The Case for Cloud Migration

Team 5

March 21, 2025

Just as any global bank, real, or otherwise, would experience, a vast amount of extremely sensitive fiscal data must be sifted through constantly and at great rates. For this reason, amongst others, we propose the *fully cloud native private cloud* approach to cloud migration. We detail four points of reasoning below. As well as that, we outline how we would handle migration internally.

1 Benefits of Cloud Migration

1.1 Scalability & Server Hammering

Currently, our bank is being run off a local machine, often a laptop (the creative reader may be able to suspend their disbelief and consider it as a server). At some point, the number of transactions in the system will exceed the storage space of that laptop (or server).

Similarly, the variable rate of transactions could possibly cause some asynchronous problems relating to reading/ writing to databases, which may lead to very unhappy customers.

Outsourcing database and server concerns to a cloud provider mitigate these risks. It allows the dynamic allocation of server overhead at almost instantaneous speed.

1.2 Endurance & Quick Deployment

Our banking app in its current state is only active as long as the local machine is active. This, by itself is an extremely lousy and short-sighted approach for a worldwide service. A single hardware failure, power outage, system crash can cause the entire service to grind to a halt.

This single point of failure, obviously, is not acceptable for a banking service. By migrating server processing to a cloud provider, we can eliminate this vulnerability and ensure continuous availability.

1.3 Security

Cloud providers offer a large suite of security features that surpass what can reasonably be implemented on a local machine. These include but are not limited to:

1.3.1 Data Encryption

Cloud providers implement encryption both in transit and at rest. This ensures that even if data is intercepted during communication or compromised at its storage location, it remains unreadable to unauthorised entities.

1.3.2 DDoS Protection

Please see Mitigating DDoS Attacks.

1.3.3 Automated Security Patching

Traditional on-premise servers require manual updates and security patching, which may introduce vulnerabilities if neglected. Cloud providers continuously monitor security threats and apply patches automatically, reducing the attack surface and eliminating known exploits in a timely manner.

1.3.4 Intrusion Detection & Threat Monitoring

Cloud environments often include real-time monitoring tools, which analyse network traffic, detect anomalies, and alert administrators about potential security incidents, allowing for rapid response and mitigation.

By leveraging these cloud security benefits, our banking service can achieve a higher level of protection compared to a self-hosted solution, ensuring customer data integrity, confidentiality, and availability.

1.4 Load Balancing

A crucial component of cloud infrastructure is load balancing, which distributes incoming traffic across multiple servers to prevent overload and ensure high availability. This is particularly important for banking services, where downtime or slow response times can severely impact user experience and trust.

1.4.1 Mitigating DDoS Attacks

One major benefit of cloud-based load balancing is its role in mitigating Distributed Denial of Service (DDoS) attacks. These attacks attempt to overwhelm a system with an excessive number of requests, leading to service degradation or failure. Cloud providers implement DDoS protection through:

- **Traffic Filtering:** Identifying and blocking malicious traffic patterns before they reach application servers.
- **Rate Limiting:** Restricting excessive requests from single sources to prevent flooding.

- **Global Anycast Networks:** Distributing traffic across multiple data centres to absorb attack volumes.

By integrating DDoS protection with load balancing, cloud providers ensure that legitimate requests are always processed efficiently while malicious traffic is minimised.

1.4.2 Performance Optimisation

Load balancers dynamically allocate resources based on real-time traffic demand, preventing bottlenecks and ensuring smooth operation. They also support:

- **Auto-scaling:** Automatically provisioning additional resources as needed.
- **Health Monitoring:** Detecting and re-routing traffic away from unhealthy instances.
- **Geographic Distribution:** Routing users to the nearest data centre for reduced latency.

Implementing cloud-based load balancing not only enhances security but also optimises performance and reliability, making it a fundamental component of a cloud-native banking solution.

2 The Migration to the Cloud

Considering the sensitive nature of the data we must manipulate, we propose a *private cloud* approach.We appreciate the possible benefits of a *hybrid cloud*, particularly its scalability and more integrated telemetry features. However, given the current projected scale of the app and the need for control over sensitive data, we believe a private cloud offers the right balance of flexibility and security for our use case.

While private clouds can be scaled to meet demands, they require proactive management, such as provisioning hardware and ensuring that capacity grows as needed. Public clouds, on the other hand, offer automated scaling with minimal intervention, which may be a benefit as our app grows in scale.

In a private cloud, while we retain full control over security measures, we also take on the responsibility of implementing and maintaining encryption, DDoS mitigation, patching, and intrusion detection. This may require more resources and expertise compared to using a managed public cloud service.

One consideration for using a private cloud is the potential for higher initial setup and maintenance costs. In contrast, public clouds operate on a pay-as-you-go model, which can reduce initial costs and increase flexibility as the app scales.

As already mentioned, we would employ the *fully cloud native* approach. This means following the 12-factor applications for creating cloud native applications.

Due to the modularity and loose coupling of our codebase, we believe transitioning to cloud would be fairly painless. The following is a non-exhaustive list of cloud appropriate applications for our bank.

- Currently, all transaction and account data is received via API into a local database on every new instance of the app. That can easily be uprooted into a cloud handled database.

- Instead of having a local database handlers, we can create database queries through some API, and switch the back-end wiring to call the cloud API, instead of the local function. Consider the following two functions:

```
function DisplayAllTransactionsUsingLocalDB()
{
  Transactions = localDatabase.getAllTransactions()
  Create new Page
  Load Transactions onto Page
  Render Page
}

function DisplayAllTransactionsUsingCloudAPI()
{
  Transactions = remoteApi.getAllTransactions()
  Create new Page
  Load Transactions onto Page
  Render Page
}
```

  Due to the loosely coupled nature of our codebase, we would be able to simply change the data source from local to cloud whilst keeping the logic the exact same.

- A cloud based logger, or ledger, would be available for use, allowing for an immutable history of transactions.

- Telemetry (e.g., user spending habits) in a private cloud allows full control over the data collection process, its processing, and its storage location.