# Privacy by design

Jfokus, 2018-02-07
Lars Albertsson
www.mapflat.com

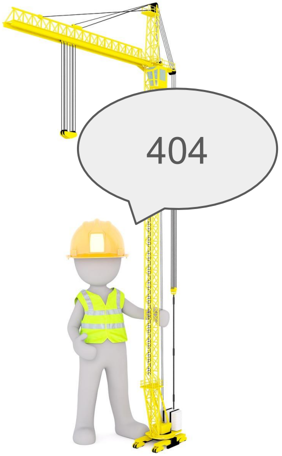
# Who’s talking?

- KTH-PDC Center for High Performance Computing (MSc thesis)
- Swedish Institute of Computer Science (distributed system test+debug tools)
- Sun Microsystems (building very large machines)
- Google (Hangouts, productivity)
- Recorded Future (natural language processing startup)
- Cinnober Financial Tech. (trading systems)
- Spotify (data processing & modelling)
- Schibsted Media Group (data processing & modelling)
- Mapflat (independent data engineering consultant)
    - ~15 clients: Spotify, 3 banks, 3 conglomerates, 4 startups, 5 *tech, misc

# Privacy protection resources

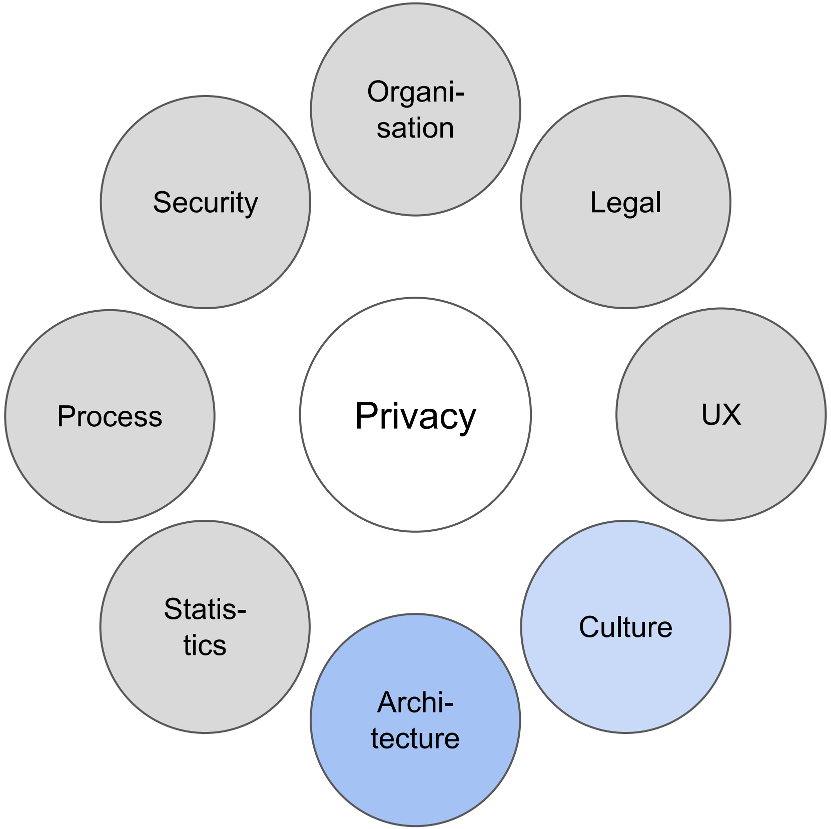
All of this might go wrong. Large fine.

Pour your data into our product.



# Privacy by design

- Required by GDPR
- Technical scope
  - Engineering toolbox
  - Puzzle pieces - not complete solutions
- Assuming that you solve:
  - Legal requirements
  - Security primitives
  - ...
- Disclaimers:
  - This is not a description of company X
  - This is not legal / compliance advice



# Requirements, engineer’s perspective

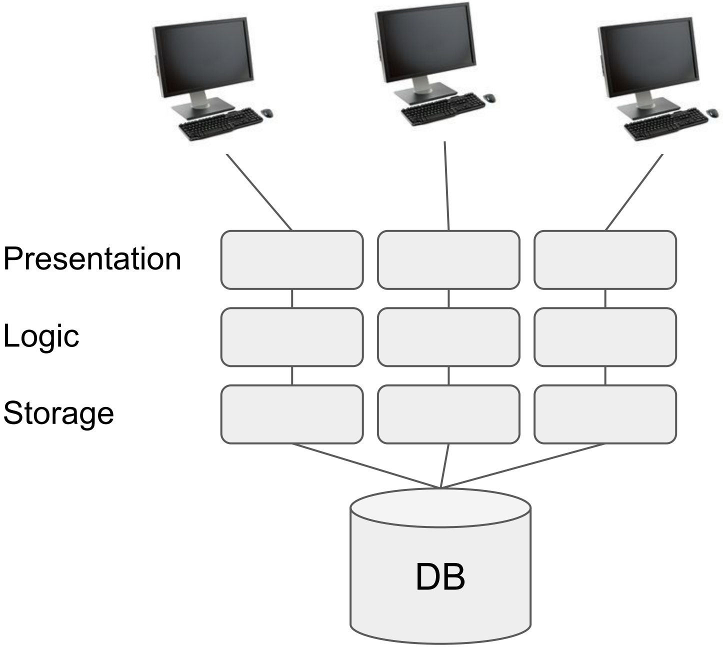
- **Right to be forgotten**
- Limited collection
- **Limited retention**
- **Limited access**
  - From employees
  - In case of security breach
- **Consent for processing**
- Right for explanations
- **Right to correct data**
- User data enumeration
- User data export

# Ancient data-centric systems

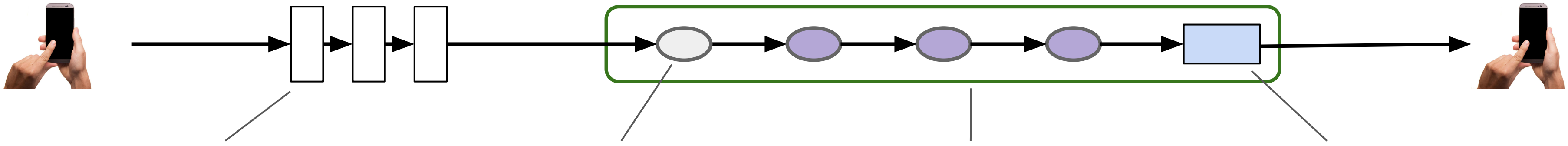
- The monolith
- All data in one place
- Analytics + online serving from single database
- Current state, mutable

- Please delete me?
- What data have you got on me?
- Please correct this data

- Sure, no problem!



# Event-oriented / big data systems



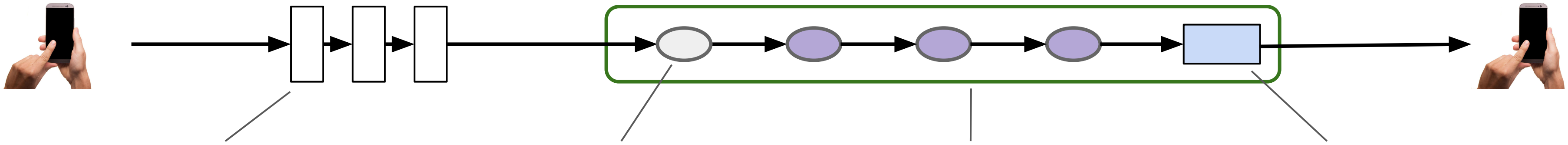
Every event

All events, ever,
raw, unprocessed

Refinement
pipeline

Artifact of
value

# Event-oriented / big data systems



Every event

All events, ever,
raw, unprocessed

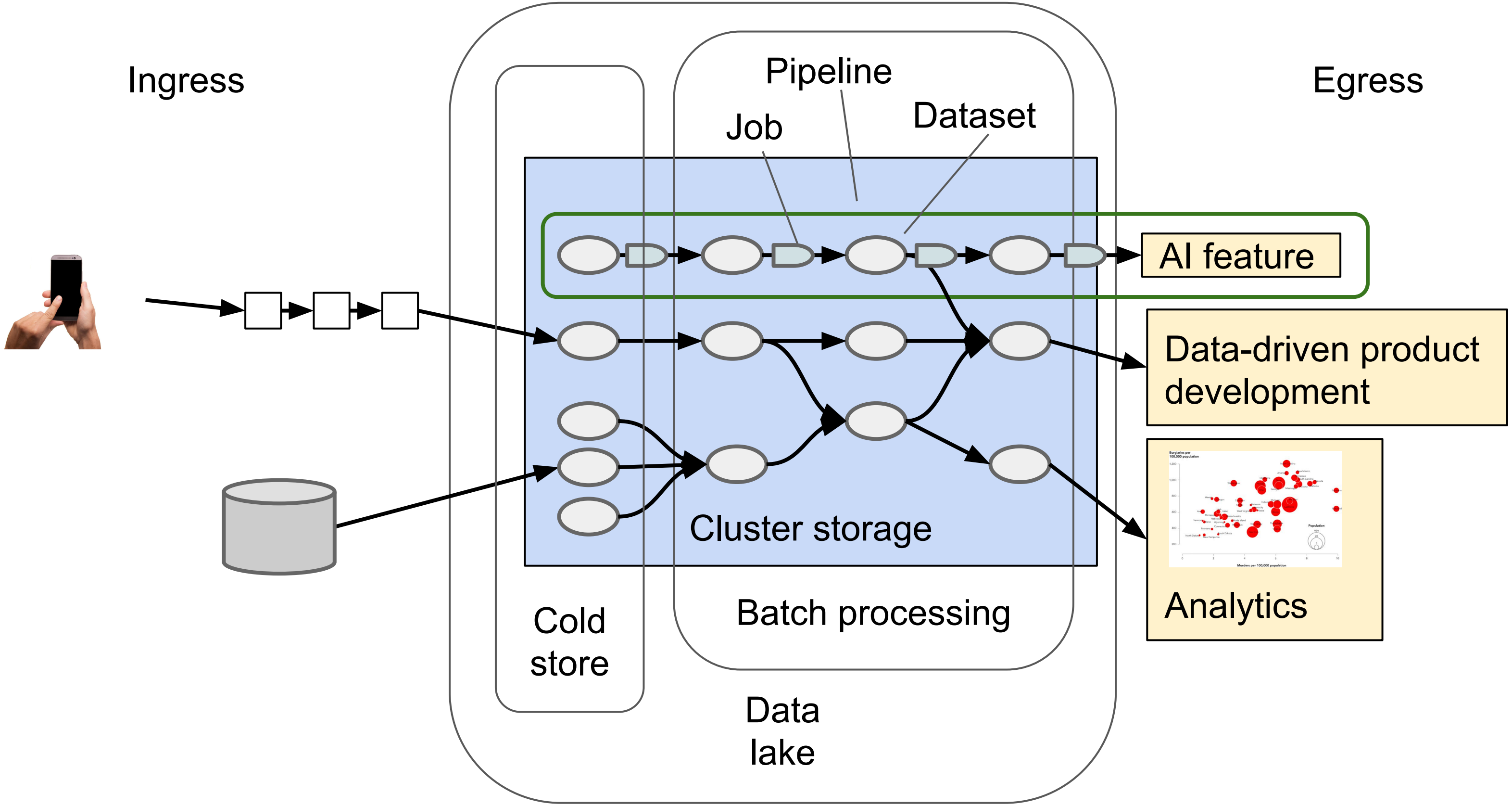
Refinement
pipeline

Artifact of
value

- Motivated by
  - New types of data-driven (AI) features
  - Quicker product iterations
    - Data-driven product feedback (A/B tests)
    - Democratised data - fewer teams involved in changes
  - Robustness - scales to more complex business logic

*Enable disruption*

# Data processing at scale

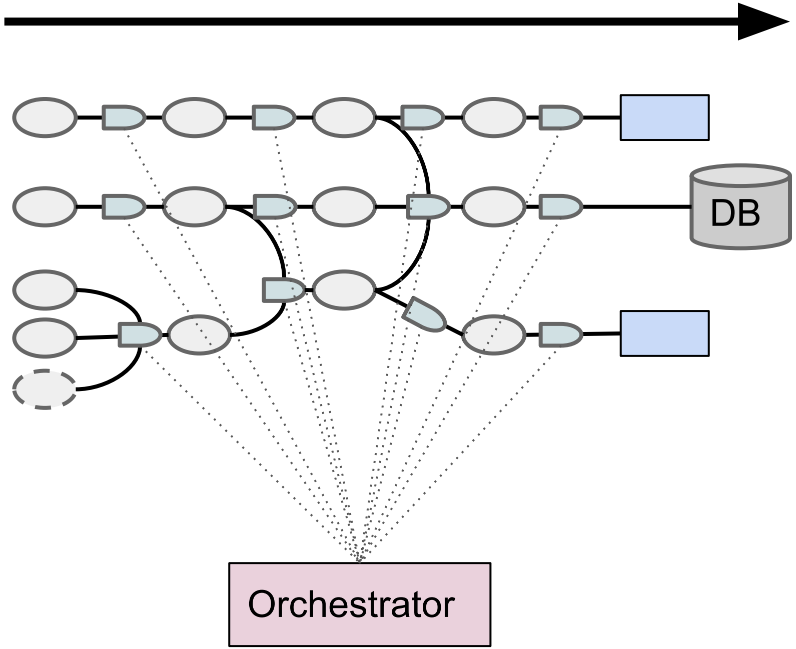


# Workflow orchestrator

- Dataset “build tool”
- Run job instance when
  - input is available
  - output missing
  - resources are available
- Backfill for previous failures
  - Robust system from fragile components
- DSL describes DAG
  - Includes ingress & egress

The most important big data component - it keeps you sane

Recommended: Luigi / Airflow



# Factors of success

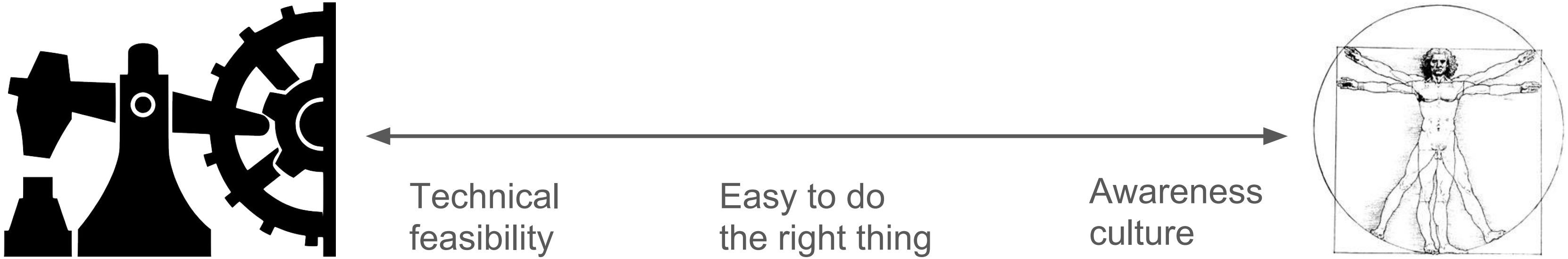
Functional architecture:

- Event-oriented - append only
- Immutability
- At-least-once semantics
- Reproducibility
  - Through 1000s of copies
- Redundancy

- Please delete me?
- What data have you got on me?
- Please correct this data

- Hold on a second...

# Solution space



Technical feasibility

Easy to do the right thing

Awareness culture

# Personal information (PII) classification

You need to establish a field/dataset classification. Example:

Is application content sensitive? Depends.

- Music, video playlists - perhaps not
- Running tracks, taxi rides - apparently
- In-application messages - probably

- Red - sensitive data
  - Messages
  - GPS location
  - Views, preferences
- Yellow - personal data
  - IDs (user, device)
  - Name, email, address
  - IP address
- Green - insensitive data
  - Not related to persons
  - Aggregated numbers

- Grey zone
  - Birth date, zip code
  - Recommendation / ads models?

# PII arithmetics

- Most sensitive data wins
  - red + green = red
  - red + yellow = red
  - yellow + green = yellow
- Aggregation decreases sensitivity
  - sum(red/yellow) = green ?
- Combinations may increase sensitivity
  - green + green + green = yellow ?
  - yellow + yellow + yellow = red ?
- Machine learning models store hidden information
  - model(yellow) = yellow or green ?
  - *Overfitting => persons could be identified*

# Make privacy visible at ground level

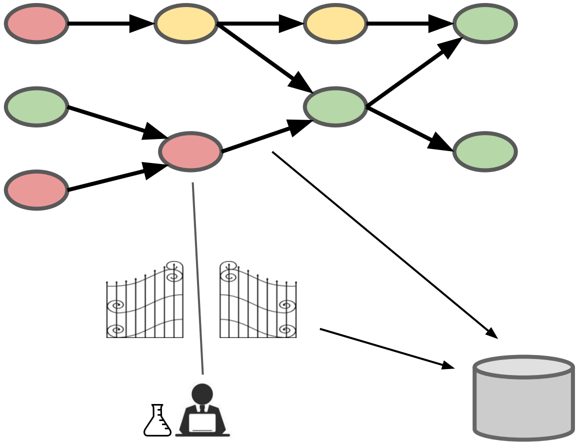
Suggestions:

- In dataset names
  - hdfs://red/crm/received_messages/year=2017/month=6/day=13
  - s3://yellow/webshop/pageviews/year=2017/month=6/day=13
- In field names
  - response.y_text = “Dear ” + user.y_name + “, thanks for contacting us …”
- In credential / service / table / ... names
- In metadata

- Spreads awareness
- Catch mistakes in code review
- Enables custom tooling for violation warnings
- Difficult to change privacy level

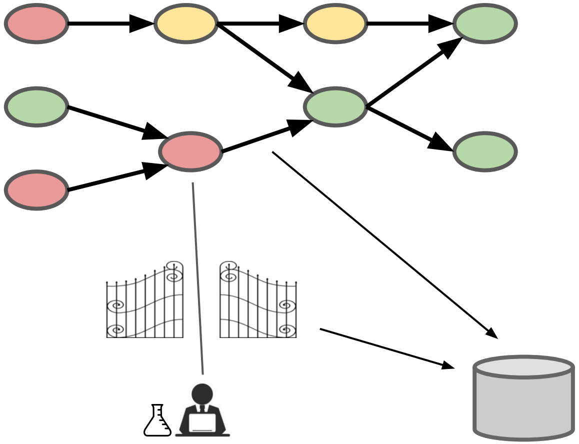
# Eye of the needle tool

- Provide data access through gateway tool
    - Thin wrapper around Spark/Hadoop/S3/...
    - Hard-wired configuration
- Governance
    - Access audit, verification
    - Policing/retention for experiment data



# Eye of the needle tool

- Easy to do the right thing
  - Right resource choice, e.g. “allocate temporary cluster/storage”
  - Enforce practices, e.g. run jobs from central repository code
  - No command for data download
- Enabling for data scientists
  - Empowered without operations
  - Directory of resources

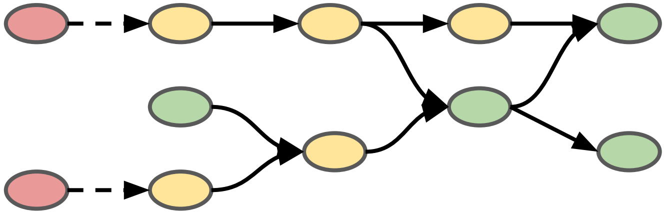


# Possible strategy: Privacy protection at ingress

Scramble on arrival

- + Simple to implement
- - Limits value extraction
- - Deanonymisation possible

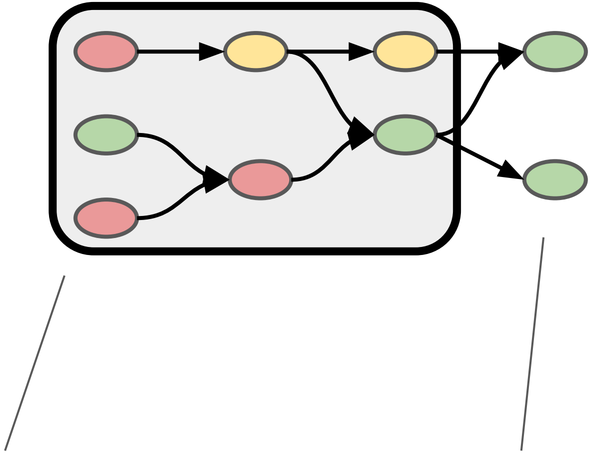
IMHO not a feasible strategy



# Privacy protection at egress

Processing in opaque box

- + Enabling
- + Simpler to reason about
- - Strict operations required
- - Exploratory analytics need explicit egress / classification



Machines are allowed to see intermediate data

Humans & services interact with exported data

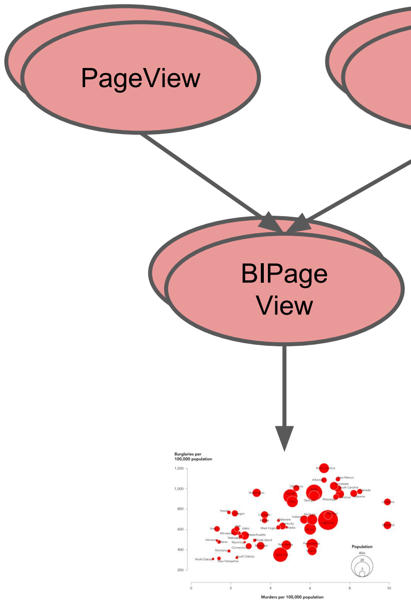
# Permission to process

- Processing personal data requires a sanction
    - Business motive is not sufficient
- Explicit sanction
    - Consent from user
    - Necessary to perform core service
- Implicit sanction
    - Required by regulations
        - Detect money laundry, fraud, abuse
        - Bookkeeping
- Not exempt user
    - Not underage
    - Not politically exposed person
    - No hidden identity

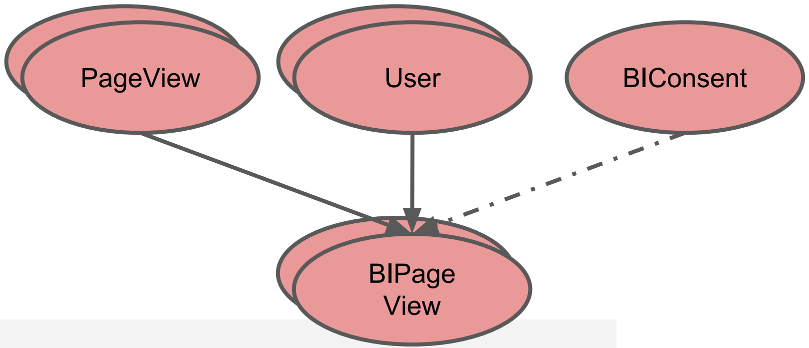
# Consent workflow

- Consent applies at processing date, not collection date

*Normal decoration join - same date*



*Consent join - always latest*

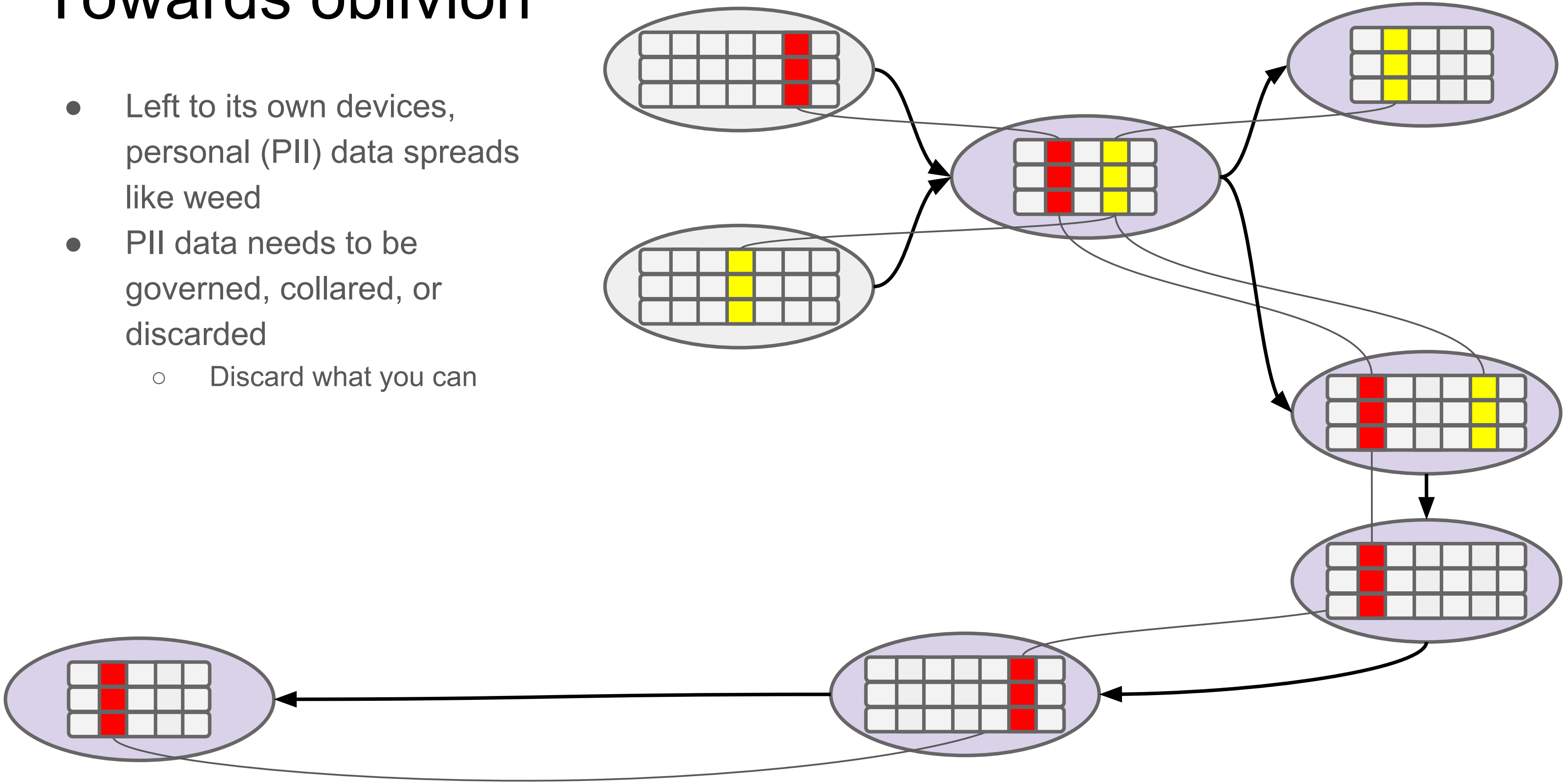


```
class BiPageView(Task):
  date = DateParameter()

  def requires(self):
    return [PageView(self.date),
            User(self.date),
            BIConsent.latest()]
```

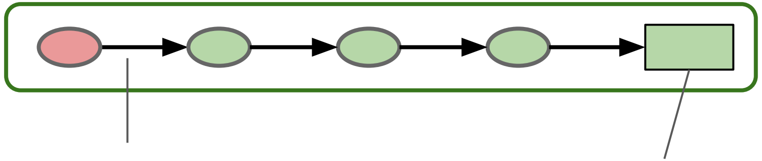
# Towards oblivion

- Left to its own devices, personal (PII) data spreads like weed
- PII data needs to be governed, collared, or discarded
  - Discard what you can



# Discard: Anonymisation

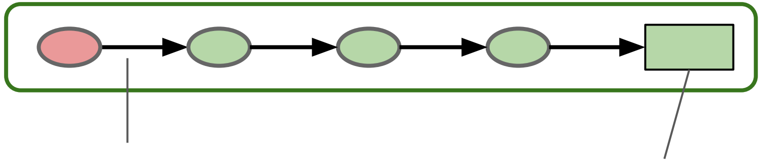
- Discard all PII
  - User id in example
- No link between records or datasets



Drop user id

Useful for metrics

- Replace with non-PII
  - E.g. age, gender, country
- Still no link
  - Beware: rare combination => not anonymised



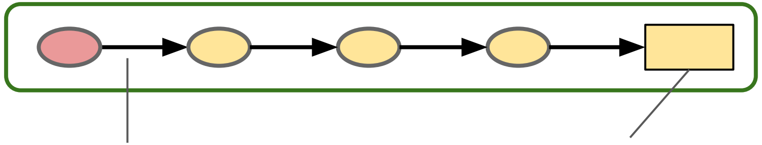
Replace user id with demographics

Useful for business insights

# Partial discard: Pseudonymisation

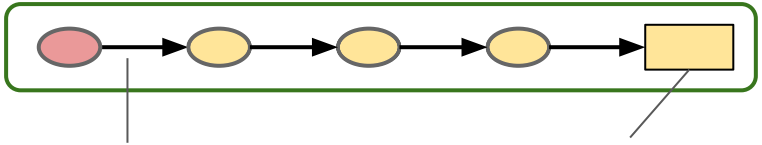
- Hash PII
- Records are linked
  - Across datasets
  - Still PII, GDPR applies
  - Persons can be identified (with additional data)
  - Hash recoverable from PII

- Hash PII + salt
  - Hash not recoverable
- Records are still linked
  - Across datasets if salt is constant



Hash user id

Useful for recommendations



Hash user id + salt

Useful for product insights

# Governance: Recomputation

- Push reruns with workflow orchestrator

- No versioning support in tools

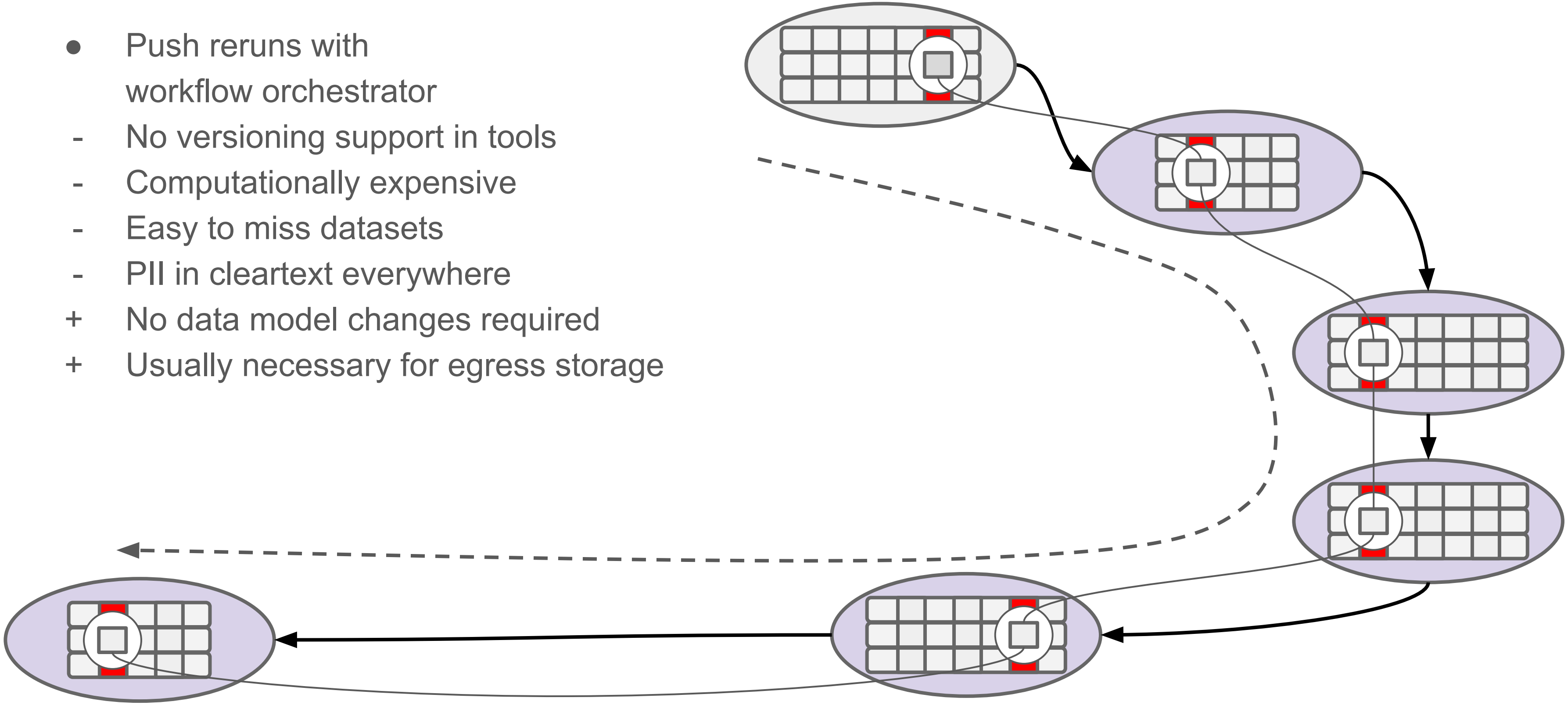
- Computationally expensive

- Easy to miss datasets

- PII in cleartext everywhere

+ No data model changes required

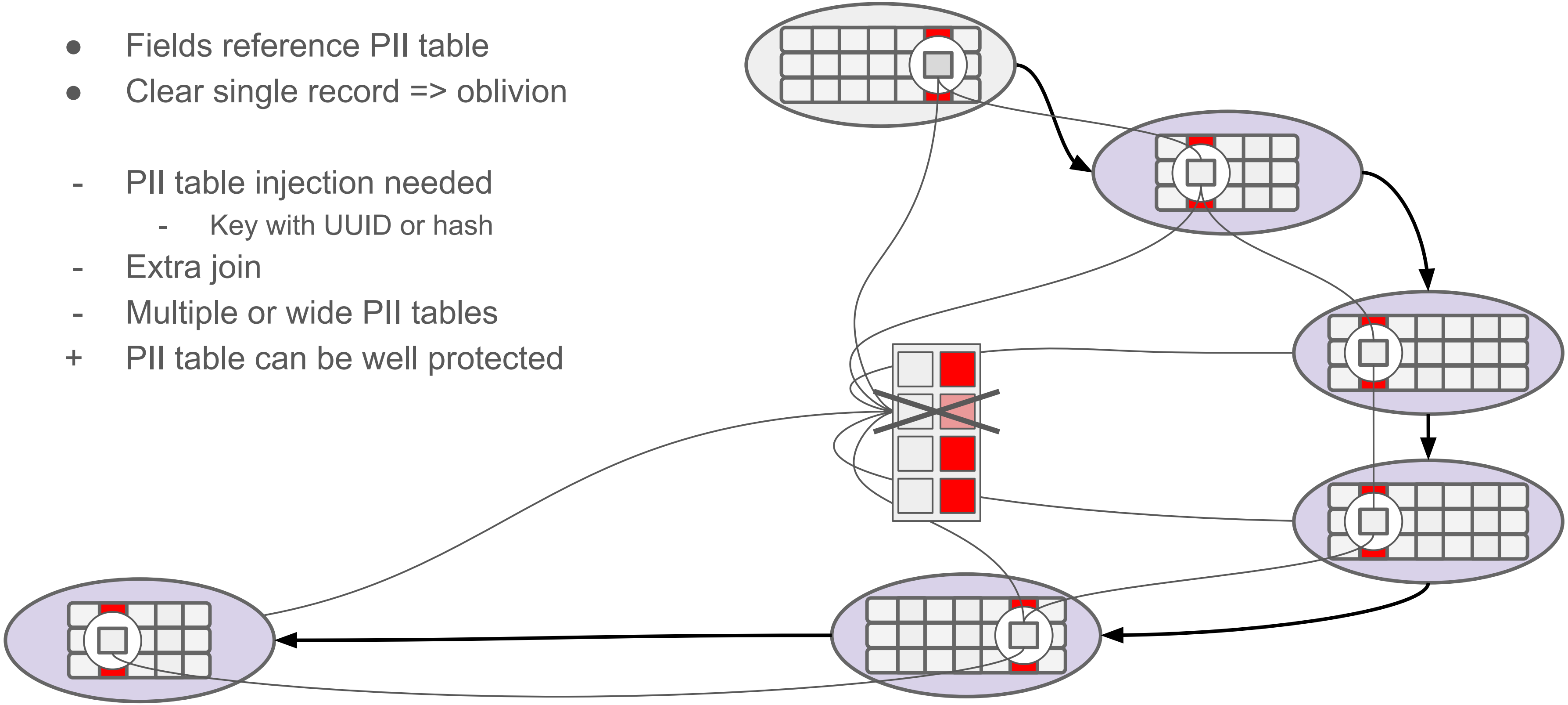
+ Usually necessary for egress storage



# Ejected record pattern

- Fields reference PII table
- Clear single record => oblivion

- - PII table injection needed
  - - Key with UUID or hash
- - Extra join
- - Multiple or wide PII tables
- + PII table can be well protected

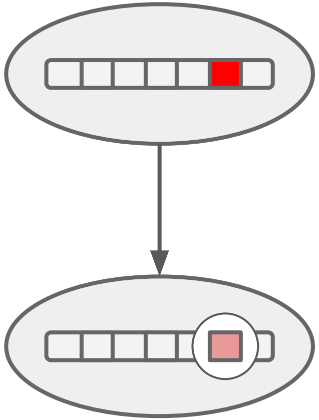


# Record removal in pipelines

- Datasets are immutable - must not remove records
- Version n+1 of raw dataset lacks record
- Short retention of old versions
- *Always depend on latest version*
    - What about changing PII, e.g. address?
      Need versioning in data model?

```
class Purchases(Task):
  date = DateParameter()

  def requires(self):
    return [Users(self.date),
            Orders(self.date),
            UserPII.latest()]
```



User PII
2017-06-12

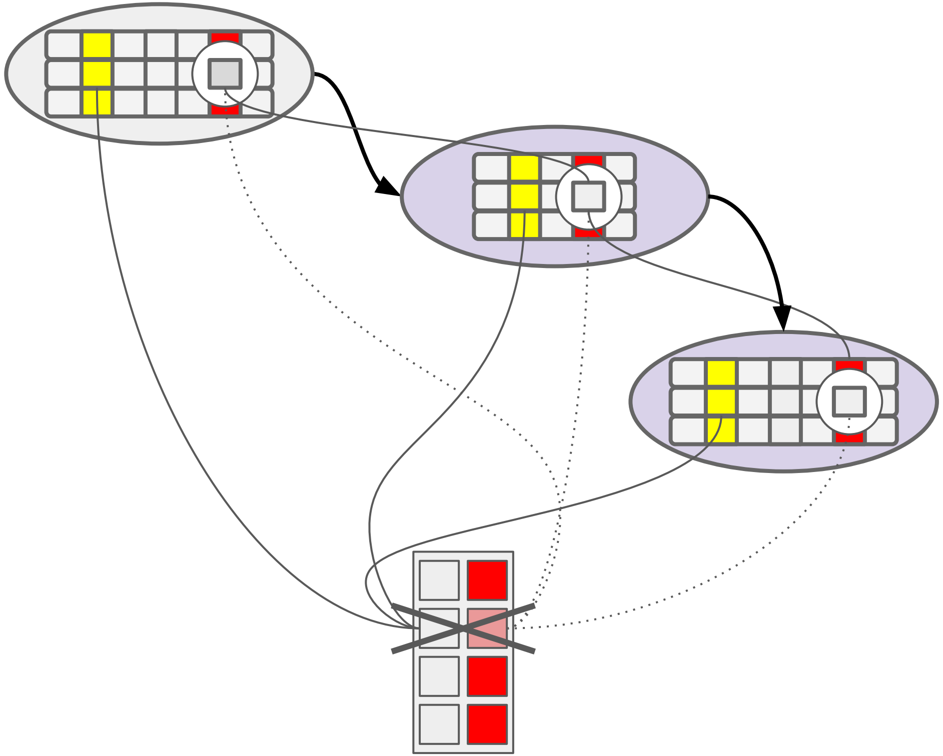
User PII
2017-06-13

# Lost key pattern

- PII fields encrypted
- Per-user decryption key table
- Clear single user key => oblivion

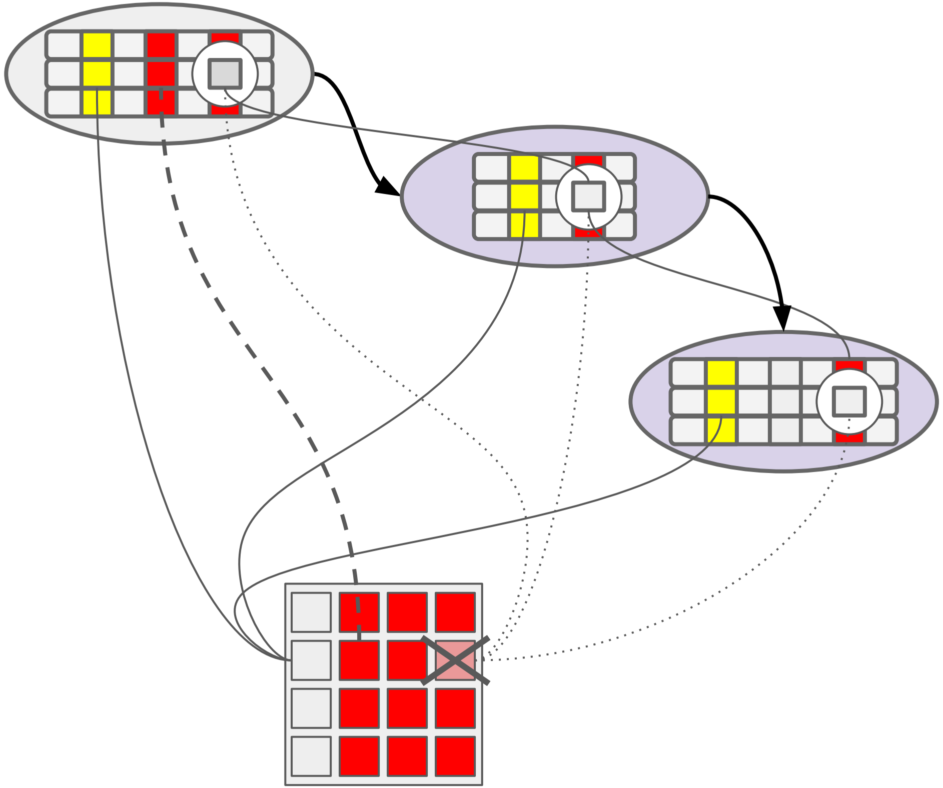
- Extra join + decrypt
    - Requires user-defined function in SQL?
- Decryption (user) id needed

+ Multi-field oblivion
+ Single dataset leak → no PII leak
+ Handles changing PII fields



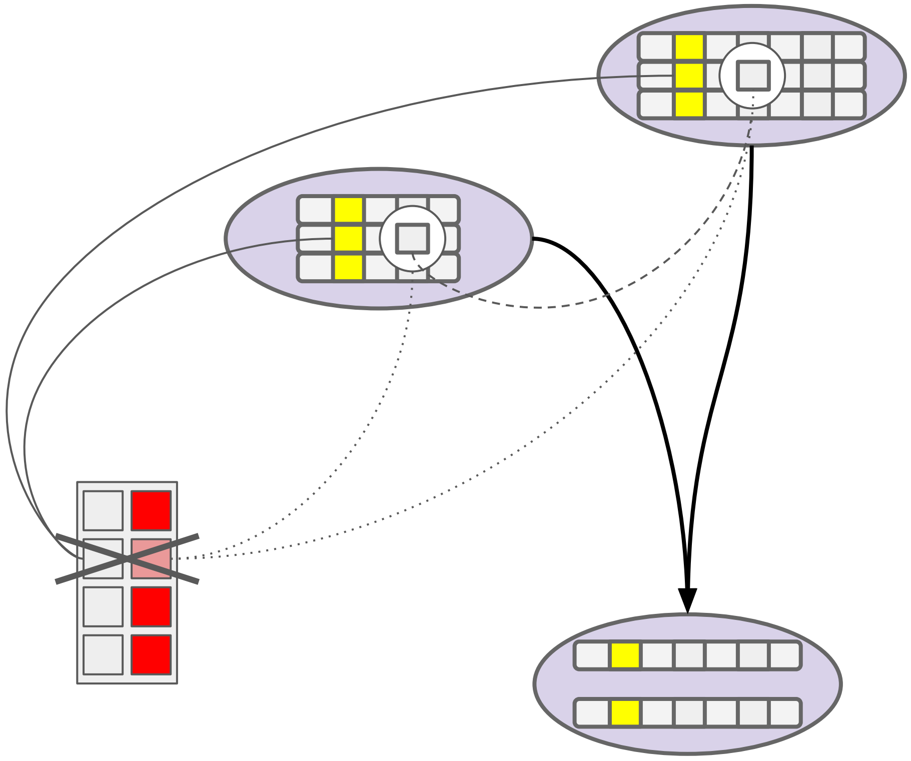
# Lost key partial oblivion

- Different fields encrypted with different keys
- Partial user oblivion
  - E.g. forget my GPS coordinates



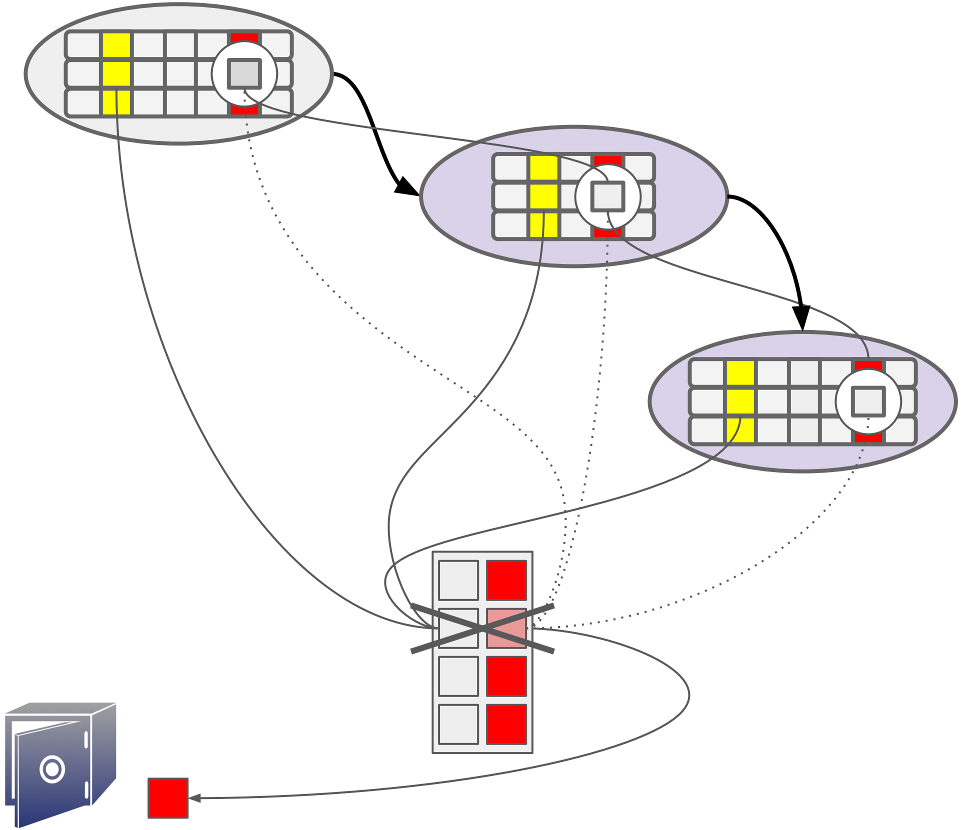
# Lost link key

- Encrypt key fields that link datasets
- Ability to join is lost
- No data loss
    - Salt => anonymous data
    - No salt => pseudonymous data

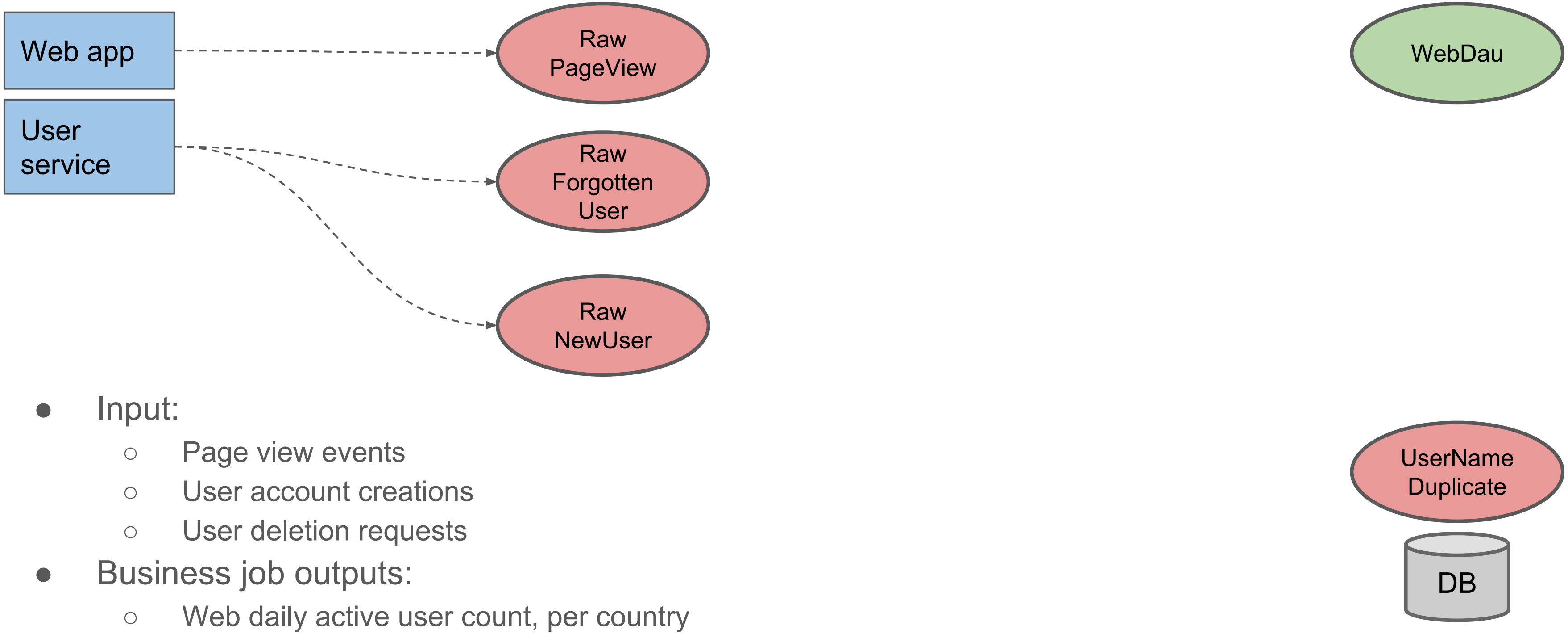


# Reversible oblivion

- Lost key pattern
- Give ejected record key to third party
    - User
    - Trusted organisation
- Destroy company copies

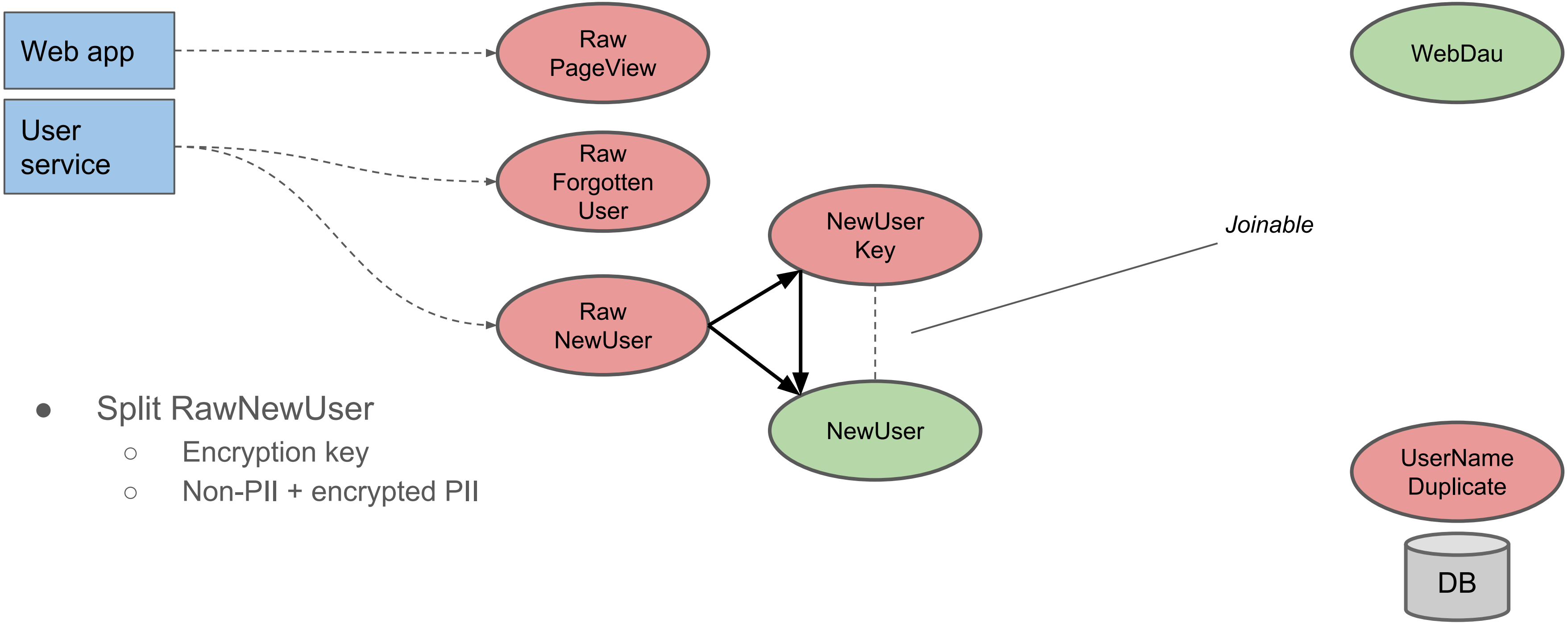


# Example: Lost key pattern



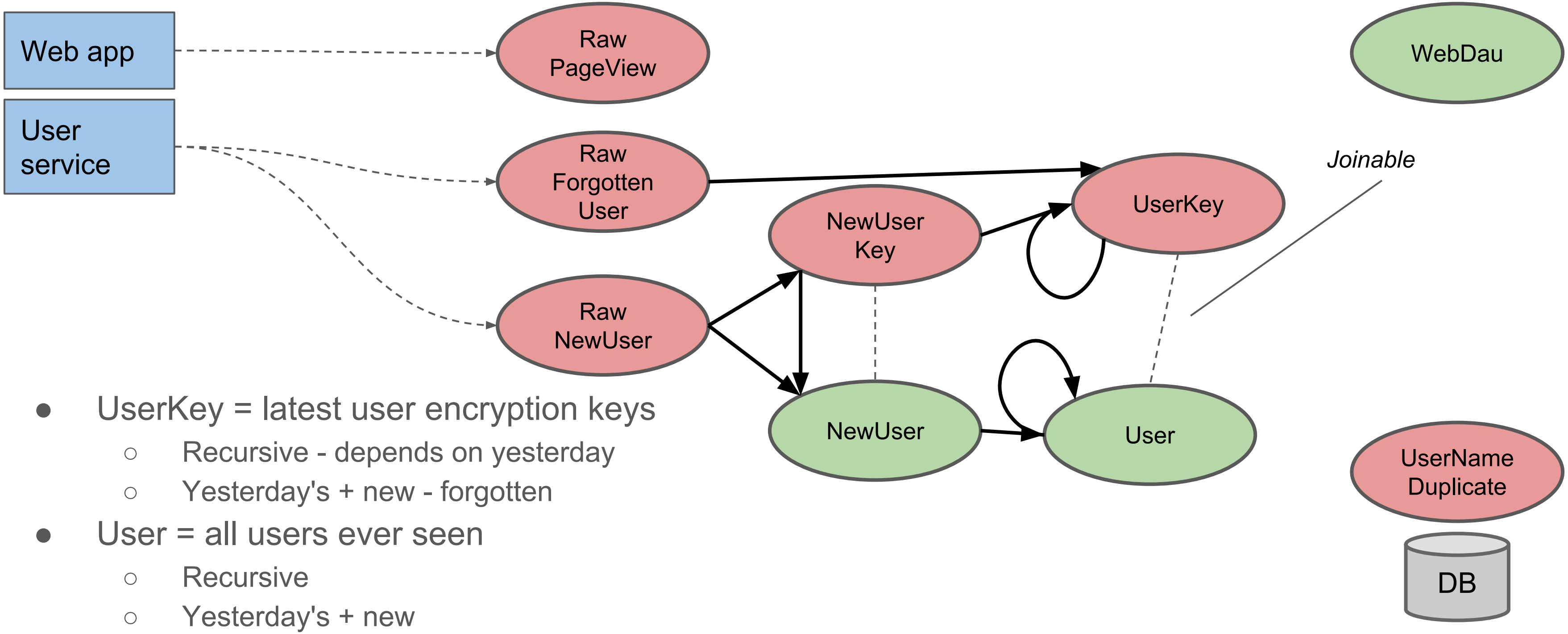
- Input:
  - Page view events
  - User account creations
  - User deletion requests
- Business job outputs:
  - Web daily active user count, per country
  - Duplicate display name detection → email

# Example: Lost key pattern



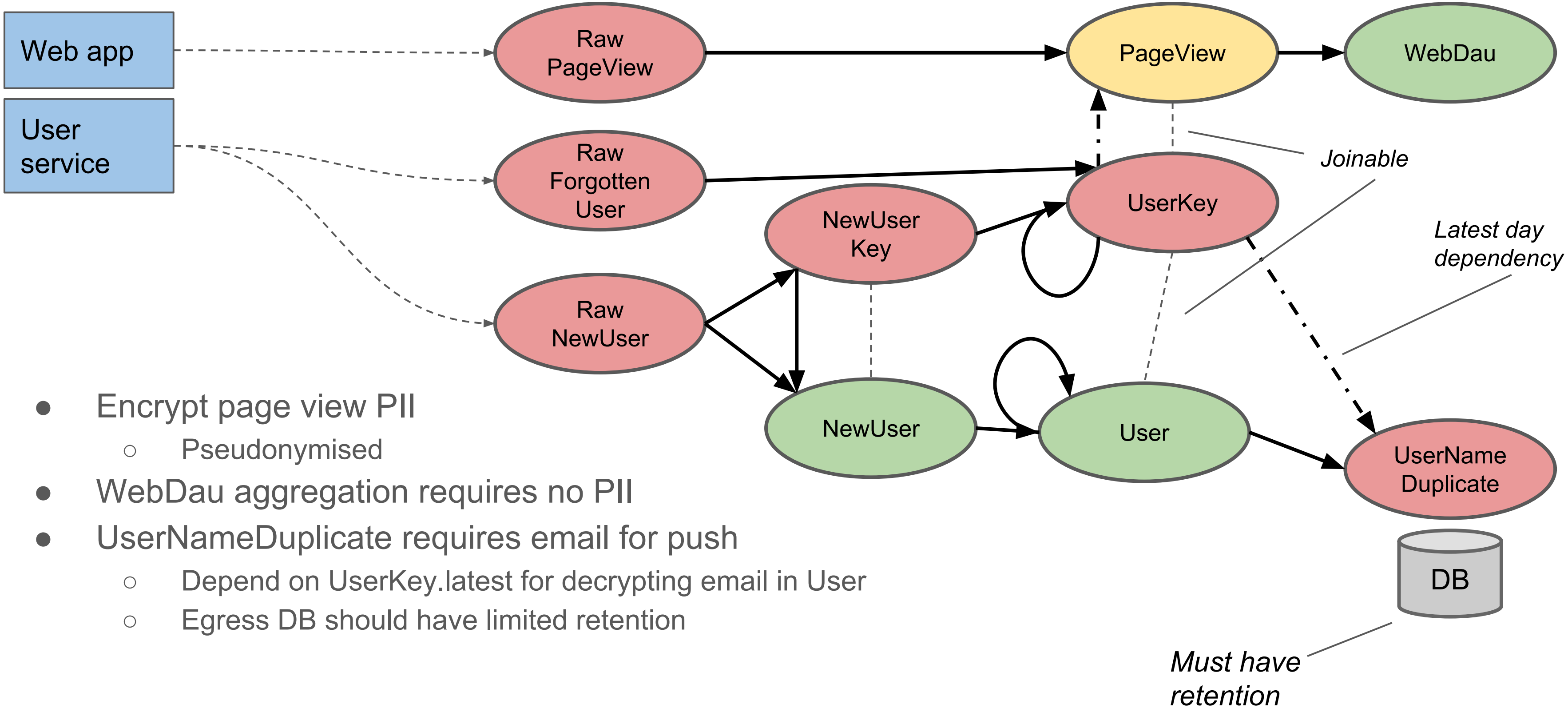
- Split RawNewUser
  - Encryption key
  - Non-PII + encrypted PII

# Example: Lost key pattern



- UserKey = latest user encryption keys
  - Recursive - depends on yesterday
  - Yesterday's + new - forgotten
- User = all users ever seen
  - Recursive
  - Yesterday's + new
    - grows forever
  - Encrypted PII

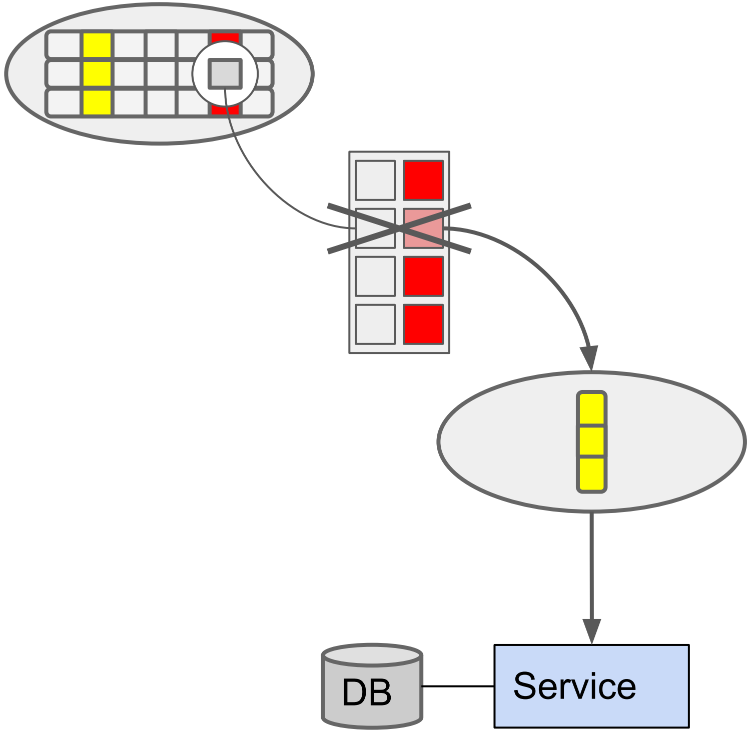
# Example: Lost key pattern



- Encrypt page view PII
  - Pseudonymised
- WebDau aggregation requires no PII
- UserNameDuplicate requires email for push
  - Depend on UserKey.latest for decrypting email in User
  - Egress DB should have limited retention

# Tombstone line

- Produce dataset/stream of forgotten users
- Egress components, e.g. online service databases, may need push for removal.
    - Higher PII leak risk



# The art of deletion

- Example: Cassandra
- Deletions == tombstones
- Data remains
    - Until compaction
    - In disconnected nodes
    - ...

*Component-specific expertise necessary*

# Deletion layers

- Every component adds deletion burden
  - Minimise number of components
  - Ephemeral >> dedicated. Recycle machines.
- Every storage layer adds deletion burden
  - Minimise number of storage layers
  - Cloud storage requires documented erasure semantics + agreements.
- Invent simple strategies
  - Example: Cycle Cassandra machines regularly, erase block devices.

*Increasing cost of heterogeneity & on premise storage.*

# Data model deadly sins

- Using PII data as key
  - Username, email

- Publishing entity ids containing PII data
  - E.g. user shared resources (favourites, compilations) including username

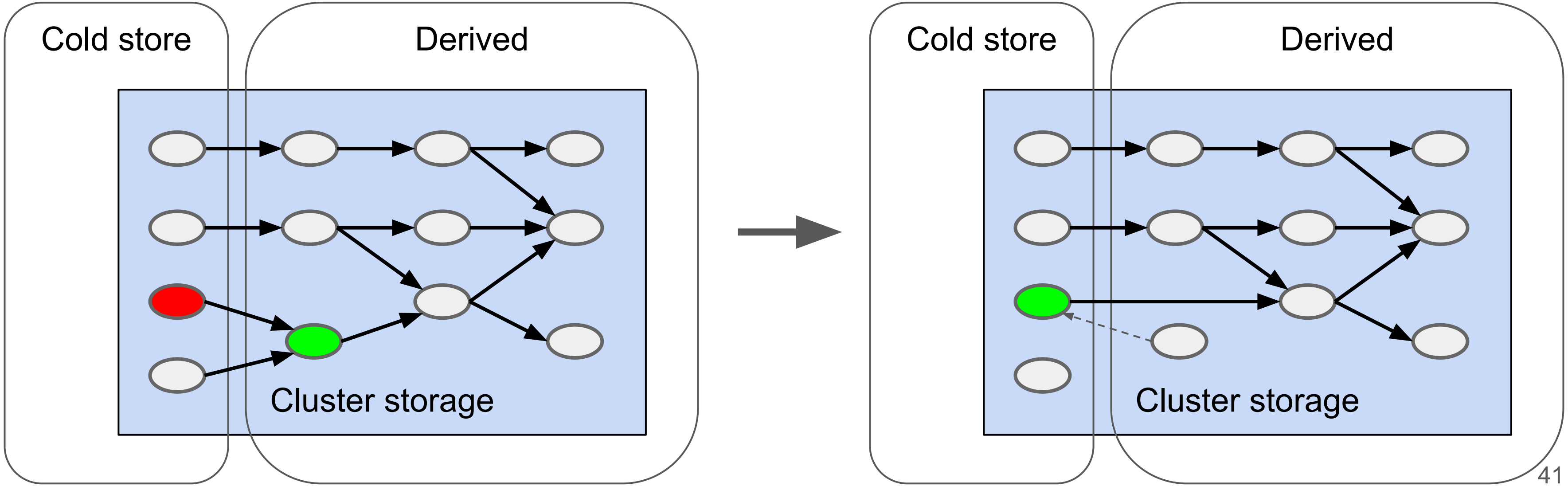
- Publishing pseudonymised datasets
  - They can be de-pseudonymised with external data
  - E.g. AOL, Netflix, ...

# Retention limitation

- Best solved in workflow orchestration
  - Creation and destruction live together
- Short default retention
  - Whitelist exceptions with long retention
- In conflict with technical ideal of immutable raw data

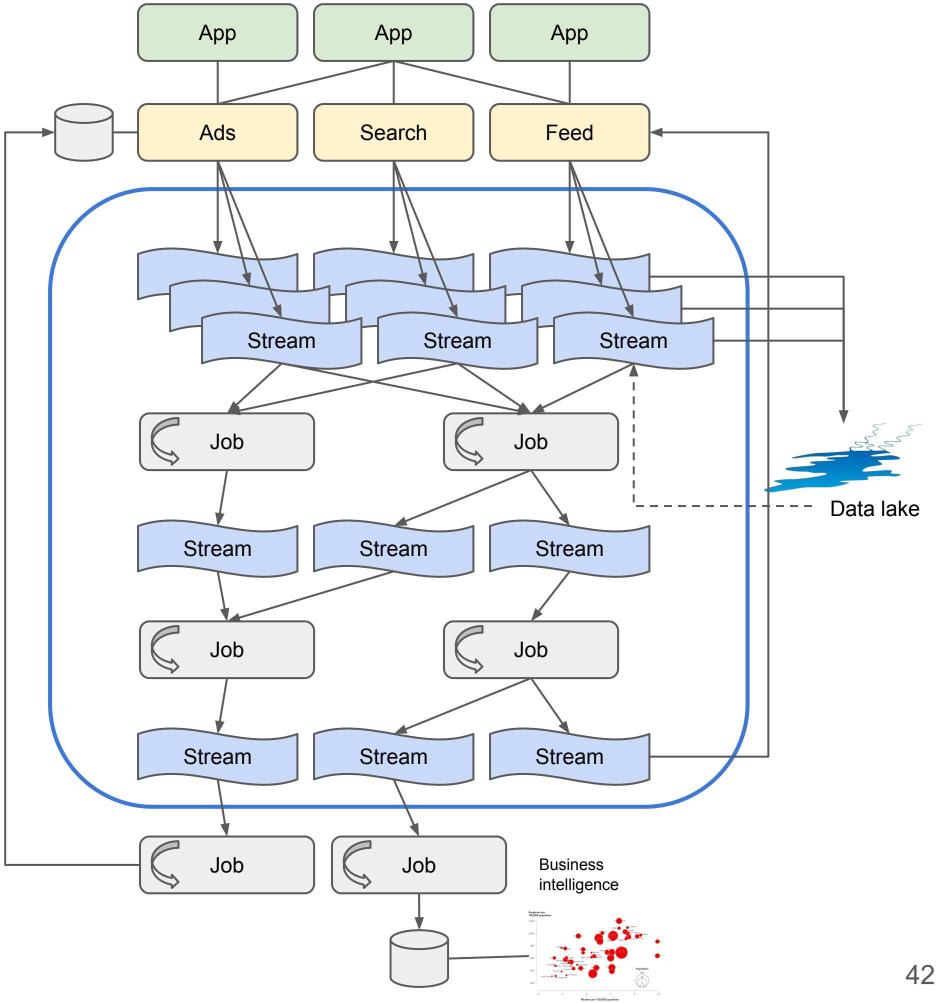
# Lake freeze

- Remove expire raw dataset, freeze derived datasets
- Workflow DAG still works



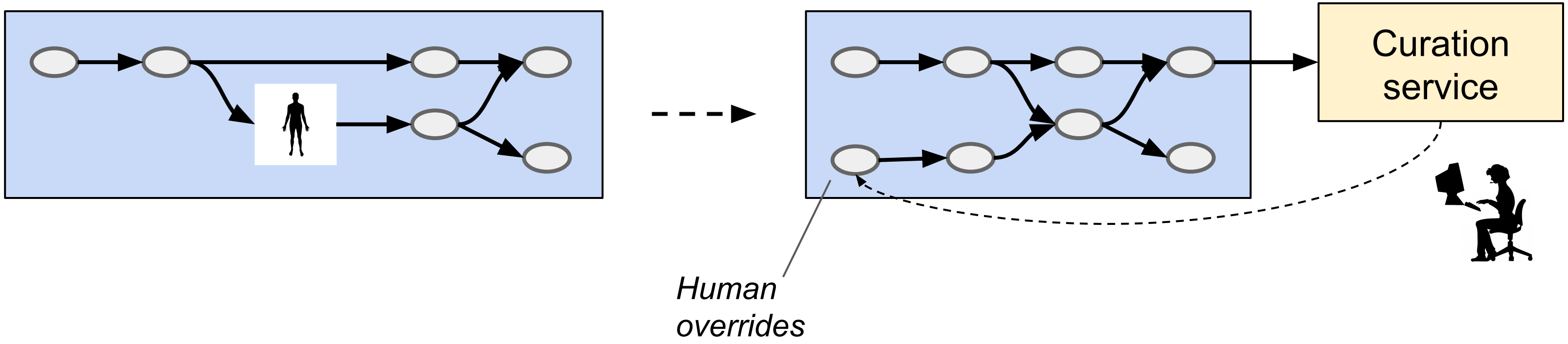
# What about streaming?

- Unified log - bus of all business events
  - Streams = infinite datasets
- Pipelines with stream processing jobs
  - Governance & reprocessing difficult
- Ejected record & lost key patterns work
  - PII or encryption key in database table
- Retention is naturally limited



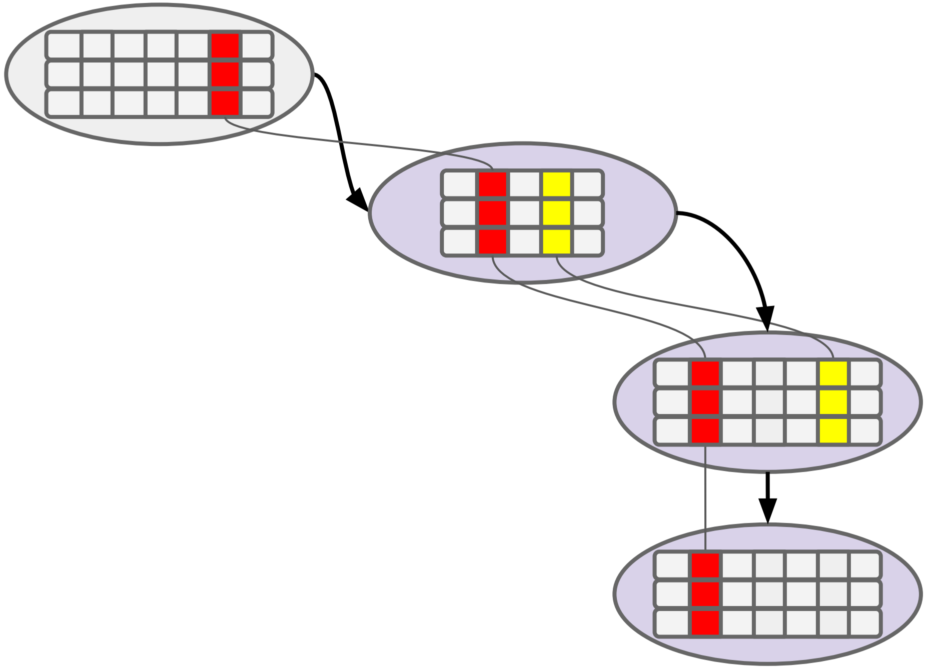
# Correcting invalid data = human in the loop

- Humans are lousy data processors
  - Expensive to execute
  - Not completely deterministic
  - Not ready to kick off at 2 am
  - Don't read Avro very well
  - Not compatible with CI/CD
- Add human curation to cold store
  - Pipeline job merges human curation input
  - Overrides data from other sources



# Lineage

- Tooling for tracking data flow
- Dataset granularity
  - Workflow manager?
- Field granularity
  - Framework instrumentation?
- Multiple use cases
  - (Discovering data)
  - (Pipeline change management)
  - Detecting dead end data flows
  - Right to export data
  - Explanation of model decisions



# Resources

- https://www.slideshare.net/lallea/protecting-privacy-in-practice
- http://www.slideshare.net/lallea/data-pipelines-from-zero-to-solid
- http://www.mapflat.com/lands/resources/reading-list
- https://ico.org.uk/
- EU Article 29 Working Party
- ENISA: "Privacy by design in big data"
- GDPR-podden

# Credits

- Alexander Kjeldaas, independent
- Lena Sundin, independent
- Oscar Söderlund, Spotify
- Oskar Löthberg, Spotify
- Sofia Edvardsen, Sharp Cookie Advisors
- Øyvind Løkling, Schibsted Media Group
- Enno Runne, Baymarkets