The truth about cloud native core banking

Thought Machine | Rethinking

Introduction

Authors:

Fabian Siddiqi
Director of Cloud Engineering
Thought Machine
fabian@thoughtmachine.net

Brian Dempsey
Director of Partnerships, North America
Thought Machine
bdempsey@thoughtmachine.net

Robin Zhang
Director of Marketing, North America
Thought Machine
rzhang@thoughtmachine.net

The banking industry is experiencing seismic change, as the advance of cloud technology, rapidly changing customer expectations and increased competition apply pressure on traditional banking models. Meanwhile, more and more customers are moving their financial transactions online – a trend that has been further precipitated by the pandemic. This generational shift in banking has impacts across all channels and services: physical branch, online, mobile, retail, and commercial banking services are all under pressure to deliver and innovate in an increasingly competitive environment. A fundamental change in consumer behaviour is setting new expectations of banking products and services. Customers are demanding ease of use, 24x7 availability, real-time data and analytics, exceptional customer support, full visibility of all banking products, and more.

In the first instalment of this whitepaper series, *Transform banking. Transform core.*, we drew the conclusion that banks must act immediately to survive and compete in the digital era. A key step in taking control of the future is to migrate their core banking systems to the cloud.

In this next instalment we will discuss the key components of cloud native technology and its benefits: scalability, flexibility, availability, elasticity, and more. To take full advantage of these benefits, core banking systems need to be built in the cloud and for the cloud – coined by many technology giants as 'cloud native'.

What is cloud native?

As expressed in the CNCF definition, cloud native systems offer speed and agility, as well as the flexibility to run in various cloud environments. To gain these benefits, cloud native core banking systems need to be built with some key features in mind: immutable infrastructure, microservices architecture, and containerisation.

Immutable infrastructure

In an immutable infrastructure, each application instance functions as a virtual machine or container. Scalability is achieved via automation — instances are created on demand, and destroyed when no longer needed. If one fails to work, a new instance is automatically provisioned for replacement.

Cloud native systems embrace immutable infrastructure and offer automatic scaling, self-healing, and zero downtime. These features are critical for modern banking as customers demand 24x7 access and real-time synchronisation of transaction data.

Microservices architecture

A microservices-based architecture is the bedrock of cloud computing. Amazon defines microservices as the following: "With a microservices architecture, an application is built as independent components that run each application process as a service. These services communicate via a well-defined interface using lightweight APIs. Services are built for business capabilities and each service performs a single function. Because they are independently run, each service can be updated, deployed, and scaled to meet demand for specific functions of an application."

"Cloud native technologies empower organisations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil."

Source: Cloud Native Computing Foundation (CNCF)

> "Containers are a great enabler of cloud native software."
>
> Cornelia David
> *Cloud Native Patterns*

Core banking systems built with a microservices-based architecture unshackles banks from the rigidity of monolithic legacy systems where any update to the system can mean lengthy software development cycles and unbearable downtime; not to mention the potential exorbitant costs involved.

In systems built around microservices, software consists of a suite of functional components that interoperate and communicate with one another. Each microservice addresses a specific business requirement and can be run independently. Together, they form the underlying system that offers flexibility, interoperability and portability that are critical for meeting the needs of modern-day banking.

Containerisation

The CNCF places microservice containerisation as the first step in their Cloud Native Trail Map – guidance for enterprises beginning their cloud native journey.

Containers are used to package a microservice with its necessary dependencies. The objective is to provide portability and consistency so that the microservice can be platform agnostic and function in any underlying infrastructure. Containerisation also offers cost and resource efficiencies by eliminating the costs of configuring runtime environments and by sharing operating system and host resources.

Managing and running containers require container orchestrators, among which the most popular is Kubernetes. Container orchestration plays a key role in providing scalability and portability.

Service meshes

When running microservice architectures in the cloud, networking between numerous deployed containers becomes a complex problem. Some common issues that need to be solved are:

Ability to automatically retry idempotent requests

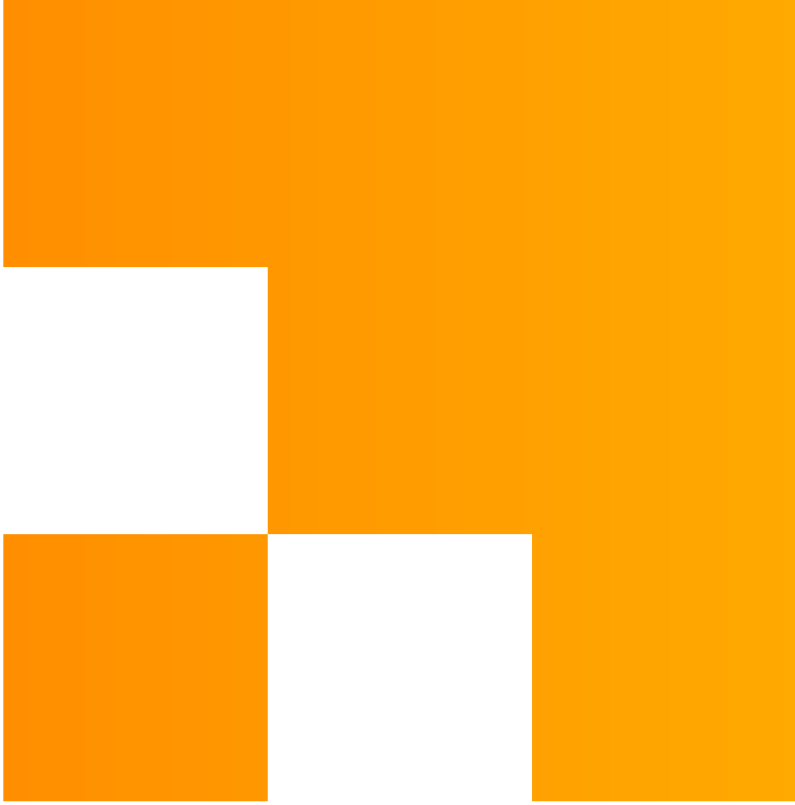
Establishing persistent connections between microservices for improved performance

Traffic shaping and request throttling

Establishing secure connections over TLS and automatic certificate rotation

Addressing these challenges at the application level is cumbersome and error prone, especially when using many different programming languages. This is where service meshes come in. A service mesh establishes a more sophisticated network topology between the microservices in a cluster. This frees up developers from having to worry about common scenarios, such as network partitions or weighted load balancing across multiple backends.

The Istio service mesh, a CNCF project, allows containerised applications to be part of a service mesh in a language-agnostic way, by attaching sidecars to Kubernetes deployments.



API communication

With microservices being loosely coupled and self-contained, they rely on APIs to communicate with each other. In the new world of microservices, more APIs have taken on a declarative approach as opposed to the traditional imperative approach. Whereby imperative APIs provide step-by-step instructions, declarative APIs are taking the center stage as they focus on the end results and let microservices figure out how to get there.

To implement declarative APIs, developers increasingly adopt REST, which is a set of architectural constraints used in API development. RESTful APIs offer a standardised, stateless architecture that makes it easy for microservices to integrate and automate microservices.

Challenges of cloud native environments

Managing scale and elasticity comes with challenges. With cloud native systems exposing their services using APIs and leveraging containers, banks adopt the notion of infrastructure as code. In this manner, an infrastructure topology can be codified as a simple script and then executed each time that infrastructure is required. By changing simple elements of the code, practically infinite amounts of identical infrastructure can be provisioned instantaneously. However, each infrastructure has its own provisioning model. Banks can be challenged to determine a strategy that works with their consistent provisioning workflow regardless of infrastructure type while leveraging the unique capabilities of each cloud provider.

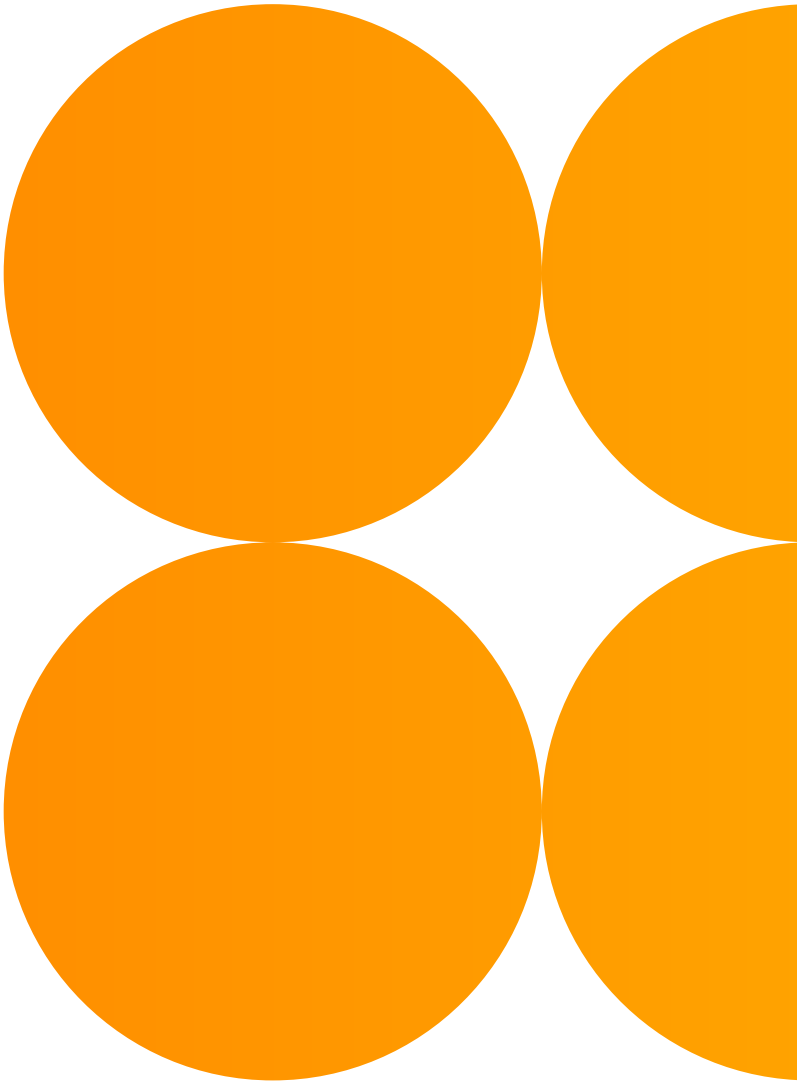
Another problem many organisations face is keeping track of all their assets. This gets aggravated especially in a multi-cloud environment, where services are heterogeneous, do not provide a single pane of glass for their management, and, thanks to infrastructure as code, can be easily deployed from every engineer within the company.

The aforementioned two challenges are common for ephemeral computing environments. Ultimately, banks need a way to detect, identify, categorise, and visualise all the assets being deployed in its enterprise systems, regardless of the cloud provider in use. To do so, cloud native systems need the ability to consolidate infrastructure assets and the relationships among them in a database supported by an intuitive graphical user interface.

Conclusion

Cloud native is the inevitable path for next-gen systems, including the core. It involves careful planning and migration from legacy technology to cloud native technology cannot be treated as a 'lift and shift' exercise. To exploit the benefits of cloud computing, core banking systems need to be built with all of the features of cloud computing in mind.

Once established on a good foundation, the sky is the limit. With a cloud native core system in place, banks can better control costs and more effectively respond to customer and regulatory demands.



About the authors

Fabian Siddiqi

Fabian leads the cloud infrastructure, production operations and security functions at Thought Machine. He is responsible for ensuring that Vault is truly cloud native and can be deployed across all of the major public cloud providers. He has 10+ years experience developing software and infrastructure, working at Google and early-stage startups.

Brian Dempsey

Brian leads partnerships in North America for Thought Machine, and is responsible for defining and driving the strategy for growing and enabling the partner ecosystem in the region. As a former management consultant, Brian has gained significant experience advising U.S. financial institutions on their digital transformation journey.

Robin Zhang

Robin leads Thought Machine's go-to-market efforts in North America. She spent the early part of her career as a software engineer doing heavy coding with C++ and designing RDBMS. Her love for business and communications ultimately led her to become a B2B marketing geek with a passion for cutting-edge technology offerings.

About Thought Machine

Thought Machine was founded in 2014 with a mission to enable banks to deploy modern systems and move away from the legacy IT platforms that plague the banking industry. We do this through our cloud native core banking platform, Vault. This next generation system has been written from scratch as an entirely cloud native platform. It does not contain a single line of code which is legacy, or pre-cloud.

Founded by entrepreneur Paul Taylor, Thought Machine's customers include Arvest, Atom bank, Curve, JPMorgan Chase, Lloyds Banking Group, Monese, SEB and Standard Chartered. We are currently a team of more than 500 people spread across offices in London, New York, Singapore, Sydney, Melbourne and have raised more than $150m in funding from Eurazeo, Draper Esprit, SEB, British Patient Capital, IQ Capital, Playfair Capital, Nyca Partners, Lloyds Banking Group and Backed.

For more information email:
contact@thoughtmachine.net

or visit:
thoughtmachine.net

Thought Machine