis peran, serta proses DataOps untuk kualitas dan rilis pipeline yang disiplin. Di sinilah infrastruktur big data menunjukkan hakikatnya: *arsitektur teknis + tata kelola + proses operasi*.

Pertanyaan diskusi (untuk kelas/FGD)

1. Apa indikator bahwa organisasi Anda butuh *lakehouse* ketimbang sekadar data lake atau data warehouse? (cidrdb.org)
2. Bagaimana membedakan masalah "teknologi kurang" vs "tata kelola lemah"? (dama.org)
3. Siapa yang seharusnya menjadi *data owner* untuk data lintas domain (mis. pelanggan) agar akuntabilitas jelas?
4. Jika tim ingin *real-time analytics*, risiko operasional apa yang meningkat (biaya, latensi, konsistensi, keamanan)? (Google Books)
5. Bagaimana menerapkan prinsip privacy-by-design di pipeline data? (NIST)
6. Apa konsekuensi memilih vendor stack tertutup vs format terbuka (mis. Iceberg/Parquet)? (Apache Iceberg)
7. Bagaimana memastikan kualitas data sebagai "produk" yang punya SLA/SLO? (dataopsmanifesto.org)
8. Risiko "technical debt" seperti apa yang khas pada sistem ML berbasis big data? (Google Research)
9. Kapan strategi data mesh cocok, dan kapan justru memperburuk fragmentasi? (Google Books)
10. Jika terjadi kebocoran data, bagian arsitektur mana yang biasanya menjadi titik lemah (IAM, enkripsi, monitoring, governance)?

Glosarium Istilah Kunci

- **ACID**: Sifat transaksi (Atomicity, Consistency, Isolation, Durability) yang menjamin konsistensi saat baca/tulis data.

- **API (Application Programming Interface)**: Antarmuka standar agar sistem saling berkomunikasi dan bertukar data.
- **Batch Processing**: Pemrosesan data berkala dalam "paket" (mis. harian/mingguan).
- **CDC (Change Data Capture)**: Teknik menangkap perubahan data dari sistem sumber (insert/update/delete) untuk replikasi/streaming.
- **Columnar Storage**: Format penyimpanan kolom (mis. Parquet/ORC) yang efisien untuk analitik.
- **Data Catalog**: "Kamus + inventaris" aset data (tabel, file, dashboard, model) beserta metadata dan pemiliknya.
- **Data Drift**: Perubahan pola distribusi data input dari waktu ke waktu yang dapat menurunkan performa analitik/AI.
- **Data Fabric**: Pendekatan arsitektur yang menekankan integrasi dan akses data lintas platform melalui metadata, otomasi, dan layanan terpadu.
- **Data Governance**: Struktur kebijakan, peran, standar, dan proses untuk memastikan data dikelola aman, berkualitas, patuh, dan bernilai. (dama.org)
- **Data Lake**: Repositori terpusat (sering berbasis object storage) untuk menyimpan data mentah hingga terolah dalam skala besar.
- **Data Lakehouse**: Pola arsitektur yang menggabungkan kelebihan data lake (murah/fleksibel) dan data warehouse (kinerja/fitur BI/ACID) pada format terbuka. (cidrdb.org)
- **Data Lineage**: Jejak asal-usul data: dari sumber, transformasi, hingga konsumsi (dashboard/model).
- **Data Mart**: Subset data warehouse yang fokus pada satu domain (mis. keuangan/marketing).

- **Data Mesh**: Paradigma sosio-teknis yang mendesentralisasi kepemilikan data ke domain, dengan data diperlakukan sebagai "produk". (Google Books)
- **DataOps**: Praktik operasional (mirip DevOps) untuk mempercepat dan menstabilkan pipeline data melalui otomasi, testing, monitoring, dan kolaborasi. (dataopsmanifesto.org)
- **Data Owner**: Penanggung jawab bisnis atas definisi, kualitas, dan penggunaan data tertentu.
- **Data Steward**: Peran operasional yang menjaga standar, kualitas, dan kepatuhan data sehari-hari.
- **Data Warehouse**: Sistem penyimpanan terstruktur untuk pelaporan dan analitik dengan skema yang dikurasi.
- **ELT**: Pola Extract–Load–Transform; data dimuat dulu ke platform lalu ditransformasi di dalamnya.
- **Encryption at Rest / in Transit**: Enkripsi saat data tersimpan / saat data ditransmisikan melalui jaringan.
- **Event Streaming**: Pola pengaliran kejadian (event) secara kontinu untuk pemrosesan near real-time. (Apache Kafka)
- **ETL**: Pola Extract–Transform–Load; transformasi dilakukan sebelum masuk storage analitik.
- **Feature Store**: Komponen MLOps untuk menyimpan, mengelola, dan menyajikan fitur ML secara konsisten untuk training dan inference.
- **IAM (Identity and Access Management)**: Sistem pengelolaan identitas dan hak akses pengguna/layanan.
- **Iceberg (Open Table Format)**: Format tabel terbuka untuk data lake berskala besar yang mendukung evolusi skema, snapshot, dan komit transaksi. (Apache Iceberg)

- **Ingestion**: Proses memasukkan data dari sumber ke platform (batch maupun streaming).
- **Kubernetes**: Platform orkestrasi kontainer untuk deployment, scaling, dan otomasi layanan data/AI. (Kubernetes)
- **Lakehouse Paper**: Literatur yang memformalkan pola lakehouse sebagai generasi baru platform analitik terpadu. (cidrdb.org)
- **Metadata**: "Data tentang data" (skema, pemilik, waktu, kualitas, klasifikasi sensitif, dsb.).
- **MLOps**: Praktik untuk mengelola siklus hidup model ML (training, deployment, monitoring, retraining) secara andal.
- **Observability (Data Observability)**: Kemampuan memantau kesehatan pipeline dan kualitas data (freshness, volume, skema, anomali).
- **OLAP**: Pemrosesan analitik (agregasi, tren, slice-dice) untuk BI.
- **OLTP**: Pemrosesan transaksi operasional (insert/update cepat) untuk aplikasi harian.
- **Orchestration**: Pengaturan eksekusi pipeline (jadwal, dependensi, retry, alert).
- **Partitioning**: Teknik membagi data (mis. per tanggal/region) agar query lebih cepat dan biaya lebih efisien.
- **Pseudonymization/Anonymization**: Teknik mengurangi keterkaitan data dengan identitas individu untuk privasi (dengan tingkat yang berbeda).
- **RBAC/ABAC**: Kontrol akses berbasis peran / berbasis atribut (konteks, label data, dsb.).
- **Schema-on-Read / Schema-on-Write**: Skema diterapkan saat membaca (fleksibel) / saat menulis (lebih terkendali).

- **SLA/SLO Data**: Janji layanan untuk data (mis. keterkinian, kelengkapan, akurasi, latensi).
- **Spark**: Mesin analitik terpadu untuk pemrosesan data skala besar (batch dan streaming). (Apache Spark)
- **Streaming Processing**: Pemrosesan data yang terus mengalir dengan latensi rendah. (Google Books)
- **UU PDP**: Kerangka hukum Indonesia untuk perlindungan data pribadi, termasuk kewajiban pengendali/pemroses dan perlindungan hak subjek data. (JDIH Kemenko Marves)

Referensi

A. Standar, Kerangka, dan Regulasi

1. **NIST Big Data Interoperability Framework (NBDIF), Volume 1: Definitions (v2)**. National Institute of Standards and Technology. (NIST Publications)
2. **ISO/IEC 20546:2019 – Information technology: Big data—Overview and vocabulary**. International Organization for Standardization/IEC. (SIS)
3. **ISO/IEC 27001:2022 – Information security management systems (ISMS)**. ISO. (ISO)
4. **NIST Privacy Framework (PF)**. NIST. (NIST)
5. **NIST SP 800-53 Rev. 5 – Security and Privacy Controls for Information Systems and Organizations**. NIST CSRC. (NIST Computer Security Resource Center)
6. **DAMA-DMBOK (Data Management Body of Knowledge)**. DAMA International. (dama.org)

7. **Undang-Undang Republik Indonesia Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi** (UU PDP). (JDIH Kemenko Marves)

B. Buku Fondasional (Arsitektur Data & Analitik)

1. Dehghani, Z. (2022). **Data Mesh: Delivering Data-Driven Value at Scale**. O'Reilly. (Google Books)
2. Kleppmann, M. (2017). **Designing Data-Intensive Applications**. O'Reilly. (Martin Kleppmann)
3. Reis, J., & Housley, M. (2022). **Fundamentals of Data Engineering: Plan and Build Robust Data Systems**. O'Reilly. (Google Books)
4. Akidau, T., Chernyak, S., & Lax, R. (2018). **Streaming Systems: The What, Where, When, and How of Large-Scale Data Processing**. O'Reilly. (Google Books)
5. Kimball, R., & Ross, M. (2013). **The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling (3rd ed.)**. Wiley. (Wiley)
6. Inmon, W. H. (2005). **Building the Data Warehouse (4th ed.)**. Wiley. (Wiley)

C. Paper Riset dan Referensi Akademik

1. Armbrust, M., et al. (2021). **Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics** (CIDR 2021). (cidrdb.org)
2. Sculley, D., et al. (2015). **Hidden Technical Debt in Machine Learning Systems** (NeurIPS). (Google Research)

D. Dokumentasi Teknologi Kunci (Open Source)

1. **Apache Spark Documentation** (overview). (Apache Spark)
2. **Apache Kafka Documentation** (introduction). (Apache Kafka)

3. **Kubernetes Documentation** (overview). (Kubernetes)
4. **Apache Iceberg Documentation** (open table format). (Apache Iceberg)

Copilot for this article - Chatgpt 5.2 Thinking, Access date: 19 December 2025. Prompting on Writer's account (Rudy C Tarumingkeng)
https://chatgpt.com/c/69451e01-9430-8324-ba59-6c952fb562cd