Taking Control of your Trading System

Pete Windle – Distinguished Engineer, Managing Director

HSBC

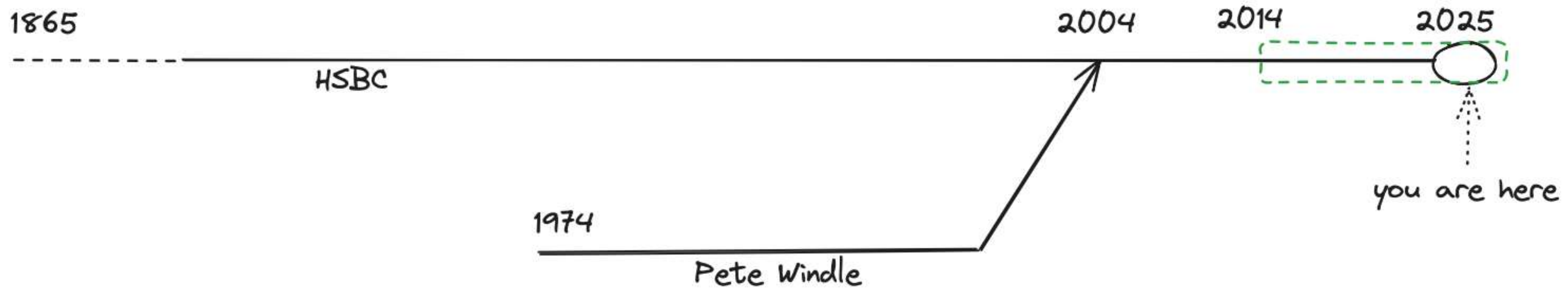
Taking Control Of Your Trading System

- I want to talk about the success we had replacing a vendor monolith with an in-house build
- As this is XT25 – and therefore by definition everyone loves Clojure and temporal databases:
  - I want to talk about some of the technical choices we made around our Trade Repository subsystem
  - Describe a few high-level patterns we found helpful along the way
- Speedrun each of those in 10 mins to leave time for Q&A?
  - Let's gooooooooooo!

History

HSBC

History



What's a "Primary Trading System"?

- Golden source of trade data for one or more businesses (a "trade repository")
- A mechanism for taking trades and booking them in the right place
- Intraday post-trade risk management functionality
- End of day risk and P&L reporting for both internal and external consumption
- …usually incorporating reference data management, upstream and downstream system integrations, etc

Reasons to buy from a vendor (2003)

- We had a strategic vision to grow HSBC's Global Markets presence
- New trading heads found that they needed more from their tech
- Global solutions for single businesses replacing local solutions for multiple businesses
- Vendors stepped in to plug the gap
- Turnkey solutions that support all products you might trade, allow you to master your reference data in one place, use it for your cross asset hedges, even use it front to back
- Every business found a vendor that excelled in their product

Mo’ Money, Mo’ Problems

- Businesses scaled up dramatically, in terms of both turnover and ticket count
- Vendor systems which might be an excellent fit for a smaller shop began to creak at HSBC scale, and opportunities were missed
- Problems of scale resulted in significant production outages
- Upgrades of these complex and monolithic architectures are costly, risky, and time consuming
- This binds you to old stacks with hardware obsolescence and vulnerabilities…

RIVER

HSBC

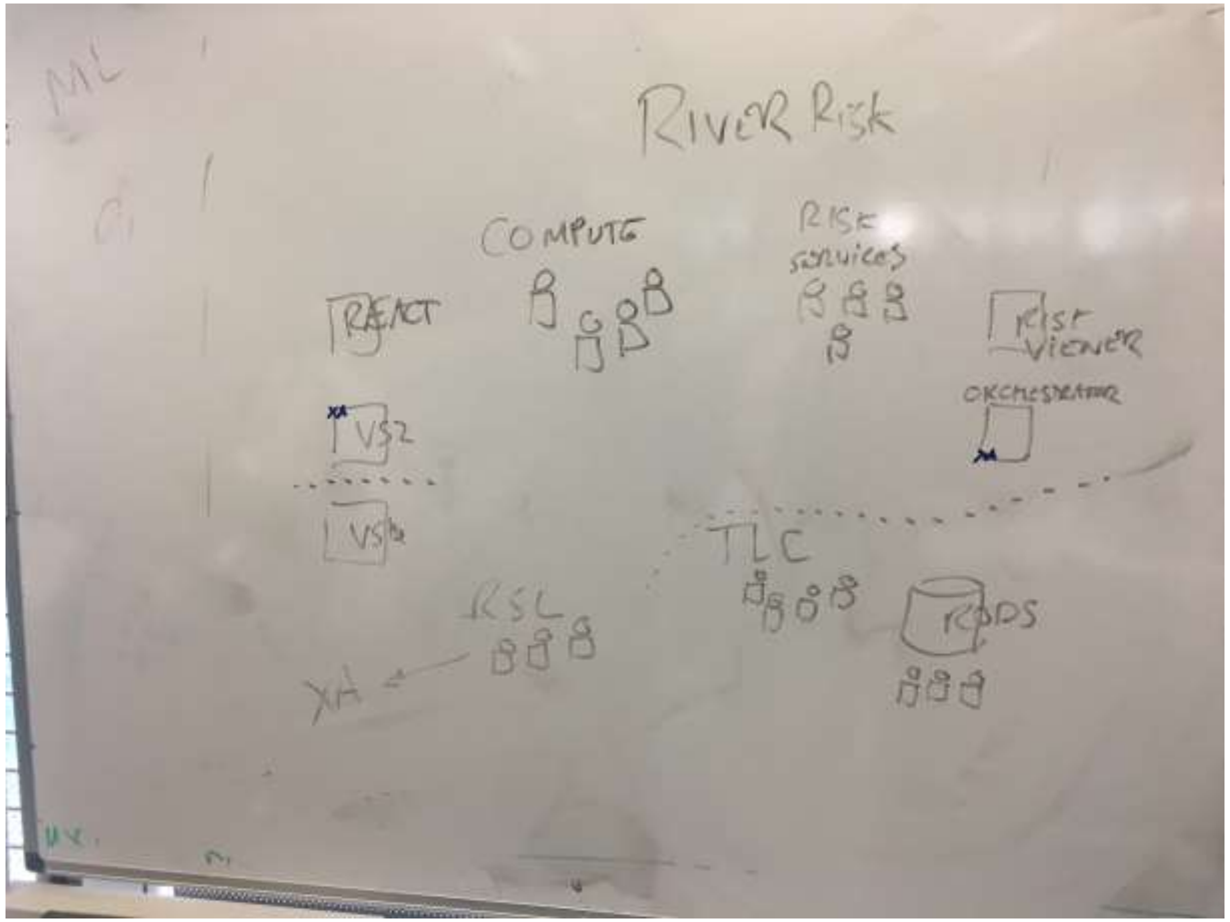
A New Strategic Vision (2014)

- Global FX was investing in technology for growth
- We had a stark choice:
  - Start a new build alternative to the vendor solution
  - Commit to a large and complex upgrade of incumbent

Why take the risk?

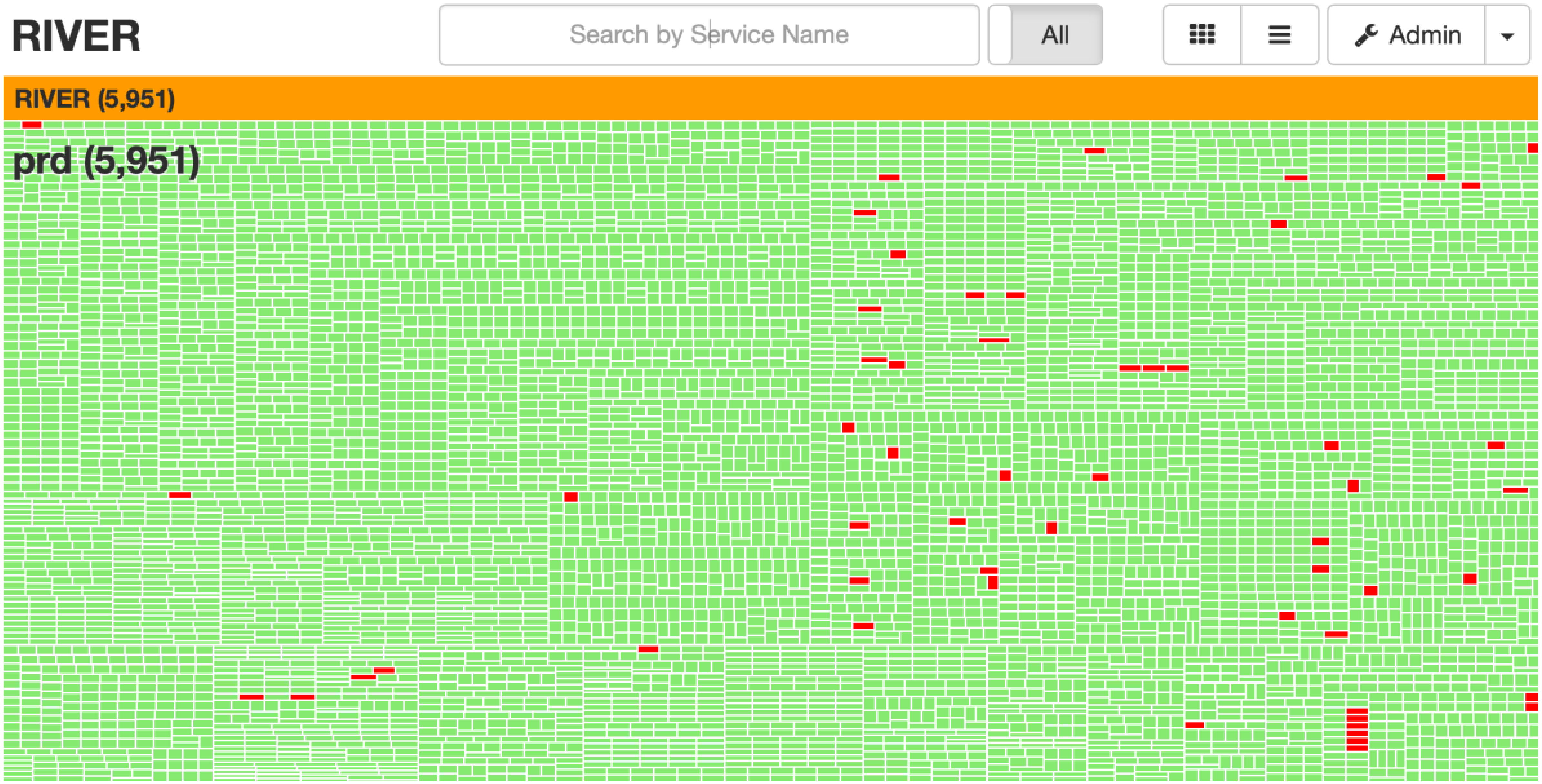
- Credible IT team ready for the challenge
- Vendor dependency had proven issues:
  - Cost and risk of vendor upgrades
  - Cadence of vendor updates (yearly vs 10x a day)
  - Scaling issues
  - Niche expensive customization skills
  - Single point of contention
- Operational resilience through transparency
- Software shaped for our business, not generic off-the-peg
- *If not now, then when?*

A New Strategic Vision (2017)

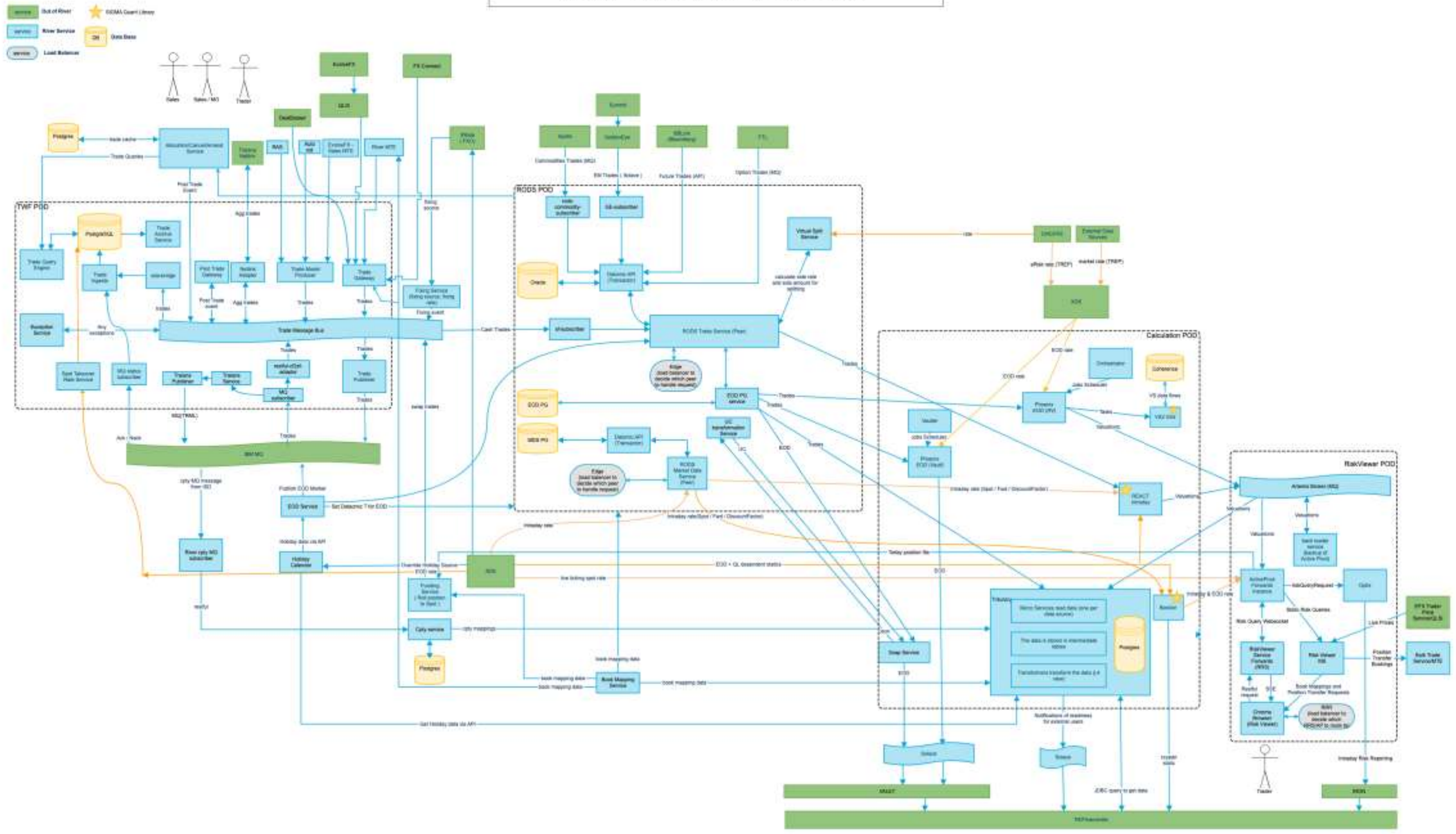


RIVER

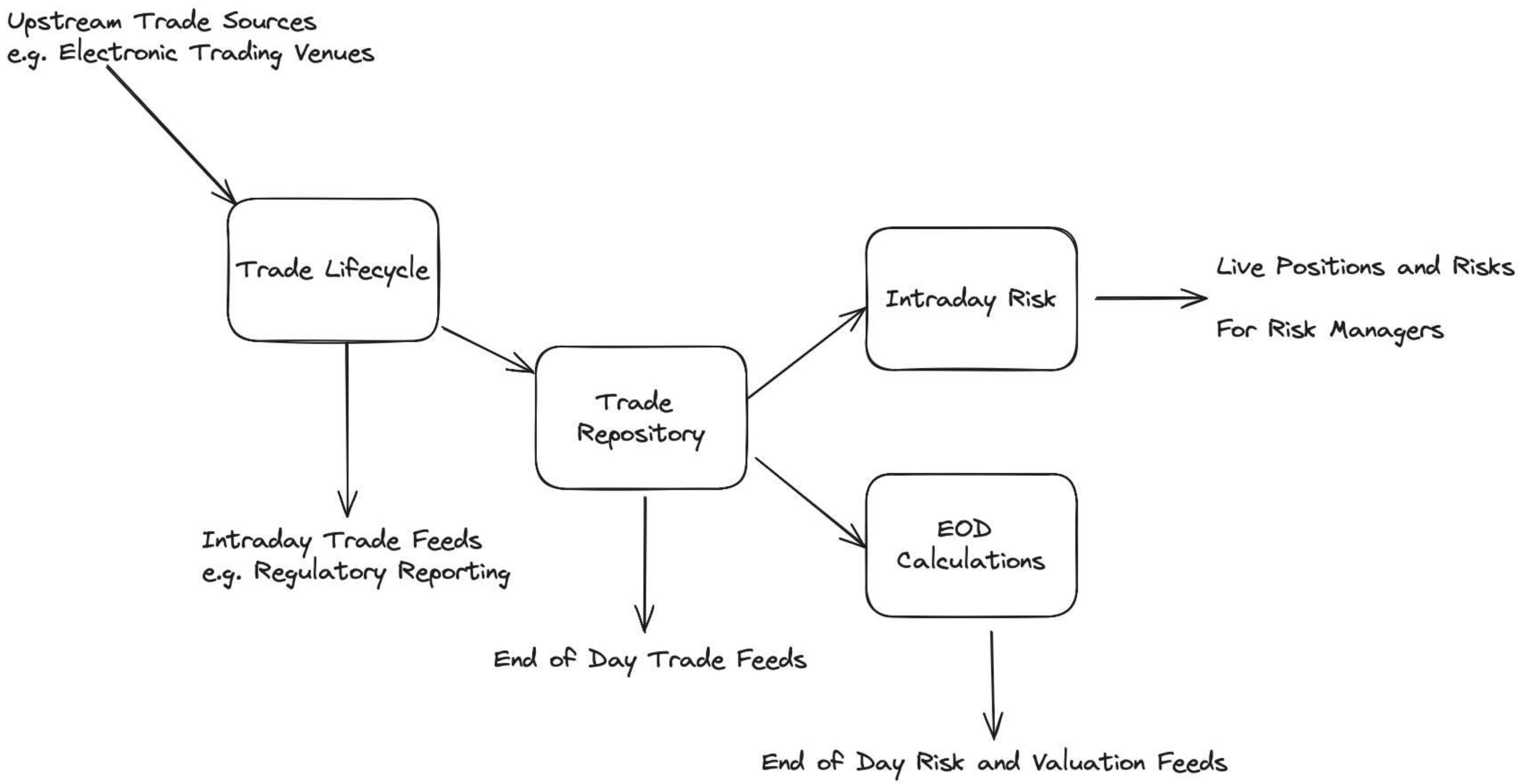
- (**R**eal-time **I**ntegrated **V**iewers, **E**ngines and **R**isk)
- Single global instance
- Composed of decoupled services
- Themselves composed of many microservices


RIVER
Search by Service Name
All
Admin
RIVER (5,951)
prd (5,951)

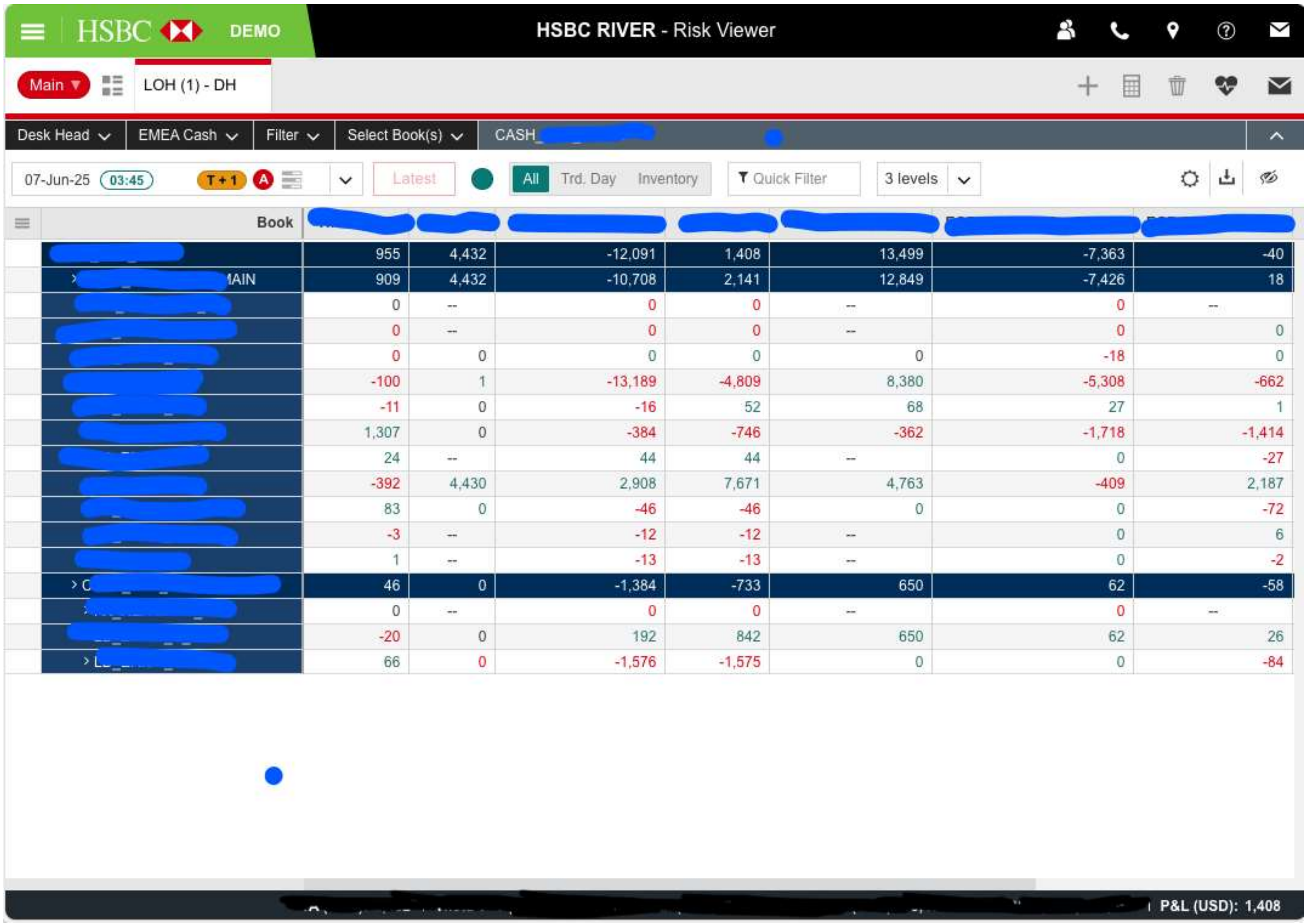
RIVER High Level Architecture



INTERNAL


Upstream Trade Sources
e.g. Electronic Trading Venues
Trade Lifecycle
Intraday Trade Feeds
e.g. Regulatory Reporting
Trade Repository
End of Day Trade Feeds
Intraday Risk
Live Positions and Risks
For Risk Managers
EOD Calculations
End of Day Risk and Valuation Feeds

RIVER



HSBC RIVER - Risk Viewer

LOH (1) - DH

CASH_[redacted]

07-Jun-25 03:45 T+1 A

3 levels

Book	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	955	4,432	-12,091	1,408	13,499	-7,363	-40
› [redacted]	909	4,432	-10,708	2,141	12,849	-7,426	18
[redacted]	0	--	0	0	--	0	--
[redacted]	0	--	0	0	--	0	0
[redacted]	0	0	0	0	0	-18	0
[redacted]	-100	1	-13,189	-4,809	8,380	-5,308	-662
[redacted]	-11	0	-16	52	68	27	1
[redacted]	1,307	0	-384	-746	-362	-1,718	-1,414
[redacted]	24	--	44	44	--	0	-27
[redacted]	-392	4,430	2,908	7,671	4,763	-409	2,187
[redacted]	83	0	-46	-46	0	0	-72
[redacted]	-3	--	-12	-12	--	0	6
[redacted]	1	--	-13	-13	--	0	-2
› [redacted]	46	0	-1,384	-733	650	62	-58
[redacted]	0	--	0	0	--	0	--
[redacted]	-20	0	192	842	650	62	26
› [redacted]	66	0	-1,576	-1,575	0	0	-84

[redacted] P&L (USD): 1,408

Trade Repository

HSBC

Trade Repository

- One of the hardest tasks is creating a full-fidelity trade repository
- Without trades, you have no risk
- Without trades, you have no end of day reporting
- Without trades, you don't have a trading system
- This needs to be 100pc accurate – an error in one trade can potentially generate false market risk, and traders can make expensive hedging mistakes based on your error
- It also needs to be **consistent** – the ability for multiple services to read the same data is essential to a distributed system

Trade Repository

RODS

Realtime Operational Data Store

Innovation Tokens

- Several incumbent attempts to provide some degree of trade coverage were evaluated and none of them were extensible to the task at hand
- Schemaless document databases gave too little structure
- Traditional RDBMS schemas had high barriers to entry at that time (DDL gated through infrastructure DBA teams, etc – an anti-pattern we've since fixed)
- In the end we started a trial with Datomic from Cognitect
- Hedging our availability and DR by using Oracle RAC as storage

Innovation Tokens

- We already had a little experience using Clojure for tools
- We decided to trial it for RODS playing to its strengths in data manipulation

- We carefully drew an API boundary around RODS to ensure that these technology innovations did not leak beyond our risk appetite

Why Datomic?

- Attribute level schema means data is reliably typed
- Flexibility of attaching attributes to entities supports complex modelling where necessary, at no cost to simpler products
- Temporal capability / Permanent transaction log
    - Allows repeatable reads – launch a number of processes with consistency guarantees
    - Support streaming of transaction data in an event-sourcing style
    - No need for clunky shadow tables – full audit history is available (back to 2018 at present)
- Transactor coalesces many application transactions into a single Oracle transaction

Operating at our scale is hard

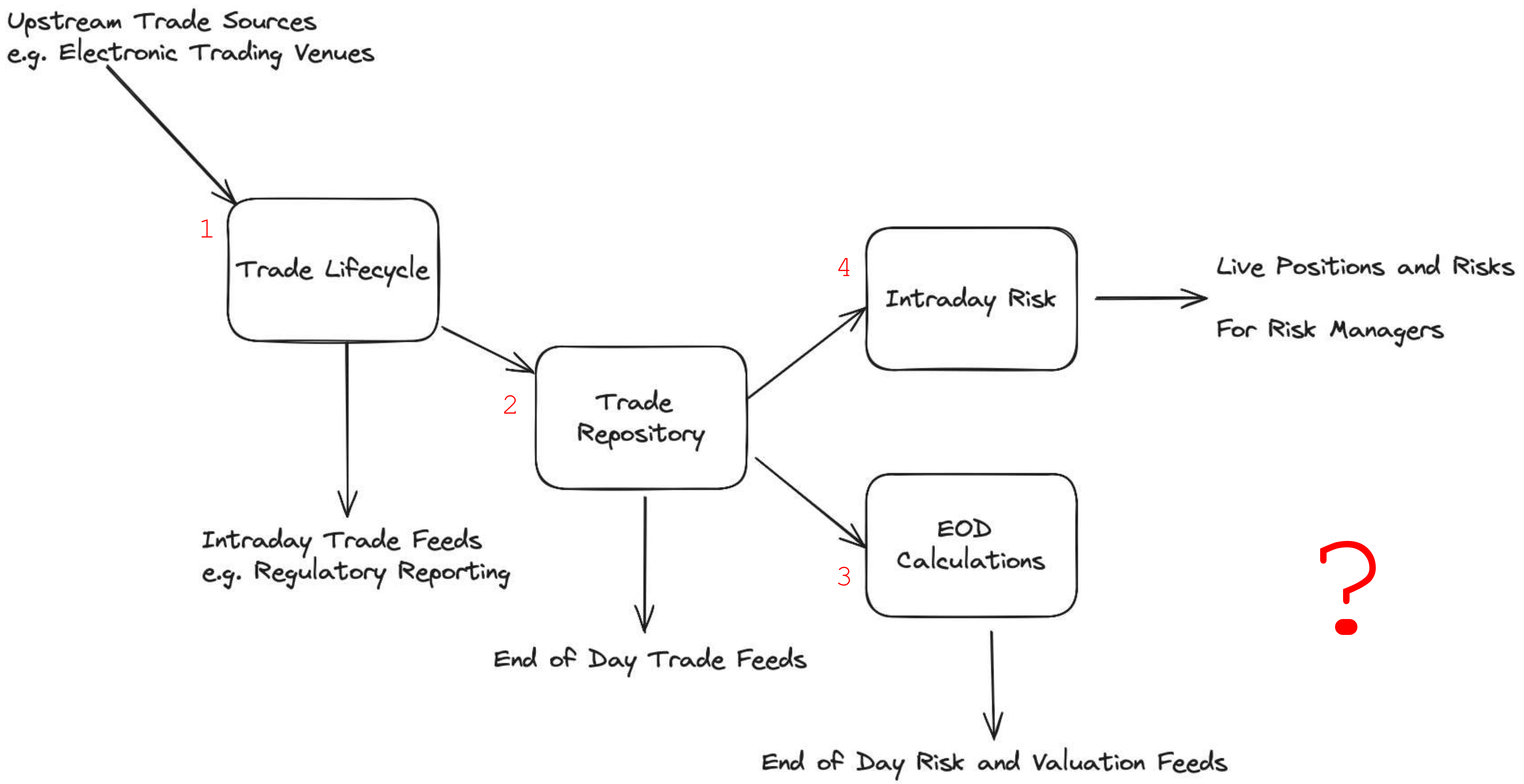
- We might process millions of events a day
- With millions of open trades and cash positions
- With new trades entering and old trades settling constantly
- Multiple businesses in one instance – FX Cash, FX Options, Commodities, STIR, EM Rates…

Patterns

HSBC

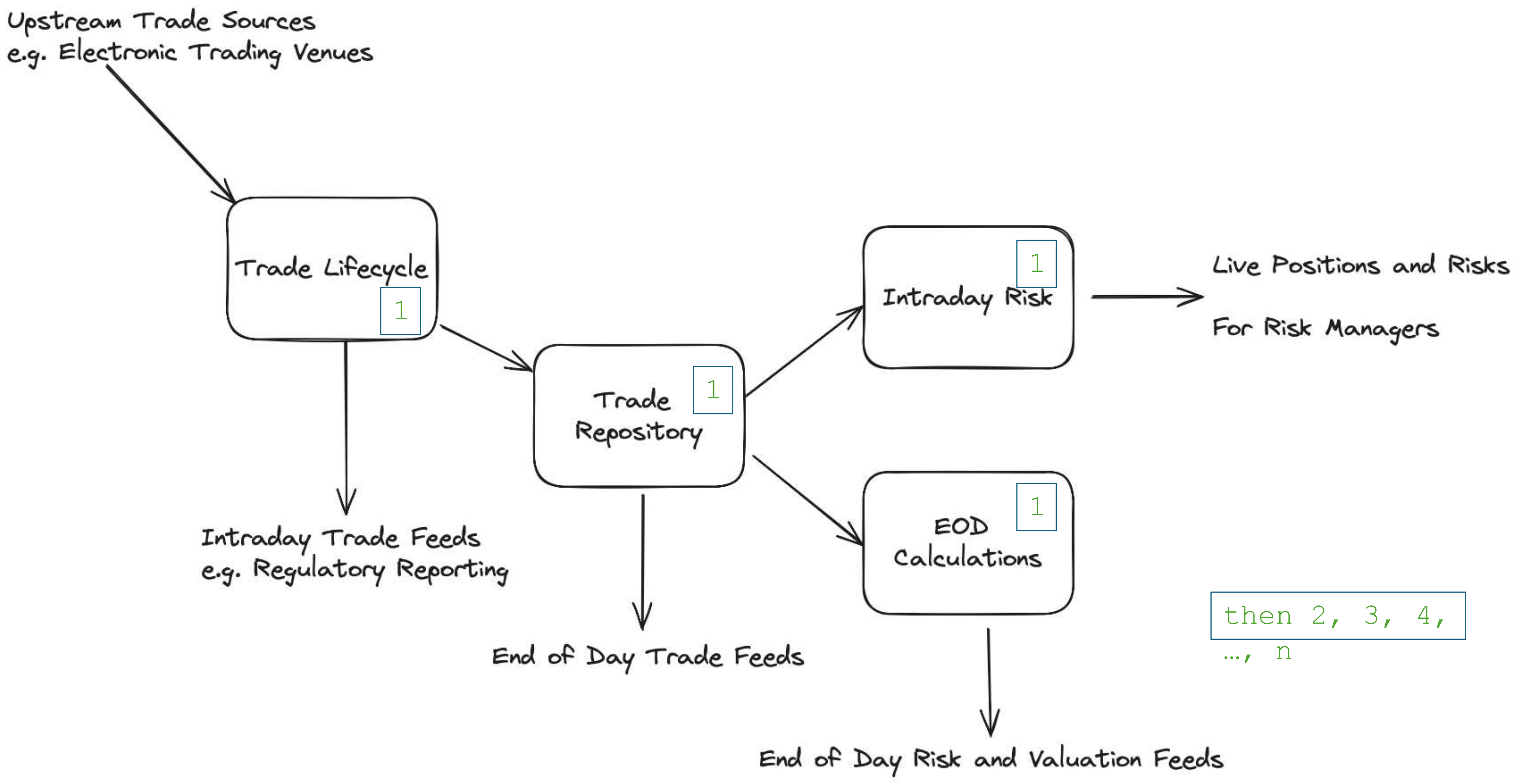
Strangler Pattern

- Run the trade repo next to your live system
- Seed it with an initial population of live trades
- Hook it up to your stream of events
- Ensure appropriate running reconciliations are in place
- Sort out your data governance and be accepted as an authorized distributor of trade data
- Start running end of day risk off system in parallel
- Build regression tooling to get everyone happy
- Start running intraday risk in parallel
- Persuade your trading desk to migrate
- You're done!


Upstream Trade Sources
e.g. Electronic Trading Venues
1
Trade Lifecycle
Intraday Trade Feeds
e.g. Regulatory Reporting
2
Trade Repository
End of Day Trade Feeds
4
Intraday Risk
Live Positions and Risks
For Risk Managers
3
EOD Calculations
End of Day Risk and Valuation Feeds
?

Strangler Pattern

- That was easy?
- 100pc fidelity **for the usecase at hand** is table stakes for a trade repo
- Therefore:
  - break off small usecases that do not require you to fix every business or product at once
  - find quick wins – if half the work has already been done in options, but you’re targeting cash first… find an options usecase
  - you want to minimize your path to production
  - and be iterating in production on software that is being actually used


Upstream Trade Sources
e.g. Electronic Trading Venues
Trade Lifecycle
1
Trade Repository
1
Intraday Risk
1
EOD Calculations
1
Live Positions and Risks
For Risk Managers
Intraday Trade Feeds
e.g. Regulatory Reporting
End of Day Trade Feeds
End of Day Risk and Valuation Feeds
then 2, 3, 4, …, n

Trade IDs

- Problems I have personally seen
  - Running out of numbers in database (housekeeping needed)
  - Running out of numbers in code (java int)
  - Downstream fixed-width files running out of space for numbers
  - …ad nauseum

Trade IDs

- Use longs, of course
- We had the inspiration to run trade IDs backwards
- Earliest live trade right now is `999,999,999,997,315`
- Latest is `999,996,045,892,579`
- Estimate that we'll run out of trade IDs in around 2 million years
- Trades can be referred to without the 9 prefix

- FAQ: Why 15 digits when 2^63 has 18 digits?

Query Engine

- The inbuilt Datomic Datalog query engine did not meet some of our non-functional requirements
- We kept the query language from previous trade repos
- We needed something capable of responding
    - quickly for specific queries ("give me this trade by id, give me all the live trades in this book settling today")
    - efficiently for bulk general queries ("give me the live trades for the UK entity")
    - running against the transaction stream to give clients filtered events
- Built a streaming query engine atop Datomic's indexes directly

Query Engine

- This was quite hard to tune and get performant
- A chain of eduction is conceptually very clean but also hard to profile accurately!
- Metrics to understand segment loading behaviour
- Even when the query can quickly identify trades, for bulk usecases such as end of day we introduced PostgreSQL based document caches
  - even though our direct Datomic queries are pretty fast, having to decompress attributes from EAVT index then form them into JSON is about 3-4x slower than having the text sitting in a JSONB column on disc

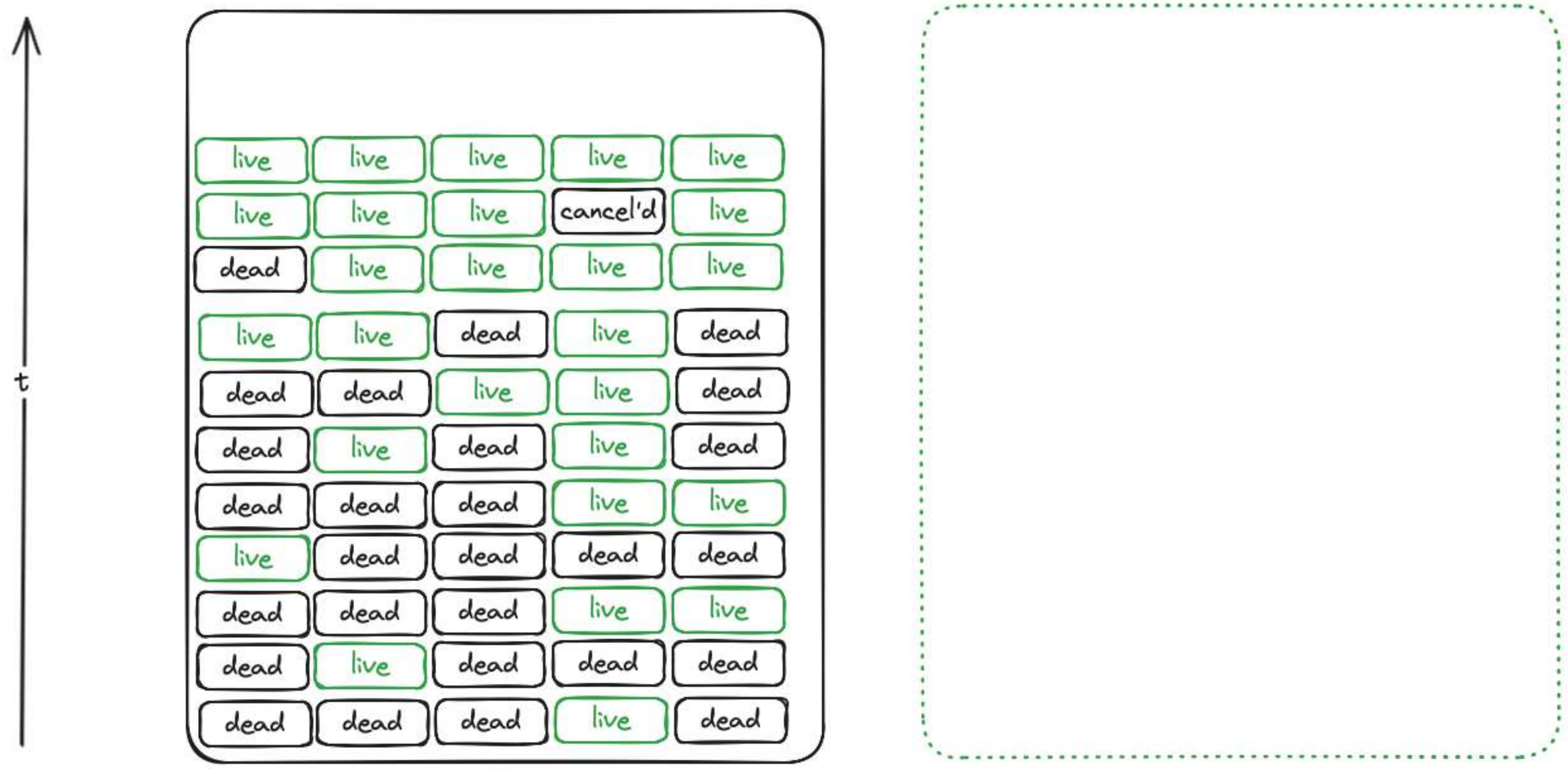
Heap Sizes and JVM

- We run our Datomic Peers with 128Gb heaps
- This allows the open trade population to live in the object cache
- We do keep a Memcached layer which allows us to rehydrate a restarting process quickly without hitting the Oracle DB
  - Pre-warming object cache deliberately
- We separate deep historic queries away from the live query processor to avoid filling the cache with spam from dead trades
- We use Azul Prime for dependable GC behaviour, although continue to evaluate OpenJDK's ZGC and other alternatives

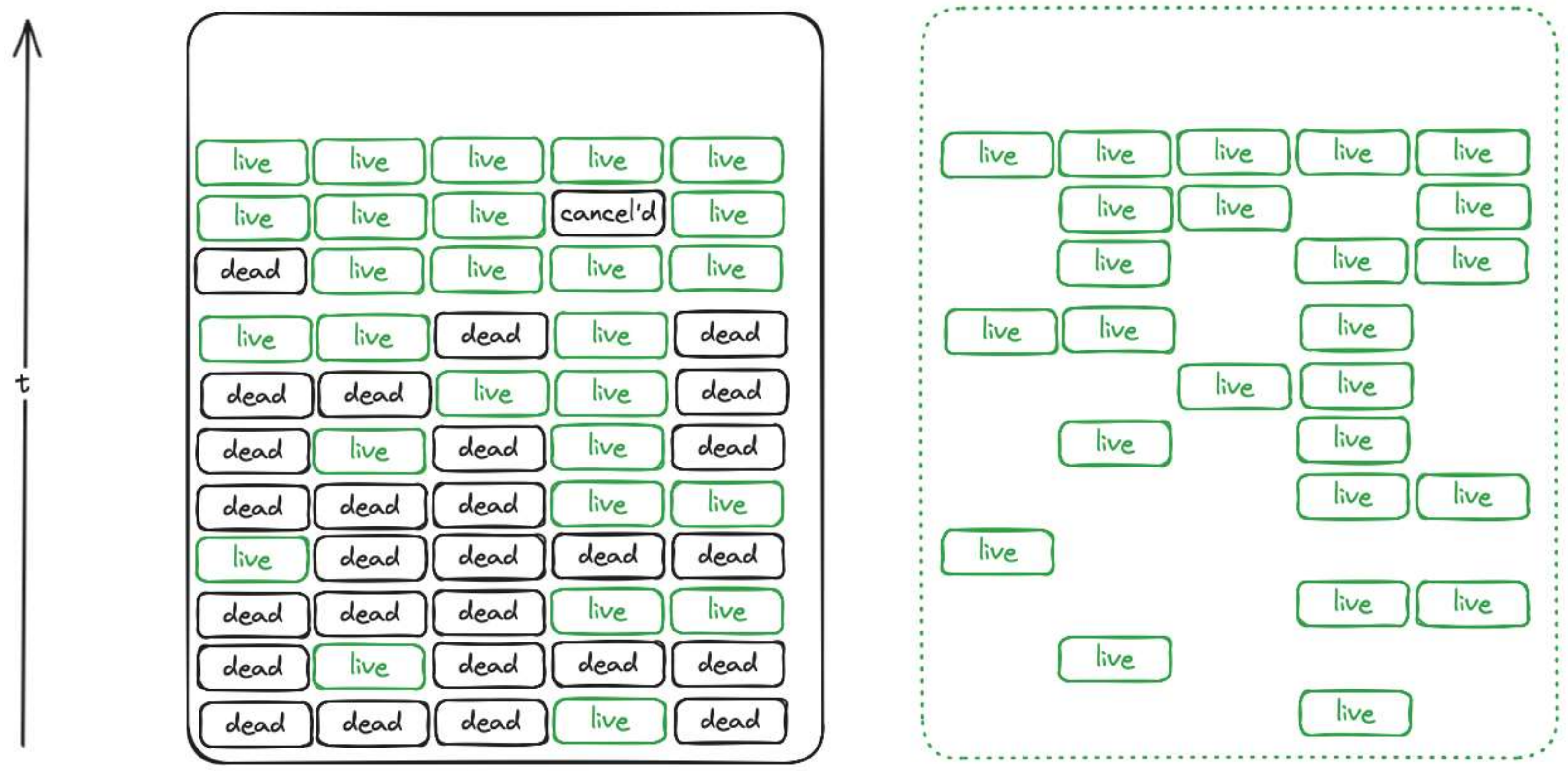
Decanting and Timesharding

- Datomic has soft limits on the amount of data you can sensibly hold in one database
- As time went on we found that queries became slower
  - We'd be asserting perhaps 50-100 million datoms/day
  - IIRC somewhere in the tens of billions the B-Tree acquires a new level
  - …and performance degrades accordingly
- We created a simple solution

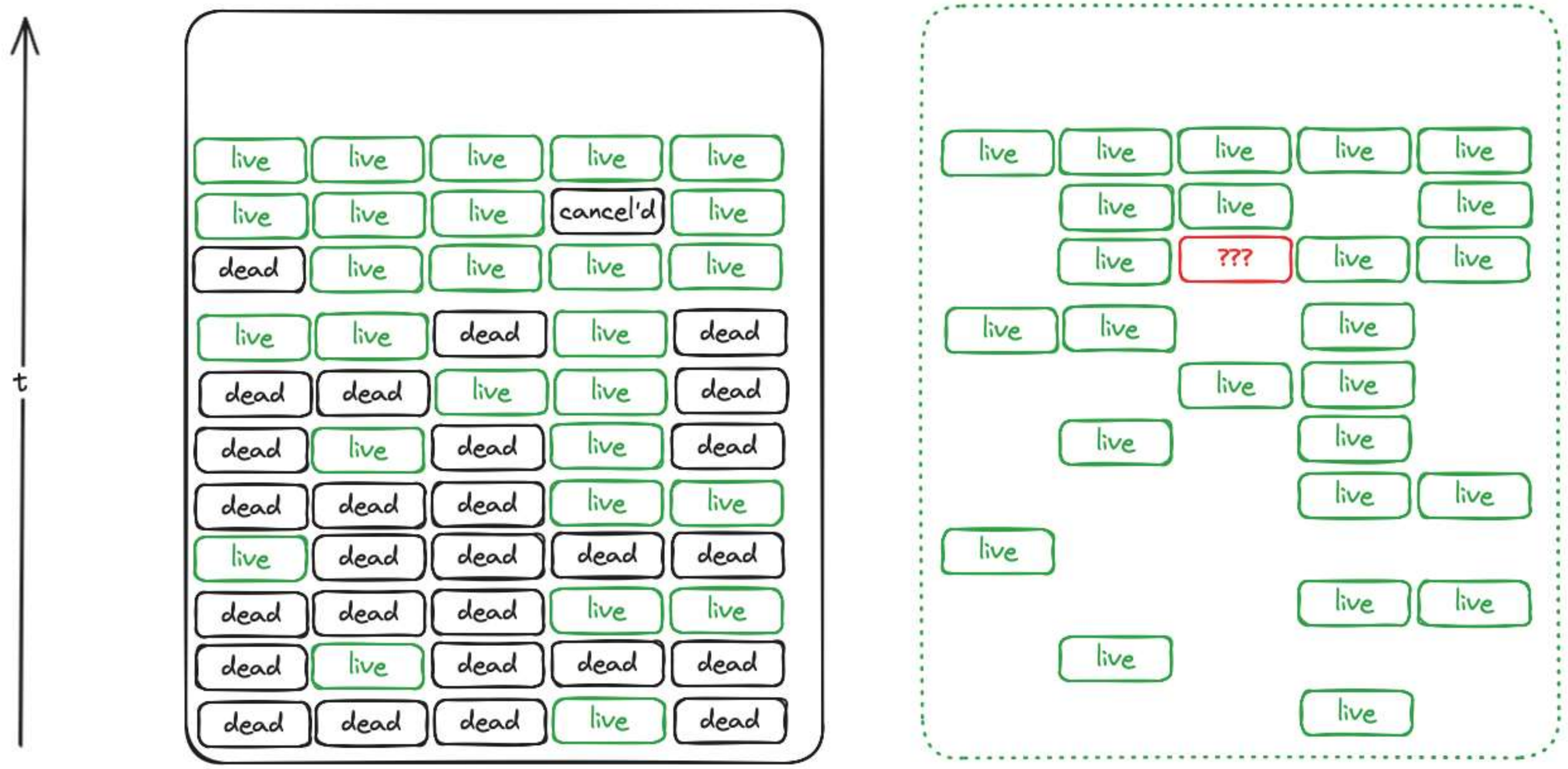
Decanting and Timesharding



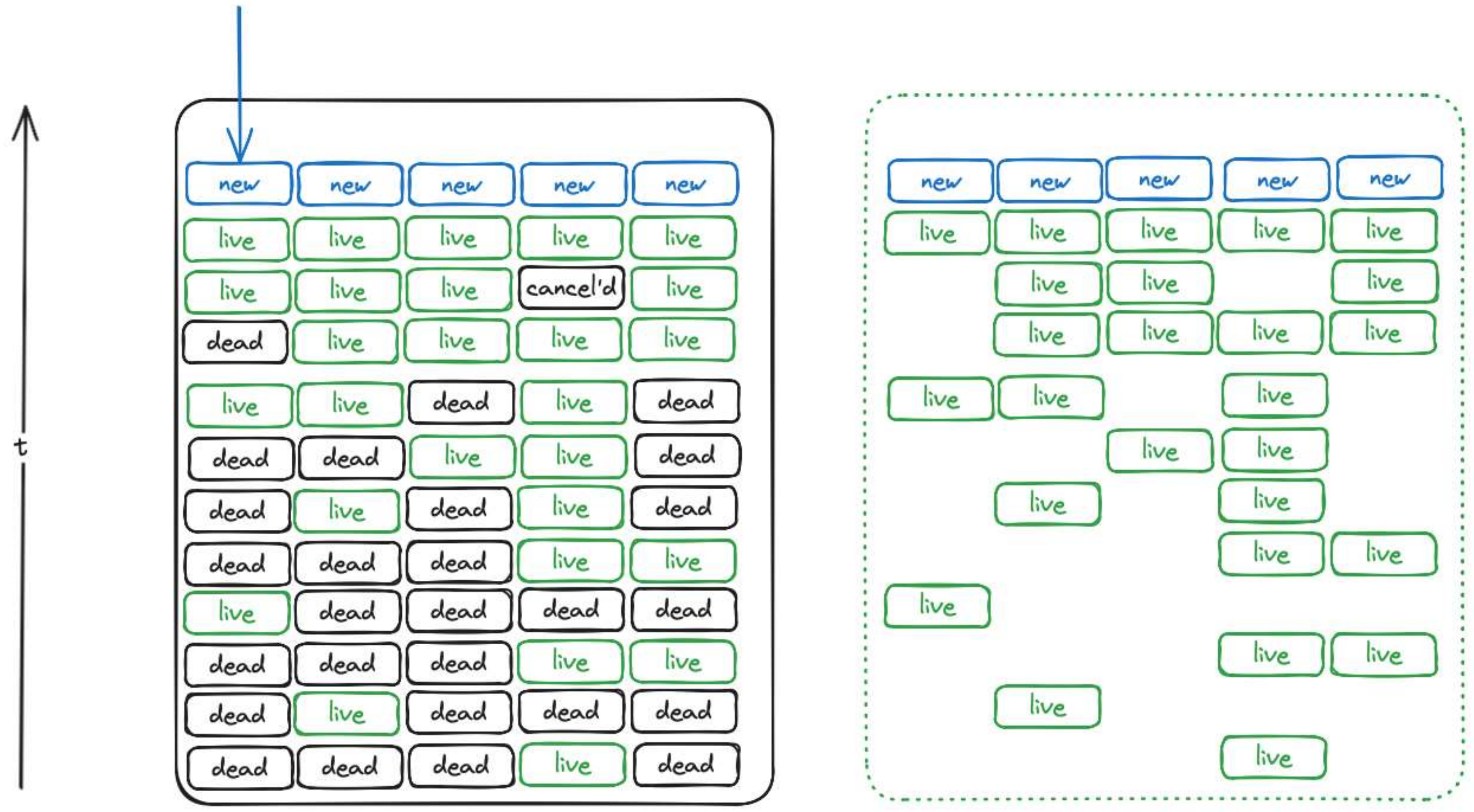
Decanting and Timesharding



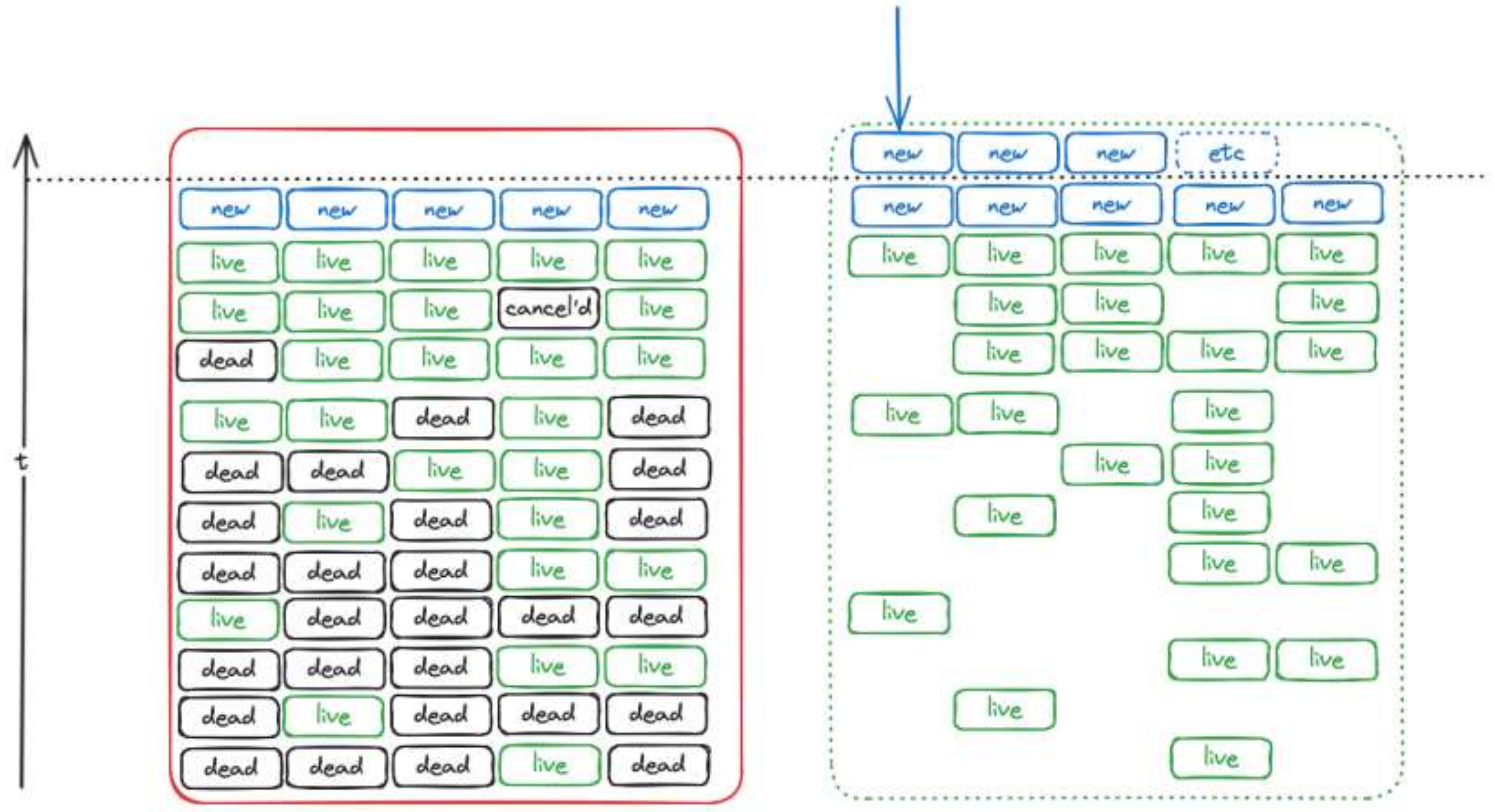
Decanting and Timesharding



Decanting and Timesharding



Decanting and Timesharding



Decanting and Timesharding

- Open trade population is copied from the live database into a fresh database
- This is kept synchronised by reading the event stream from the live database and therefore applying writes to both databases
- A reconciliation process takes place to ensure that the new shard is consistent and complete
- Then the roles are shifted, the new database becomes the "live" database (as of a specific datomic-t) and the "old live" database becomes another timeshard
- When you do a historic as-of query, the API uses your timestamp is used to work out which back-end shard to address

The End

HSBC

Conclusion

- HSBC runs its own primary trading system “RIVER” for Global FX
- It continues to enable the FX business to provide clients with the best experience and pricing, whilst managing the bank’s risk responsibly and effectively
- Splitting the problem into multiple pieces with clear responsibilities allowed us to scale **the team** horizontally
- All technology choices have their rough edges if you work them hard enough, that’s engineering!

Coda

- Catch me in the hallway track if you have any questions
- Or if you're shy peter.windle@hsbcib.com

- Thank you to the teams involved globally, especially in Stirling and Guangzhou for RODS