Diagram Arsitektur Bank XYZ

Dari gambar tersebut, dapat kita simpulkan bahwa ekosistem *Core Banking* XYZ telah menerapkan standar arsitektur yang kuat, *'Continuous Integration/Deployment'*, serta dokumentasi API yang terstruktur. Berikut adalah beberapa poin utama terkait ekosistem arsitektur bank yang kompleks ini:

1. Ekosistem Layanan Bank yang Kompleks

Ekosistem bank XYZ mencakup berbagai layanan dan aplikasi, seperti ATM, mobile banking, internet banking, serta aplikasi internal untuk karyawan. Setiap platform menggunakan teknologi yang berbeda-beda, atau disebut *'technology stack'* yang beragam.

2. Penerapan Service-Oriented Architecture (SOA)

Bank XYZ sudah menerapkan desain arsitektur perangkat lunak **Service-Oriented Architecture (SOA)**, di mana fungsi-fungsi aplikasi dipecah menjadi *service* yang lebih kecil dan independen. Hal ini memungkinkan setiap *service* untuk diintegrasikan dan digunakan bersama dengan aplikasi lain melalui protokol komunikasi jaringan.

Keuntungan arsitektur SOA meliputi:

- **Independen**: Setiap *service* memiliki fungsi yang mandiri, seperti otentikasi pengguna dan pemrosesan pembayaran.
- **Reusable**: Dirancang untuk dapat digunakan di berbagai aplikasi atau modul tanpa perlu membuat *service* yang sama berulang kali.
- **Interoperabilitas**: Mendukung standar komunikasi untuk berinteraksi dengan berbagai sistem teknologi, sehingga memudahkan integrasi antara sistem lama (legacy) dan baru.
- **Flexibility**: *Service* yang tidak saling tergantung membuat perubahan pada satu layanan tidak langsung mempengaruhi layanan lainnya, meningkatkan fleksibilitas dan skalabilitas sistem.

3. Pendukung DevOps dengan OpenShift

Untuk mendukung siklus pengembangan dan integrasi dari banyaknya aplikasi dan platform, PT.XYZ menggunakan **OpenShift** dari Red Hat. OpenShift merupakan platform berbasis kontainer yang memungkinkan aplikasi berjalan di mana saja yang mendukung Docker. Bahasa pemrograman yang digunakan adalah Java dengan framework **Spring Boot** yang relevan dengan pendekatan **SOA**, termasuk dukungan untuk *Service Registry/Discovery* menggunakan **Eureka**. Kontainer yang dikelola di OpenShift dikelola dalam group (*pods*) yang berjalan dalam *virtual machine* yang disebut **Worker Nodes**. *Worker Nodes* bertanggung jawab dalam pemrosesan data cluster, menangani lalu lintas jaringan, dan maintain *pods* yang berjalan.

4. Continuous Integration/Deployment dengan Jenkins

Bank XYZ menggunakan **Jenkins** sebagai alat *'Continuous Integration/Deployment'* yang andal untuk mengelola pengembangan aplikasi dan otomatisasi pengujian. Jenkins bisa diinstal sebagai server mandiri (*Standalone*) atau dijalankan di dalam OpenShift sebagai *pods*. Instruksi langkah dalam otomatisasi (*Continuous Deployment*) yang dapat dijalankan oleh mesin CI/CD disebut 'CI Pipeline'. Script pipeline dalam Jenkins (Jenkinsfile) dapat kita deklarasikan aplikasi yang di *'Build'* untuk berjalan pada container OpenShift memungkinkan administrator DevOps melakukan konfigurasi terpusat pada OpenShift.

5. Pengelolaan *'Big Data'* dengan Elastic Stack

Tentu saja dari setiap aktivitas aplikasi yang berjalan, akan menghasilkan data baik transaksional maupun data seperti log aktivitas pengguna dan lainnya. Salah satu platform yang mendukung untuk pengolahan data dalam jumlah besar adalah Elastic Stack. Produk unggulan dari Elastic diantara lain:

- **Kibana**: Merupakan tool untuk melakukan observasi / analisis data yang powerful. Berbagai sumber data seperti log aktivitas, monitoring aplikasi, data transaksi, dan sebagainya dapat disajikan secara visual dalam grafik yang cukup lengkap. Fitur-fitur yang ada pada Kibana antara lain:
  - **Kibana Lens**: Membuat berbagai grafik popular seperti 'Bar Chart', 'Area Chart', 'Line Chart', 'Donut Chart', 'Data Table', dan sebagainya secara mudah dan intuitif.
  - **Kibana Canvas**: Media untuk kreatifitas dalam pembuatan dashboard yang dinamis. kita dapat memasukkan elemen seperti logo, warna, dan custom CSS sehingga info yang disajikan sangat berkesan dan dapat di personalisasi sesuai kebutuhan.
  - **Elastic Maps**: Penyajian analisa data dengan visualisasi geospasial akan sangat mudah dipahami kebanyakan orang. Dengan lisensi peta 'OpenStreetMap' kita dapat menyampaikan informasi yang dipadukan dengan lokasi sehingga penyajian data dapat lebih interaktif.
  - **Kibana Alerting**: Data metrik seperti penggunaan resource aplikasi, uptime, pelanggaran keamanan, dengan threshold tertentu. Kita bisa meintegrasikan pengiriman alert via email, slack, dsb.
  - **Kibana Dashboard**: Semua chart, maps, dan visualisasi lainnya yang sudah kita buat dengan fitur diatas, dapat kita sajikan terintegrasi pada halaman dashboard. Pengguna dapat dengan mudah menavigasi filter dan parameter lainnya untuk menyesuaikan informasi yang disajikan.
- **Elasticsearch**: Adalah jantung dari platform elastic. Merupakan salah satu mesin database non relational yang cukup popular digunakan belakangan ini. Memiliki kemampuan untuk menyimpan berbagai jenis data, skalabilitas yang mudah, serta peforma pencarian data yang cepat untuk skala penyimpanan data yang besar.
- **Logstash**: *Server-side* data pipeline yang mampu menerima data dari berbagai sumber. Data tersebut selanjutnya di transformasi sesuai kebutuhan, kemudian disimpan pada database Elasticsearch.

6. Standar Dokumentasi Spesifikasi API - OpenAPI

Development sistem yang baik tentunya disertai dengan pendokumentasian yang baik pula. Salah satu bentuk dokumentasi teknis dari pengembangan system adalah terkait spesifikasi API. Saat ini API yang banyak diaplikasikan untuk komunikasi system adalah **RESTful**. **OpenAPI** adalah standar untuk mendeskripsikan spesifikasi API dalam format **JSON** atau **YAML**.

Untuk memudahkan dokumentasi (API khususnya), ada platform berbasis *'Open Source'* yang dapat kita gunakan, salah satunya adalah **Swagger Editor**. Swagger Editor dapat kita setup pada server kita sendiri dengan NodeJS dan dependensinya yang sudah terinstall.

Keuntungan utama pendokumentasian API pada Swagger adalah kita dapat dengan mudah *'share'* spesifikasi API. Vendor/pihak lain dapat mengakses dan mencoba secara langsung API yang sudah kita buat hanya dengan mengakses halaman website melalui browser.