A new benchmark in banking experience.

Using event driven architecture to create inspired customer experiences through real-time interactions and business insights.

EQUAL EXPERTS

Today, more than ever—and whether they realise it or not—customers judge banks on their technology. Why?

Because people want easy and immediate money management. People want their banking experiences to be personalised and relevant, based on real-time interactions and activity. In recognising this, one of Australia's most prominent and pioneering digital banks created a **mobile app to combine event driven architecture with industry-leading user experience.** The formula proved an immediate hit, rapidly accumulating a large customer base, with many deferring from the traditional 'big four'. It was equally powerful for the digital banks' internal processes too. The architecture offered more dependable services overall, through reduced interdependencies. It offered increased flexibility for integrations, and incredibly powerful automated data-logging for regulatory purposes. And, it enabled the digital bank to quickly deploy new features and differentiators for customers. Here's how event driven architecture can play a crucial role in delivering new benchmarks for forward-thinking banks.

This case study will help you to understand:

How good digital services can be launched to customers within days.

Why a Digital Platform must be treated as a product, rather than a project.

The importance of culture in a Digital Platform ecosystem.

Table of contents.

Our digital bank partner burst onto the scene with a desire to do banking differently. In contrast to traditional approaches, they embraced event driven technologies to make things fun, fast, personal, and easy.

Imagine yourself at a restaurant. You finish a delicious meal and the bill comes to your table. You tap your card to pay and start chatting with the waiter about how much you've enjoyed yourself. Before the receipt is printed—and even before you finish the conversation—you receive a detailed transaction notification on your mobile app. It tells you exactly how much you've spent on 'eating out' for the month so far. It also provides extra context about the restaurant you've just paid; with a Google map, opening times, contact information, reviews, and more.

In the few hundredths of a second that pass while this information is collected, categorised and displayed for you, the bank's backend services are busy too.

The bank has received a payload notification from the global payment processor, converted the information into ISO standard format, passed it to the core banking platform to record the transaction, adjusted the bank balance, logged the transaction for reporting and analytics, and completed a whole other host of automated banking or operational processes; all immediately triggered by your initial tap-to-pay.

EQUAL EXPERTS

That initial tap-to-pay could also trigger anything from compiling real-time regulatory reports, to saving a data point that may trigger a future promotion or customer benefit. For example, if you spend less on the 'eating out' category in the next few months, you might get access to a new account feature or customer benefit. It doesn't even matter if that feature or benefit doesn't exist yet; the information required to facilitate it in the future is being automatically stored and can be easily accessed at any point (and for a wide variety of purposes) in the future.

This scenario is possible—along with many, many more like it—thanks to events and event driven architecture.

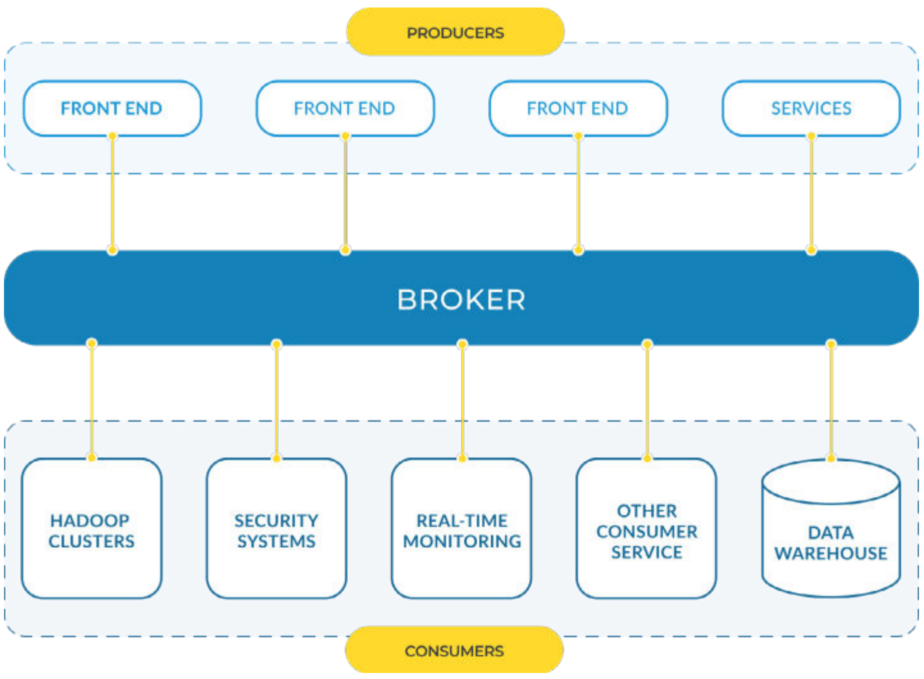
Defining and understanding events

Events are effectively incredibly versatile messages which are generated when certain actions are completed in the digital bank's platform, or in any one of the systems connected to the digital bank's platform.

Events are stored in immutable logs called 'topics' and received by event 'consumers'. An event can trigger subsequent actions or business processes immediately, or can be stored for an indefinite period of time until another combination of events, or catalytic number of events, initiates the designated action.

Examples of events can include:

- A customer profile is created or updated
- An account is created
- A transaction occurs
- A customer applies for a loan
- A customer is approved for a loan
- An eKYC check is completed for identity verification



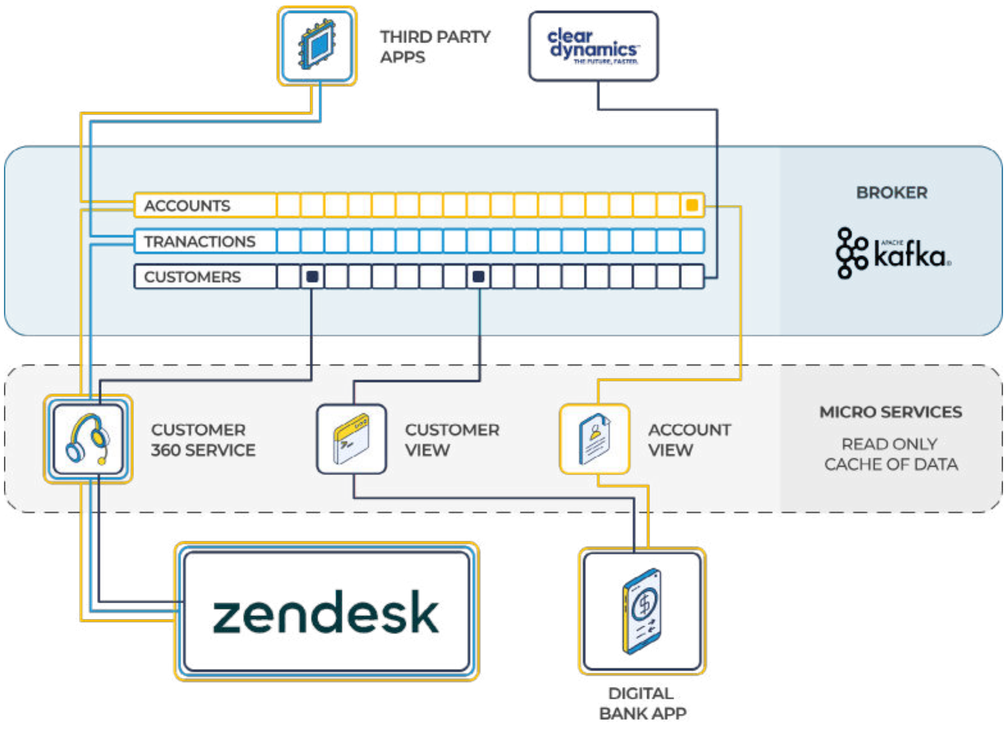
Understanding the basics of events

An event represents a point-in-time state for a specific object of data. Events are ‘streamed’ from systems or microservices that generate them and ‘consumed’ by other systems that want to use them for various business processes. Events are managed by an ‘event backbone’—in the case of our work with this digital bank, Apache Kafka—which manages events in a way that separates the producer and the consumer. Crucially, events can be archived (written to disk storage in the cloud over time, for example) but cannot be updated. Therefore, each topic represents an immutable log of information which can be easily accessed and consumed by different consumers.

As events are created, they are automatically appended to topics, which queue all events from the inception of the topic. Topics are typically defined and structured to reflect key business domains. Example topics from our work with the digital bank include Accounts, Transactions and Customers.

Event producers and consumers can operate completely independently of each other. Additionally, we can configure multiple consumers of the same event, and each consumer can consume events at their own pace, for markedly different purposes.

For example, a customer updating their account information might be logged as an event in the ‘accounts’ topic, but it could also be consumed by a dedicated microservice whose sole task is to check and reconcile potential discrepancies in account information across platforms.



02

Events provide the real-time information you need to deliver the tailored, immediate banking experience that modern customers expect.

Examples from our work with this pioneering digital bank include:

Seamless, speedy, and more meaningful customer service

In partnership with the digital bank, we use events to make customer service fast and highly personalised. Any time a customer calls with an enquiry, service agents can immediately access the relevant customer's accounts, detailed transaction information, cards—whatever details are necessary to their enquiry. This is immediate and automated, and available without the slow processes associated with querying the core-banking platform, or any need to call on an eKYC solution or similar. It's all seamlessly and immediately available via Zendesk, with information populated using events.

This ensures customers don't have to endure the frustration of repeating information to different departments, or across multiple customer service calls.

Events provide a far more complete view of your customers

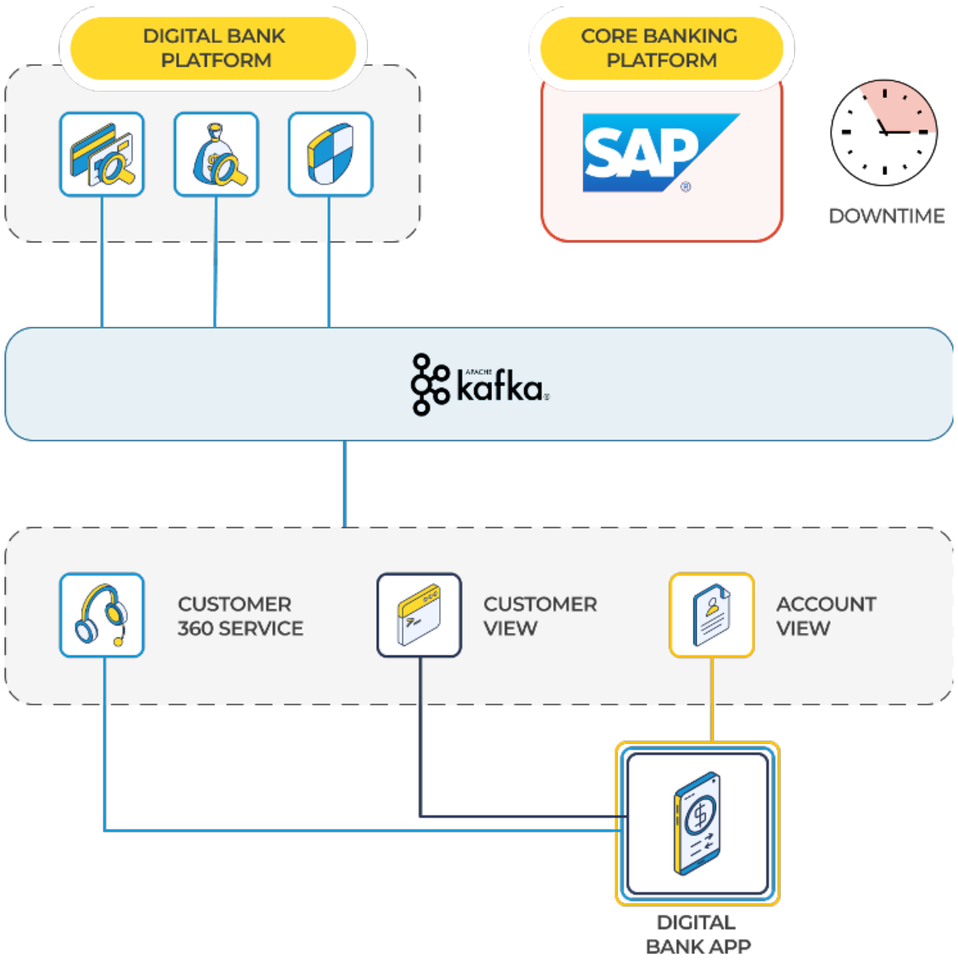
We created a microservice to consume a variety of events from the 'customers', 'transactions', and 'accounts' topics, along with other information like card data and details from the eKYC platform. It brings all of those events in from a variety of sources and builds a cache. The master of these details is always held as an immutable log in the Kafka event backbone.

In more traditional architectures, data is structured and optimised for transaction processing, or reporting, or for reading. One advantage of event streaming is that each event consumer can re-structure the data it receives, to configure information in the way it's needed for a specific process. We can leverage significant efficiencies by tuning this microservice to call on a wide range of different data sources, interpret them as needed, and rapidly surface a full view of each customer.

'Always on' access to key banking functions and features

Frustratingly, many large-scale banks still partake in inconvenient scheduled maintenance; the dreaded "we will be offline to perform updates from 11pm to 3am". With our event driven architecture, the digital bank's core banking platform is entirely decoupled from the customer mobile app. This means **a reduction in dependencies between systems and therefore less risk of outages to service.** From a customer's perspective, it means the app and many of its key functions are effectively 'always on'.

As a result, core banking applications could be taken offline—for downtime or database predictions—without disrupting key customer experiences. Even with the core banking platform offline, people can still use the mobile app to check balances, view detailed transaction histories, lock accounts or cards, and more.



Create opportunities for deeper customer engagements, even before they're designed

Unless they are specifically engineered to retain change logs over time, most monolithic systems only provide a singular, current view of your data as it exists today. With the digital bank's ability to access a full history of events from the inception of a specific topic, we can apply historical data to evolving business considerations.

For example, we might consider building a new loyalty feature into the customer-facing mobile app. In order for the new feature to work, customers may need to match a specific behavioural profile from the past 12 months; perhaps they have to deposit more than $8,000 into the app in a number of consecutive months. With our event driven architecture, we can easily build out a historical data profile of specific customers, feed it into a new service, and begin applying the new customer feature where it's relevant, as it is triggered. The next time a customer processes the necessary number and amount of deposits, they get immediate access to the new feature.

Better understand customer journeys and motivations by reconstructing views of data

In a traditional approach, banks are entirely reliant on the current state of an object. For example, a customer's current account balance. With an event driven architecture, we can reconstruct and view the series of transactions that led to a particular balance, or any other series of sophisticated interactions that lead to a current customer's state. This is incredibly useful for optimising end user experience in customer apps, or in rapidly detecting potentially fraudulent transactions.

The digital bank's only channel is a mobile customer app for day-to-day banking.

The mobile customer banking app interfaces with the digital bank's platform. The platform itself is a microservices, event-driven stack hosted on Amazon Web Services.

In addition to leveraging event streaming for business process and customer experience, the microservices use events to integrate with a variety of third-party solutions. Each microservice is built and deployed on Amazon EKS as docker containers. The core banking platform is SAP.

Tools and third-party integrations

aws
eway
iress
APACHE kafka
kibana
logstash
okta
Prometheus
SAP
ThreatMetrix A LexisNexis Risk Solutions Company
VISA
zendesk

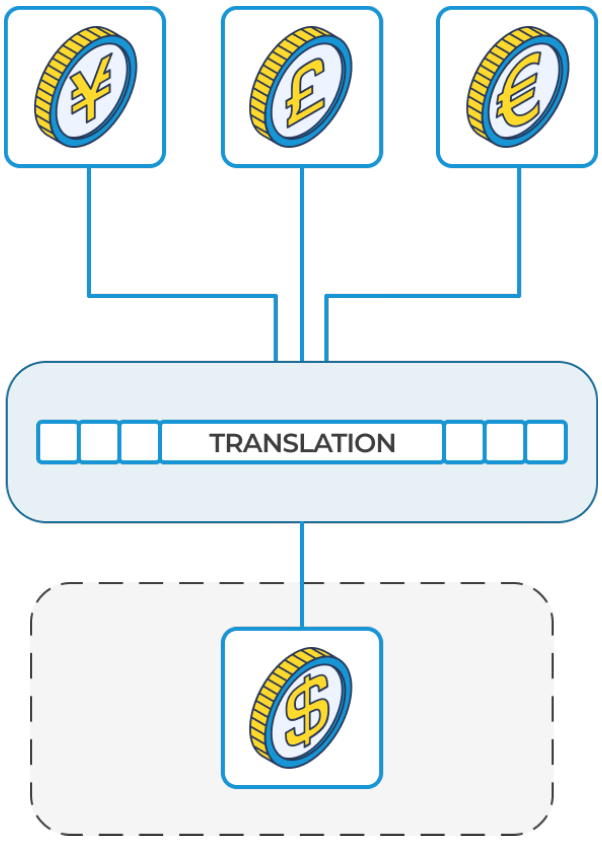
Using events and microservices to prevent 'domain bleeding'

Whenever you integrate with third-party systems, you inevitably come across varying platform interfaces with different requirements for processing data. 'Domain bleeding' occurs when the informational requirements for third-party systems create inconsistencies in labelling or data structure between domains.

In most instances, there is a requirement to preserve a domain-specific language that makes sense for your business. For example, our digital bank platform referred to savings as a 'stash', while the core banking platform—SAP—used the more traditional 'savings account' terminology.

To preserve our preferred data structures and terminologies, we created an event driven microservice with the sole purpose of transforming and restructuring various data attributes while passing information to third-parties.

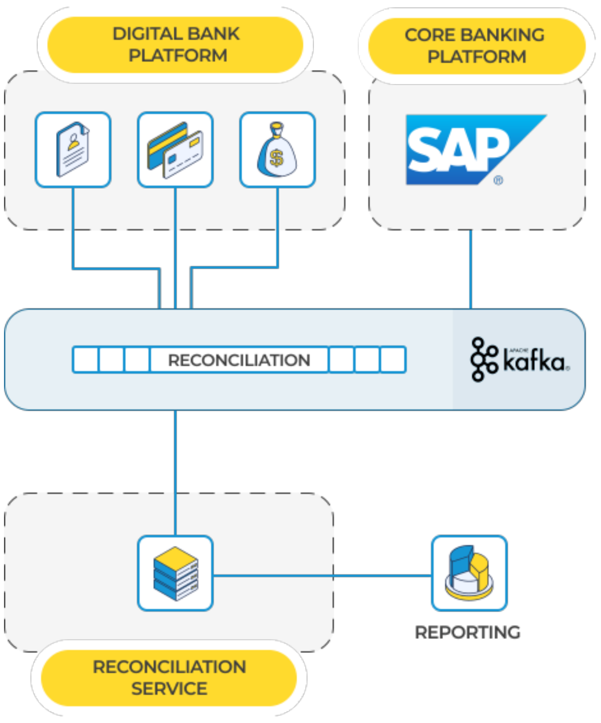
The same approach was used for integration with a third-party loan originations platform, and others. The service would effectively translate information to meet the needs of the various external interfaces we integrated with, while ensuring that everything worked seamlessly in our platform and retained our bank's preferred terminologies and data structures.



Like any Authorised Deposit-Taking Institution (ADI), the digital bank was bound to comply with various regulatory and reporting requirements by a national regulating body.

We structured events to ship data, in real time, for the purposes of automating reports and facilitating internal reconciliation processes.

Rather than manually instigating a batched, end-of-month reporting process—which is common practice—a regulatory reporting service performed the dedicated function of identifying any relevant events in the queue and sending them to a data repository. As a result, various regulatory reports were being updated and compiled from events in real-time, and ready for processing whenever needed.



Using events to automate reconciliation

The team used event streaming to automate reporting in accordance with APRA's financial claims scheme APS 910.

In our architecture, the digital bank's platform was the master for all account, transaction, and balance information. There was a regulatory requirement to perform frequent reconciliations between the digital bank platform and the core banking platform, SAP, to provide assurance that the bank had a clear and consistent view of all customer contact information.

To ensure any changes in the digital bank platform were reflected in the core banking platform, we created event feeds from both. The data from these feeds was pulled using Kafka Connect and written to a dedicated service. This service performed ongoing checks for reconciliation to ensure parity across both customer data sets, and generated reports in accordance with the regulatory requirement.

05

STRATEGIES FOR DATA MANAGEMENT.

With a wealth of incredibly valuable data being generated by events—for reporting, customer experience, and more—cost-effective management becomes crucial.

In our digital bank's architecture, events are stored in the Kafka backbone, before eventually being written to disk storage in the cloud. Today, there are a wide variety of convenient and highly configurable managed service options available for offline event stream storage.

Ultimately, the data generated by an organisation—and particularly a bank, digital bank or fintech—is one of its most valuable assets. In order to continue delivering innovative products and services to customers, in the way they expect, the opportunities presented by event-based architectures are crucial.

Want to know more?

Are you interested in this project?
Or do you have one just like it?
Get in touch. We'd love to tell you more about it.