Pertemuan 15 — *Data Governance*, GRC, dan *Observability*

Data yang dapat dipertanggungjawabkan, dipantau, dan terjangkau, serta persiapan UAS

Hendri Karisma

Semester Ganjil 2026/2027

Program Studi Teknik Informatika, STMIK Tazkia

Alur pertemuan hari ini

1. *Data governance* dan DAMA-DMBOK
2. *Catalog*, *classification*, *lineage*, *data contract*
3. GRC dan *risk register*
4. UU PDP, GDPR, ISO/IEC 27001 dan 27701
5. *Retention* vs *right to erasure*
6. *Data observability*: lima pilar
7. *Incident*, *runbook*, SLO, *recovery*
8. FinOps dan energi
9. Satu semester dalam satu peta
10. Praktikum gabungan
11. Presentasi progres tugas besar UAS

Sasaran pertemuan ini

Untuk *pipeline* Spark tugas besar Anda: tunjukkan **siapa pemilik** tiap tabel, **dari mana** angka di *dashboard* berasal, **kewajiban apa** yang dipenuhi, dan **bagaimana Anda tahu** bila datanya salah atau terlambat.

Data governance dan DAMA-DMBOK

Dua hal yang sering tertukar

Data governance

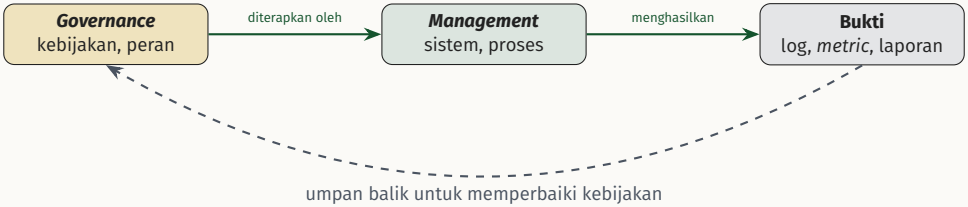
Mengatur: menetapkan kebijakan, peran, hak keputusan, dan akuntabilitas atas data. Menjawab “siapa berhak memutuskan apa” dan “bagaimana kita tahu aturan dipatuhi”.

Contoh: “Definisi resmi *nasabah aktif* ditetapkan Divisi *Retail* dan disahkan komite data.”

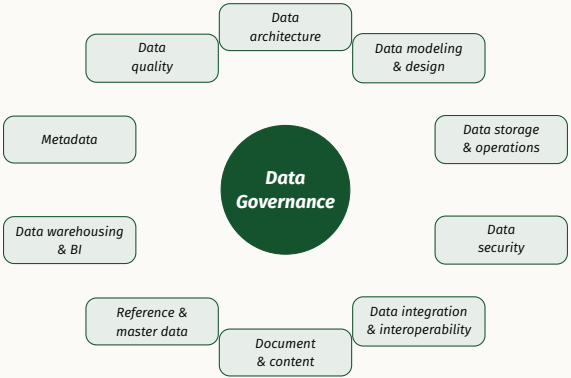
Data management

Mengerjakan: merancang, membangun, dan mengoperasikan sistem yang menjalankan kebijakan itu, *pipeline*, katalog, *data quality test*, kontrol akses.

Contoh: model dbt `gold.nasabah_aktif` mengikuti definisi itu, diuji setiap hari, dan tercatat di katalog.



Roda DAMA: *governance* di tengah



DAMA-DMBOK (*Data Management Body of Knowledge*) membagi manajemen data menjadi sebelas bidang pengetahuan, dengan **data *governance*** di pusat yang mengikat sepuluh lainnya.

- Bidang yang sudah kita bangun: *data modeling* (P2), *storage* dan operasi (P3, P7), integrasi (P6, P11), *lakehouse* dan BI (P4, P13), *security* (P14).
- Hari ini: ***governance***, **metadata**, ***data quality***, dan ***master data***.

Sumber: DAMA International, DAMA-DMBOK, edisi ke-2.

Kepemilikan dan *stewardship*

Peran	Tanggung jawab	Contoh di bank syariah digital
Data *owner*	Akuntabel atas *dataset*: siapa boleh akses, berapa lama disimpan, definisi resminya	Kepala Divisi Pembiayaan untuk data akad murabahah
Data *steward*	Menjaga *data quality* dan definisi sehari-hari, menjawab pertanyaan pengguna	Analis senior pembiayaan
Data *custodian*	Menjalankan penyimpanan, *backup*, dan kontrol teknis sesuai kebijakan	Tim *data platform* (kita)
Data *consumer*	Memakai data sesuai tujuan yang diizinkan	Tim risiko, *data scientist*
DPO	Mengawasi kepatuhan pelindungan data pribadi	Pejabat pelindungan data pribadi bank

Jebakan klasik

"Data milik tim IT." Tim *platform* adalah ***custodian***, bukan ***owner***. Keputusan bisnis atas data (definisi, siapa boleh melihat) harus dipegang pemilik bisnisnya.

Catalog, classification, lineage, data contract

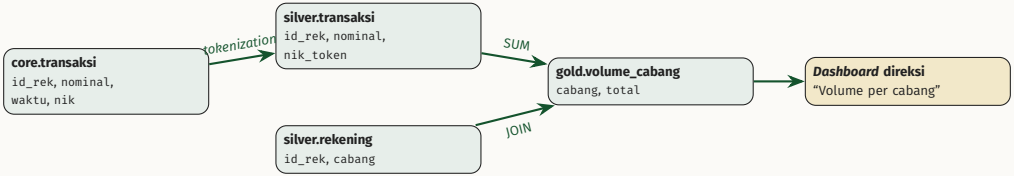
Data classification

Kelas	Contoh	Kontrol minimum
Publik	Daftar produk, lokasi cabang	Integritas
Internal	Agregat transaksi per cabang, *metric* operasional	Akses pegawai, SSO
Rahasia	Strategi harga margin, model *fraud*, laporan internal regulator	RBAC ketat, *encryption*, audit log
Data pribadi umum	Nama, jenis kelamin, kewarganegaraan	Dasar pemrosesan, *masking*, tujuan terbatas
Data pribadi spesifik	Data keuangan pribadi, biometrik, kesehatan, data anak	*Tokenization*, akses sangat terbatas, penilaian dampak

- UU PDP membedakan data pribadi **umum** dan **spesifik**; data keuangan pribadi termasuk spesifik, artinya hampir seluruh data inti bank masuk kategori ini.
- *Classification* ditempel sebagai *tag* di katalog, lalu **dipakai mesin**: kebijakan ABAC P14 membaca *tag* `PII.Spesifik` untuk menentukan *masking*.

Sumber: UU No. 27 Tahun 2022 tentang Pelindungan Data Pribadi (jenis data pribadi).

Lineage: dari mana angka ini berasal?



- ***Lineage* tingkat tabel**: tabel mana bergantung pada tabel mana.
- ***Lineage* tingkat kolom**: `total` berasal dari `nominal`; `nik` **tidak** mengalir ke gold.
- Dipakai untuk: analisis dampak sebelum mengubah *schema*, menelusuri angka salah, membuktikan PII tidak bocor ke *layer* yang salah.

OpenLineage

Standar terbuka untuk mengirim *event lineage* (*run*, *job*, *dataset*) dari Airflow, Spark, dbt, dan Flink ke penerima seperti Marquez, OpenMetadata, atau DataHub.

Data contract: perjanjian antara produsen dan konsumen

Masalah klasik: tim aplikasi mengganti nama kolom `nominal` menjadi `amount`; *pipeline silver* gagal pukul 02.00; *dashboard* direksi kosong pagi harinya.

Isi sebuah *data contract*

- *Schema* dan arti setiap kolom
- Jaminan *data quality* (*not null*, rentang nilai, keunikan)
- SLA: *freshness*, *availability*
- *Classification* dan kolom PII
- Pemilik dan jalur eskalasi
- Aturan perubahan: versi, masa transisi

Diterapkan di mana

- *Schema registry* Kafka dengan aturan kompatibilitas (P11)
- Uji dbt dan Great Expectations di batas *bronze* → *silver* (P6)
- CI produsen: perubahan yang melanggar kontrak ditolak sebelum *deploy*

Ringkasan

Kontrak memindahkan deteksi kerusakan dari ***downstream* dan terlambat** ke ***upstream* dan sebelum terjadi**.

Contoh *data contract* (YAML)

```yaml
kontrak: transaksi_rekening
versi: 1.2.0
pemilik: divisi-dana@bank.example
steward: analis-dana@bank.example
klasifikasi: PII.Spesifik
sla:
  kesegaran_maks: 5m
  ketersediaan: 99.5%
skema:
  - nama: id_transaksi
    tipe: string
    unik: true
  - nama: nik
    tipe: string
    pii: true          # wajib ditokenisasi di silver
  - nama: nominal
    tipe: decimal(18,2)
    aturan: "nominal > 0"
  - nama: waktu
    tipe: timestamp
perubahan: "kompatibel-mundur; hapus kolom = versi mayor"
```

- Format ini ilustrasi; ada spesifikasi terbuka seperti *Open Data Contract Standard* (ODCS) yang dapat diadopsi.
- Dari *file* ini dapat dibangkitkan otomatis: *schema* Avro untuk Kafka, uji dbt, dan *classification tag* di katalog.
- Kontrak disimpan di Git bersama kode produsen dan di-*review* kedua pihak.
- Angka SLA di contoh adalah ilustrasi, ditetapkan bersama pemilik bisnis.

GRC dan *risk management*

GRC: *Governance, Risk, Compliance*

Governance

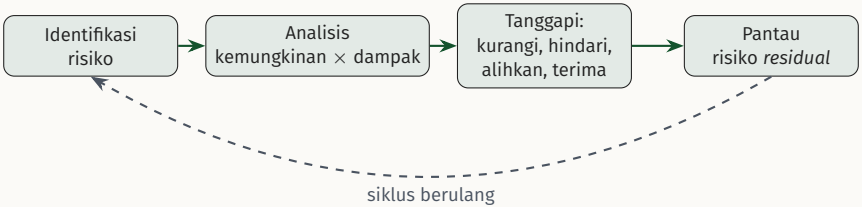
Struktur, peran, dan kebijakan: komite data, pemilik data, DPO, kebijakan *classification* dan retensi.

Risk

Mengidentifikasi apa yang bisa salah, menilai kemungkinan dan dampaknya, memilih kontrol, memantau risiko yang tersisa.

Compliance

Memenuhi kewajiban eksternal (UU PDP, regulator) dan internal (kebijakan, standar ISO), serta **membuktikannya**.



Matriks risiko: kemungkinan × dampak



Skala dan ambang warna adalah contoh; setiap organisasi menetapkan sendiri dalam kebijakan manajemen risikonya.

- **Risiko inheren**: sebelum ada kontrol.
- **Kontrol**: tindakan yang menurunkan kemungkinan atau dampak (*tokenization* menurunkan dampak kebocoran; RBAC menurunkan kemungkinannya).
- **Risiko *residual***: yang tersisa setelah kontrol.
- ***Risk appetite***: tingkat risiko yang bersedia diterima manajemen. Risiko *residual* di atas *appetite* wajib ditangani lebih lanjut atau diterima secara resmi oleh pejabat yang berwenang.

Contoh *risk register* platform data

ID	Risiko	K	D	Inh.	Kontrol	Res.	Pemilik
R1	Data nasabah bocor lewat *bucket* yang terbuka	3	5	15	*Block public access*, pemindaian *configuration*, *tokenization* PII	5	*Platform lead*
R2	Kredensial *pipeline* bocor di repositori	4	4	16	Vault, *secret scanning* di CI, rotasi otomatis	4	*Platform lead*
R3	Laporan regulator salah karena data terlambat atau ganda	3	4	12	*Data quality test*, *idempotency*, rekonsiliasi dengan sumber	4	*Data owner* keuangan
R4	Permintaan hapus nasabah tidak dapat dipenuhi di arsip	3	3	9	*Crypto-shredding*, katalog lokasi data pribadi	3	DPO
R5	Pegawai mengekspor data massal sebelum keluar	2	5	10	*Masking* dinamis, batas ekspor, deteksi anomali di audit log	4	Keamanan informasi

K = kemungkinan, D = dampak, Inh. = skor inheren (K $\times$ D), Res. = skor *residual* setelah kontrol. Semua angka ilustrasi untuk latihan; penilaian sebenarnya dilakukan bersama pemilik risiko.

UU PDP, GDPR, dan ISO

UU No. 27 Tahun 2022: peran dan dasar pemrosesan

Siapa siapa

- **Subjek data pribadi**: nasabah.
- ***Data controller*** (pengendali data pribadi): pihak yang menentukan tujuan dan cara pemrosesan, bank.
- ***Data processor*** (prosesor data pribadi): pihak yang memproses atas nama pengendali, misalnya penyedia *cloud* atau *vendor* analitik.

Berlaku penuh sejak berakhirnya masa transisi dua tahun pada Oktober 2024.

Dasar pemrosesan (Pasal 20)

- Persetujuan yang sah dan eksplisit
- Pemenuhan perjanjian (misalnya akad pembiayaan)
- Kewajiban hukum pengendali
- Kepentingan vital subjek data
- Tugas untuk kepentingan umum atau kewenangan
- Kepentingan yang sah lainnya

Ringkasan

Setiap *pipeline* yang memproses data pribadi harus dapat menjawab: **dasar pemrosesannya apa, dan untuk tujuan apa**.

Hak subjek data dan dampaknya pada *pipeline*

Hak subjek data (antara lain)	**Pekerjaan teknis yang muncul**
Mendapat informasi tujuan dan dasar pemrosesan	Katalog mencatat tujuan dan dasar per *dataset*
Mengakses dan memperoleh salinan datanya	Kemampuan mencari **semua** data milik satu nasabah di seluruh *lake* (butuh *lineage* dan kunci identitas yang konsisten)
Memperbaiki data yang tidak akurat	Koreksi mengalir ke *silver* dan *gold* (MERGE, P4)
Mengakhiri pemrosesan, menghapus, atau memusnahkan data	Penghapusan di tabel aktif, *snapshot*, arsip, dan *backup*; *crypto-shredding* (P14)
Menarik persetujuan	Status persetujuan ikut menjadi filter di *pipeline* pemasaran
Keberatan atas keputusan otomatis yang berdampak signifikan	*Scoring* pembiayaan perlu jalur peninjauan manusia dan penjelasan

Sumber: UU No. 27 Tahun 2022, bagian hak subjek data pribadi. Rumusan pasal lengkap: lihat UU PDP.

Kewajiban pengendali yang paling terasa di platform data

- **Notifikasi kegagalan pelindungan** kepada subjek data dan lembaga paling lambat **3** $\times$ **24 jam** (Pasal 46). Artinya deteksi dan penilaian insiden harus cepat, audit log dan *lineage* sangat menentukan.
- **Pejabat pelindungan data pribadi** (DPO) wajib ditunjuk pada kondisi tertentu, antara lain pemrosesan berskala besar (Pasal 53).
- **Penilaian dampak** pelindungan data pribadi untuk pemrosesan berisiko tinggi, misalnya *profiling* dan keputusan otomatis.
- **Transfer ke luar negeri** hanya dengan syarat tertentu (Pasal 56), relevan saat memilih *region cloud*.

Sanksi

Sanksi administratif dapat berupa peringatan, penghentian sementara pemrosesan, penghapusan data, hingga denda administratif paling tinggi **2% dari pendapatan tahunan** (Pasal 57). Ada pula ketentuan pidana untuk perbuatan tertentu.

Untuk *data engineer*

Kewajiban di kiri tidak bisa dipenuhi tanpa katalog, *classification*, *lineage*, audit log, dan kemampuan menghapus, semuanya pekerjaan kita.

Sumber: UU No. 27 Tahun 2022. Nomor pasal perlu dicek ulang terhadap naskah resmi.

GDPR sebagai pembanding

Aspek	UU PDP (Indonesia)	GDPR (Uni Eropa)
Notifikasi insiden	3×24 jam	72 jam kepada otoritas pengawas
Denda administratif maksimum	2% pendapatan tahunan	Hingga 4% omzet global atau EUR 20 juta, mana yang lebih besar
Peran	Pengendali, prosesor, DPO	*Controller*, *processor*, DPO
Hak penghapusan	Ada	*Right to erasure*
Data sensitif	Data pribadi spesifik	*Special categories*

- UU PDP banyak terinspirasi GDPR, sehingga praktik industri berbasis GDPR (*privacy by design*, *data minimization*, DPIA) umumnya relevan.
- GDPR penting bagi perusahaan Indonesia yang melayani penduduk Uni Eropa.

Sumber: Regulation (EU) 2016/679, Pasal 33 (notifikasi) dan Pasal 83 (denda).

ISO/IEC 27001 dan ISO/IEC 27701

ISO/IEC 27001:2022 — ISMS

Standar sistem manajemen keamanan informasi: organisasi menetapkan lingkup, menilai risiko, memilih kontrol, dan memperbaiki terus-menerus (*Plan-Do-Check-Act*).

Annex A memuat 93 kontrol dalam empat tema:

- **Organisational**: kebijakan, *classification*, pemasok, insiden
- ***People***: penyaringan, *training*, kerahasiaan
- ***Physical***: area aman, perangkat, media
- ***Technological***: *access control*, kriptografi, *logging*, *masking*, *backup*

ISO/IEC 27701 — PIMS

Perluasan untuk manajemen informasi privasi: kewajiban *controller* dan *processor*, hak subjek data, penilaian dampak. Sering dipakai sebagai kerangka operasional untuk menunjukkan kepatuhan UU PDP atau GDPR.

Ringkasan

Sertifikasi ISO tidak sama dengan patuh hukum, tetapi memberi **sistem** yang membuat kepatuhan bisa dibuktikan secara berulang.

Regulasi sektor keuangan (POJK tentang manajemen risiko teknologi informasi dan pelindungan konsumen) menambah kewajiban spesifik bagi bank; rujuk naskah POJK yang berlaku.

Memetakan kewajiban ke kontrol teknis

Kewajiban	Kontrol teknis di platform	Bukti untuk auditor
Data pribadi dilindungi kerahasiaannya	*Tokenization* di *silver*, RBAC/ABAC, *encryption* (P14)	*Configuration* kebijakan, hasil uji penolakan
Pemrosesan sesuai tujuan	*Tag* tujuan di katalog, akses per tujuan	Entri katalog, log akses
Notifikasi insiden 3×24 jam	Audit log terpusat, *alert* anomali, *runbook* insiden (hari ini)	Waktu deteksi pada latihan insiden
Hak akses dan hapus subjek data	*Lineage* tingkat kolom, *crypto-shredding*	Catatan permintaan dan waktu penyelesaian
Retensi sesuai ketentuan	*Lifecycle policy*, *object lock* (P3)	*Configuration lifecycle*, laporan pemusnahan

Ringkasan

Inilah inti GRC bagi *data engineer*: setiap baris kewajiban harus menunjuk ke kontrol yang **nyata di sistem** dan bukti yang **dapat ditunjukkan**.

Retention, *right to erasure*, dan amanah

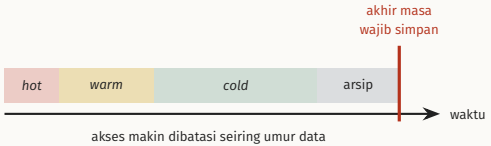
Retensi vs hak penghapusan

Dua kewajiban yang tampak bertabrakan:

- Bank **wajib menyimpan** catatan transaksi selama jangka waktu yang ditetapkan peraturan (misalnya ketentuan dokumen perusahaan dan anti pencucian uang).
- Nasabah berhak meminta datanya **dihapus** bila tidak ada lagi dasar pemrosesannya.

Cara menyelesaikannya

Kewajiban hukum adalah dasar pemrosesan yang sah. Data yang wajib disimpan tetap disimpan, tetapi **dibatasi** pemakaiannya (hanya untuk audit dan hukum), dipindah ke *tier* arsip, dan dihapus begitu masa wajibnya habis. Data di luar kewajiban itu (misalnya preferensi pemasaran) dihapus segera.



Panjang tiap *tier* hanya ilustrasi; jangka waktu sebenarnya mengikuti kebijakan retensi bank.

Immutable storage (P7)

WORM, *ledger*, dan *blockchain* tidak bisa dihapus. Maka data pribadi disimpan *off-chain* dan terenkripsi, yang *immutable* hanya *hash*-nya; penghapusan lewat *crypto-shredding* (P14).

Audit dan nilai amanah

Saat diaudit, siapkan

- Inventaris data pribadi: di mana, kelas apa, milik siapa
- *Risk register* dan status kontrolnya
- Matriks kewajiban → kontrol → bukti
- Log akses dan hasil uji kontrol berkala
- Catatan insiden dan latihan insiden
- Catatan permintaan subjek data dan waktu penyelesaiannya

Amanah

Data nasabah adalah titipan. Prinsip amanah memberi alasan yang lebih dalam daripada sekadar takut denda: menjaga titipan, memakainya hanya sesuai izin, dan mengembalikan atau memusnahkannya bila sudah tidak berhak disimpan.

Ringkasan

Kepatuhan yang dibangun ke dalam sistem membuat audit menjadi kegiatan rutin mengambil bukti.

When to use: alat *governance*

Alat	Pakai bila	Pertimbangkan alternatif bila
OpenMetadata	Katalog terbuka lengkap: glosarium, *classification*, *lineage*, *data quality test* dalam satu aplikasi	Organisasi sudah memakai katalog bawaan platform
DataHub	Metadata berbasis *event* dalam skala besar, banyak sumber, tim *platform* yang kuat	Tim kecil yang butuh instalasi sederhana
OpenLineage + Marquez	Standar *lineage* lintas Airflow, Spark, dbt, Flink	Katalog yang dipakai sudah mengumpulkan *lineage* sendiri
Unity Catalog / Lake Formation	Seluruh platform di Databricks atau AWS	Butuh netralitas *vendor*
Spreadsheet GRC	Awal program, *risk register* kecil	Ratusan kontrol dan banyak auditor (pakai alat GRC khusus)

Best practice: mulai dari *dataset* paling kritis (data pribadi spesifik dan laporan regulator), otomatiskan metadata teknis, dan tetapkan pemilik sebelum membeli alat.

Data observability

Pipeline hijau, data salah

Senin pukul 08.00 *dashboard* manajemen bank syariah digital menunjukkan pembiayaan murabahah yang cair kemarin **turun 60%**. Rapat direksi panik. Ternyata:

- DAG Airflow selesai **sukses**, semua *task* hijau
- Satu cabang mengganti format tanggal, sehingga 40% baris terbuang di *silver layer* karena gagal di-*parse*
- Tidak ada *exception*, tidak ada *alert*

Monitoring aplikasi tidak cukup

Monitoring infrastruktur menjawab "apakah mesinnya hidup?". *Data observability* menjawab "apakah **datanya** benar, lengkap, dan tepat waktu?". *Pipeline* bisa hidup sempurna sambil mengirim data yang salah.

Istilah: *data downtime*

Selang waktu ketika data hilang, terlambat, atau salah, dan biasanya baru ketahuan dari keluhan pengguna, bukan dari sistem.

Ringkasan

Tujuan pertemuan ini: masalah data ditemukan lebih dulu oleh **sistem Anda**, sebelum direksi melihatnya.

Lima pilar

Kepercayaan terhadap data

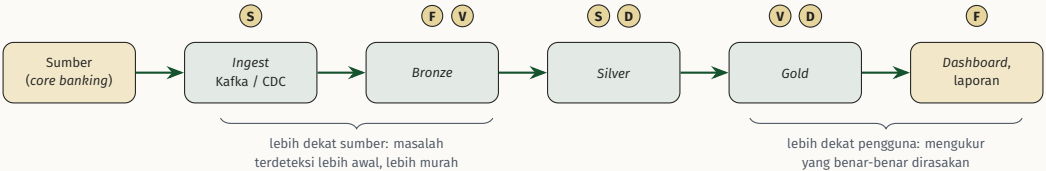
Freshness	**Volume**	***Schema***	***Distribution***	***Lineage***
Kapan data terakhir diperbarui? Terlambat dari jadwal?	Jumlah baris wajar? Tiba-tiba sepa-ruh atau dobel?	Kolom bertambah, hilang, atau berubah tipe?	Nilai *null*, rentang, rata-rata masih wajar?	Dari mana data ini, dan siapa yang terdampak?

Sumber: Moses, Gavish, dan Vorwerck, *Data Quality Fundamentals*, O'Reilly, 2022.

Ringkasan

Empat pilar pertama menjawab **apa yang salah**. Pilar kelima (*lineage*, bagian *governance* tadi) menjawab **seberapa luas dampaknya** dan **di mana akarnya**.

Di mana memasang pemeriksaan



F = *freshness*, V = *volume*, S = *schema*, D = *distribution*.

- ***Schema*** paling murah dicek saat *ingest*: *schema registry* (P11) dapat menolak *message* yang melanggar kompatibilitas.
- ***Freshness*** di sisi pengguna (*dashboard*) adalah yang dijanjikan dalam SLO.
- Jangan memasang semuanya di semua tabel: mulai dari tabel *gold* yang dipakai keputusan penting, lalu telusuri ke *upstream* lewat *lineage*.

When to use: alat *observability*

Alat	Pakai bila	Catatan
Uji dbt	Aturan yang sudah diketahui: `not_null`, `unique`, relasi, rentang	Hanya berjalan saat dbt berjalan
Soda Core / **Great Expectations**	Pemeriksaan deklaratif di luar dbt, termasuk *freshness* dan *volume*	Perlu dirawat seperti kode
Elementary	*Observability* untuk proyek dbt: anomali *volume*, *freshness*, laporan	Terikat ekosistem dbt
OpenLineage + Marquez / OpenMetadata	Analisis dampak dan akar masalah lewat *lineage*	Butuh integrasi di Airflow, Spark, dbt
Prometheus + Grafana + Alertmanager	*Metric* teknis: *consumer lag* Kafka, durasi *task*, memori	Bukan pemeriksa isi data
Layanan komersial	Tim besar yang ingin deteksi otomatis tanpa merawat aturan	*Cost* lisensi; data sensitif keluar?

Praktik industri

Gabungkan dua *layer*: ***metric* teknis** (Prometheus: *lag*, durasi) dan ***metric* isi data** (*freshness*, *volume*, *schema*, *distribution*). Satu tanpa yang lain meninggalkan titik buta.

Incident, SLO, dan *recovery*

Siklus penanganan insiden

Komunikasi berjalan sepanjang siklus: pemilik data, pengguna *dashboard*, kepatuhan bila menyangkut data pribadi



Mitigasi lebih dulu daripada perbaikan

Tandai *dashboard* “data sedang diperiksa”, hentikan *job downstream* agar data rusak tidak menyebar, kembalikan tabel ke *snapshot* sebelumnya (*time travel* Iceberg, P4).

Bila menyangkut data pribadi

Kebocoran data pribadi (*data breach*) adalah juga insiden kepatuhan: libatkan DPO; UU PDP mewajibkan pemberitahuan kegagalan pelindungan paling lambat 3 × 24 jam (bagian UU PDP tadi).

Contoh *runbook*: “tabel transaksi *silver* terlambat”

```
runbook: freshness-silver-transaksi
pemicu: freshness > 30 menit (SLO 15 menit)
pemilik: tim-data-platform (on-call)
langkah:
  1. Cek consumer lag Kafka topik
     core.transaksi (Grafana)
  2. Lag naik?  -> cek job Flink/Spark:
     status, checkpoint terakhir, error
  3. Lag nol tetapi data lama?
     -> sumber berhenti: cek Debezium
        connector & WAL PostgreSQL
  4. Job mati -> restart dari checkpoint
     terakhir; JANGAN reset offset ke
     latest (data hilang)
  5. Tandai dasbor saldo: "data tertunda"
  6. Setelah pulih: verifikasi volume per
     jam vs acuan, tutup insiden
eskalasi: > 60 menit -> SEV-2
```

Ciri *runbook* yang baik

- Ditulis untuk orang yang dibangunkan pukul 03.00 dan belum pernah menyentuh sistem itu
- Langkah berurutan, dengan perintah atau tautan *dashboard* yang tepat
- Menyebut **apa yang tidak boleh dilakukan**
- Punya kriteria eskalasi yang jelas
- Diuji: dijalankan saat latihan, diperbarui setelah setiap insiden

Postmortem tanpa menyalahkan

Isi *postmortem*

1. Ringkasan: apa yang terjadi, dampak, durasi
2. Linimasa: kapan mulai, kapan terdeteksi, kapan pulih
3. Akar masalah, teknik “5 mengapa”
4. Apa yang berjalan baik, apa yang tidak
5. Tindak lanjut dengan pemilik dan *deadline*

Contoh “5 mengapa” (format tanggal)

1. Mengapa angka murabahah turun? 40% baris terbuang.
2. Mengapa terbuang? Gagal *parse* tanggal.
3. Mengapa gagal? Satu cabang mengganti format.
4. Mengapa tidak terdeteksi? Tidak ada pemeriksaan *volume*.
5. Mengapa tidak ada? Tidak ada *data contract* dengan tim cabang.

Ringkasan

Tanpa menyalahkan (*blameless*): yang dicari adalah celah sistem, bukan orang yang salah. Orang yang takut disalahkan akan menyembunyikan insiden berikutnya.

SLI, SLO, SLA untuk data

- **SLI** (*indicator*): apa yang diukur. Contoh: persentase pemuatan per jam yang tiba $\leq$ 15 menit setelah jam berakhir.
- **SLO** (*objective*): target internal. Contoh: 99% pemuatan tepat waktu dalam 30 hari.
- **SLA** (*agreement*): janji kepada pihak lain beserta konsekuensinya. SLA selalu lebih longgar daripada SLO.

SLI data yang lazim

Freshness, kelengkapan (*completeness*), ketepatan waktu laporan, *end-to-end latency hot path* (*event* sampai skor *fraud*).

Hitungan *error budget*

30 hari $\times$ 24 pemuatan per hari $=$ 720 pemuatan. SLO 99% $\Rightarrow$ boleh terlambat

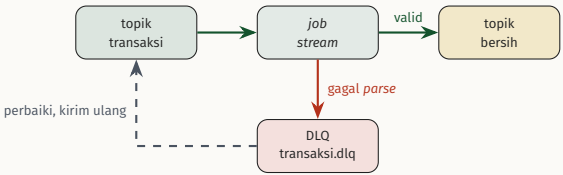
$$720 \times (1 - 0{,}99) = 7{,}2 \approx 7 \text{ pemuatan.}$$

Bila sudah 6 kali terlambat di pertengahan bulan, *budget* hampir habis: tunda perubahan berisiko, dahulukan *reliability*.

Jangan menargetkan 100%

Cost-nya naik tajam dan tidak ada ruang untuk perubahan. Tetapkan SLO dari kebutuhan keputusan, bukan dari keinginan.

Dead letter queue dan *replay*



- **DLQ**: *message* yang tidak dapat diproses dipindah ke topik terpisah beserta alasan gagalnya, sehingga *stream* utama tidak berhenti.
- Pantau jumlah *message* di DLQ sebagai *distribution metric*.

Replay dari Kafka (P11)

Kafka menyimpan *message* selama masa retensi. Bila logika *job* salah, jalankan *consumer group* baru atau setel ulang *offset* ke waktu sebelum kesalahan, lalu proses ulang.

Syaratnya: retensi cukup panjang dan *sink* **idempoten** (P6, P11), agar pemrosesan ulang tidak menggandakan data.

Retensi adalah keputusan

Retensi lebih panjang = jendela pemulihan lebih lebar, tetapi *storage cost* lebih besar, dan data pribadi tersimpan lebih lama (bagian *retention* tadi).

FinOps dan energi

Tiga sumber *cost*

Compute

Mesin yang menjalankan Spark, Flink, Trino, ClickHouse. Ditagih per jam-mesin atau per byte yang dipindai (layanan *serverless*).

Biasanya bagian terbesar.

Storage

Per GB per bulan, berbeda per *tier* (P3), plus *cost* per permintaan dan *retrieval cost* di *tier* dingin.

Murah per GB, tetapi menumpuk.

Network / egress

Data keluar dari *cloud* atau antar-*region*. Sering terlupa saat merancang.

Kejutan terbesar di tagihan.

Prinsip FinOps

Cloud cost adalah tanggung jawab bersama tim teknik, keuangan, dan bisnis. Siklusnya: ***inform*** (lihat siapa memakai apa) → ***optimize*** (kurangi pemborosan) → ***operate*** (jadikan kebiasaan dan kebijakan).

Sumber: FinOps Foundation, *FinOps Framework*. Harga *cloud*: selalu cek daftar harga terkini penyedia.

Lima tuas penghematan

Tuas	Cara kerja	Risiko
Partition* dan *clustering	*Query* hanya membaca *partition* dan blok yang relevan; byte terpindai turun	*Partition* terlalu kecil: *small files*
***Tiering* otomatis**	*Lifecycle policy* memindah data lama ke *tier* dingin (P3)	*Retrieval cost* bila sering dibaca
Autoscaling	*Cluster* membesar saat beban tinggi, mengecil (atau mati) saat sepi	Waktu *start* lambat
Spot* / *preemptible	Mesin sisa kapasitas, jauh lebih murah, bisa diambil kembali sewaktu-waktu	Hanya untuk *job* yang tahan diulang
Ukuran yang tepat	*Batch* puluhan GB di DuckDB/Polars satu mesin, bukan *cluster* (P5 vs P9–P10)	Batas memori satu mesin

Spot cocok untuk *batch* Spark yang idempoten (P6), dan tidak cocok untuk *job hot path* yang menjaga *latency* skor *fraud*.

Pemrosesan data juga punya jejak karbon

Energi listrik yang dipakai pusat data menghasilkan emisi sesuai sumber listriknya. Hitungan kasar:

$$\text{emisi} \approx \underbrace{P \times t}_{\text{energi (kWh)}} \times \text{PUE} \times \text{faktor emisi}$$

P: daya mesin (kW); t: lama berjalan (jam); PUE (*Power Usage Effectiveness*): rasio energi total pusat data terhadap energi mesin.

Contoh ilustrasi

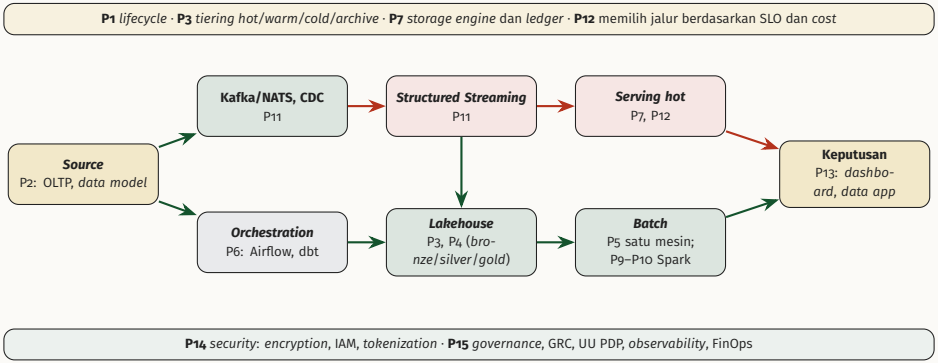
Job Spark harian: 10 mesin $\times$ 0,3 kW $\times$ 2 jam $=$ 6 kWh. Dengan PUE 1,5 menjadi 9 kWh per hari, $\approx$ 3.300 kWh per tahun. Kalikan dengan faktor emisi jaringan listrik setempat (kg CO_2e per kWh; gunakan angka resmi terbaru) untuk mendapat emisi.

- **PUE** (*Power Usage Effectiveness*): total energi pusat data dibagi energi perangkat TI; 1,0 berarti tanpa *overhead* pendinginan.
- Tuas FinOps hampir selalu juga tuas karbon: byte terpindai lebih sedikit, mesin lebih sedikit, data dingin di media hemat energi.
- Waktu dan lokasi berpengaruh: menjalankan *batch* di *region* atau jam dengan listrik lebih bersih (*carbon-aware scheduling*).
- Pelaporan emisi organisasi (ESG) makin sering diminta; pusat data termasuk di dalamnya.

Sumber: Barroso, Holzle, dan Ranganathan, *The Datacenter as a Computer*, ed. 3 (PUE dan efisiensi energi).

Satu semester dalam satu peta

Semua pertemuan, satu platform



UTS membangun jalur hijau di satu mesin; UAS membawanya ke Spark skala besar sampai *dashboard* atau aplikasi, dengan jalur merah sebagai nilai tambah.

Praktikum gabungan

Praktikum 1: OpenMetadata dan *lineage* otomatis

```yaml
# ingestion OpenMetadata dari Trino
source:
  type: trino
  serviceName: lakehouse_bank
  serviceConnection:
    config:
      type: Trino
      hostPort: trino:8080
      catalog: iceberg
      username: om_reader      # akun baca saja
  sourceConfig:
    config:
      type: DatabaseMetadata
      schemaFilterPattern:
        includes: ["silver", "gold"]
sink:
  type: metadata-rest
workflowConfig:
  openMetadataServerConfig:
    hostPort: http://openmetadata:8585/api
```

1. Jalankan OpenMetadata dengan Docker Compose resmi.
2. Tarik metadata dari Trino (*silver*, *gold*) dan dbt (`manifest.json`) agar *lineage* model dbt ikut tergambar.
3. Isi pemilik, *steward*, deskripsi, dan glosarium untuk tiga tabel *gold*.
4. Tandai kolom PII dengan *classification* `PII.Spesifik`.
5. Tangkap layar *lineage* kolom `nik` → `nik_token`, dan tunjukkan bahwa kolom itu tidak sampai ke gold.

Sumber: Dokumentasi OpenMetadata (*connectors*); struktur *configuration* dapat berubah antar versi.

Praktikum: pemeriksaan *freshness* dan *volume* dengan *alert*

```python
# obs/cek_tabel.py -- dijalankan Airflow tiap 15 menit
import duckdb, json, os, urllib.request
from datetime import datetime, timezone

SLO_MENIT = 15
con = duckdb.connect()
con.sql("INSTALL iceberg; LOAD iceberg;")  # + kredensial MinIO

telat = con.sql("""
  SELECT date_diff('minute', max(event_time), now())
  FROM iceberg_scan('s3://lake/silver/transaksi')
""").fetchone()[0]

n, mu, sigma = con.sql(open("obs/volume.sql").read()).fetchone()
z = (n - mu) / sigma if sigma else 0.0

masalah = []
if telat > SLO_MENIT: masalah.append(f"freshness: telat {telat} menit")
if abs(z) > 3:        masalah.append(f"volume: z = {z:.1f}")

if masalah:
    teks = "[silver.transaksi] " + "; ".join(masalah)
    teks += " -- runbook: freshness-silver-transaksi"
    req = urllib.request.Request(os.environ["WEBHOOK_URL"],
        data=json.dumps({"text": teks}).encode(),
        headers={"Content-Type": "application/json"})
    urllib.request.urlopen(req)
```

- `volume.sql`: jumlah baris jam terakhir dibanding rata-rata dan simpangan jam yang sama pada empat pekan sebelumnya (*seasonal z-score*).
- `WEBHOOK_URL` disimpan di Vault atau *secret* Airflow (P14), bukan di kode.
- *Alert message* menyebut *runbook*-nya: orang yang menerima langsung tahu langkah pertama.
- Simpan juga setiap hasil ke `obs.hasil_cek` untuk *dashboard* riwayat.

Praktikum gabungan: yang dikumpulkan

Artefak *governance* dan GRC

1. Tabel *silver* dan *gold* tugas besar terdaftar di OpenMetadata: *owner*, *steward*, deskripsi, *classification* PII, *lineage*.
2. Satu *data contract* YAML yang divalidasi oleh *pipeline*.
3. *Risk register* lima risiko teratas beserta kontrol dan *owner*.
4. Matriks kewajiban UU PDP $\rightarrow$ kontrol $\rightarrow$ bukti, merujuk kontrol P14 yang sudah dipasang.

Artefak *observability*

1. Pemeriksaan *freshness* dan *volume* pada tabel *gold* yang dibaca *dashboard*, dijadwalkan Airflow, dengan *alert*.
2. Satu *runbook* untuk *alert* itu.
3. SLO tertulis untuk tabel yang tampil di *dashboard* UAS.

Kriteria

Setiap baris menunjuk bukti yang bisa dibuka (*file*, tangkapan layar, log). Kontrol yang belum ada ditulis sebagai celah, bukan disamarkan.

Presentasi progres tugas besar UAS

Presentasi progres UAS: 10 menit per kelompok

Yang harus sudah berjalan

1. *Pipeline* Spark *bronze* → *silver* → *gold* berjalan di *cluster* beberapa *worker*.
2. Data berukuran besar (atau *generator* yang bisa diskalakan) sudah masuk *bronze*.
3. Minimal satu angka pengukuran *scaling* (*strong* atau *weak*) dan pembanding *baseline* satu mesin dari UTS.
4. Draf *dashboard* atau *data app* yang membaca *gold* lewat *serving store*.

Format

- 3 menit arsitektur (diagram, SLO, pilihan *store*)
- 4 menit demo: jalankan satu *job*, tunjukkan Spark UI dan *dashboard*
- 3 menit hambatan dan rencana sampai UAS

Yang sering terlewat

Tokenization PII di *silver* (P14), *lineage* sampai ke *dashboard*, dan pengukuran yang diulang lebih dari sekali.

Tugas untuk pertemuan berikutnya

1. Kumpulkan artefak praktikum gabungan (*catalog*, *data contract*, *risk register*, matriks UU PDP, pemeriksaan *freshness*/*volume*, *runbook*).
2. Lengkapi tugas besar UAS: pengukuran *scaling* yang diulang, *dashboard* atau *data app* final, kontrol *security*, dan artefak *governance*.
3. Latih demo end-to-end: dari data masuk *bronze* sampai angka tampil di *dashboard*, dalam waktu presentasi.

Pertemuan depan

UAS, presentasi tugas besar "*Pipeline* data skala besar dengan Apache Spark sampai *dashboard* atau aplikasi".

Terima kasih

Pertanyaan?