BINUS UNIVERSITY
GLOBAL CLASS

Final Exam Answer

Nama Mahasiswa: *Christopher Angelo - 2440041503*

Kelas: *Distributed Cloud Computing (COMP6736001)* – Dosen: *Mr. Said Achmad*
Tenggat Waktu: *Feb 3rd, 2023*

Question 1

Sebutkan dan jelaskan perbedaan mendasar dari arsitektur *monolithic* dan *microservice*, tuliskan kelebihan beserta kekurangan dari masing-masing arsitektur.

Answer 1. Arsitektur *monolithic* dan *microservice* saling bertumpang-belakang.

Monolithic. Arsitektur *Monolithic* adalah arsitektur program yang membangun sebuah sistem dalam satu kesatuan, berhubungan internal secara ketat. Secara natural, arsitektur ini akan bersifat tersekat (*contained*), tersentralisasi dan berdiri sendiri secara independen.

Saat membangun sistem menggunakan arsitektur monolithic, ada beberapa kelebihan dan kekurangan yang harus diperhatikan. Kelebihan dan kekurangan berikut saling bertumpang tindih (*double-edged sword*) dengan arsitektur *microservice*:

- **Development** — Oleh karena kode yang satu-kesatuan, proses *development* akan lebih mudah; Tidak perlu memikirkan bagaimana satu sistem berinteraksi dengan sistem lain. Tetapi juga, proses *development* mungkin akan lebih lambat jika terstruktur tidak optimal oleh karena satu kesatuan yang saling berhubungan dengan satu sama lain.
- **Technology Adoption** — Jika ada teknologi baru (*programming language*, *framework*, *library*, etc.) yang ingin diadopsi, kemungkinan akan lebih sulit untuk diadopsi oleh karena satu kesatuan yang saling berhubungan dengan satu sama lain, yang mungkin akan mem-efek pada keseluruhan sistem.
- **Deployment** — Sistem *monolithic* akan lebih mudah untuk di-*deploy* oleh karena hanya ada perlu satu kesatuan yang perlu di-*deploy*. Tetapi perlu diperhatikan bahwa setiap perubahan yang dilakukan (walaupun hanya 1 baris kode), akan memerlukan proses *deployment* ulang pada seluruh aplikasi.
- **Reliability** — Jika terjadi kegagalan pada satu bagian dari sistem yang tidak di-*handle* secara optimal, maka keseluruhan sistem mungkin akan terganggu, sampai pada titik dimana seluruh aplikasi akan terganggu / down.
- **Scalability** — Skalabilitas yang mudah dilakukan hanyalah dengan secara vertikal (*Scaling up*). Oleh karena itu, jika ada salah satu komponen sistem yang perlu di-scale, maka seluruh sistem harus di-scale. Sistem *monolithic* akan lebih sulit untuk di-*scale* secara horizontal karena sifatnya yakni *contained* dalam satu kesatuan.

Microservice. Arsitektur *microservice* adalah arsitektur yang membangun sebuah sistem dalam banyak bagian kecil (*micro*) yang saling terpisah. Melawan arsitektur *monolithic*, *service* yang dibangun akan harus bersifat kecil dan independen; masing masing *service* akan memiliki *business logic* dan mungkin *database* tersendiri.

Berikut kelebihan dan kekurangan dari arsitektur *microservice*. Umumnya, kelebihan dan kekurangan dari arsitektur ini akan bertumpang tindih dengan arsitektur *monolithic*:

- **Development** — Pada awal mula pembuatan sistem, infrastruktur sistem perlu didesain secara konkret untuk menjamin kecepatan development agar menjadi sesuai dengan standar. Hal ini disebabkan oleh karena banyaknya *service* yang harus dibangun, sehingga proses untuk 'menghubungkan' service tersebut menjadi kompleks. Tetapi jika diatur dengan baik, proses *development* akan lebih cepat dan efisien.
- **Deployment** — Sifat *microservice* yang kecil dan independen membuat service untuk di*deploy* secara terpisah satu sama lain. Hal ini akan mendorong teknologi '*continuous deployment*' dimana setiap perubahan yang dilakukan dapat di-*push* secara berlanjutan, kadang 2 sampai 3 kali sehari.
- **Reliability** — Karena setiap *service* memiliki *business logic* dan *database* tersendiri, jika terjadi kegagalan pada salah satu *service*, maka hanyalah *service* tersebut yang perlu di*deploy* ulang (atau *rollback*) menghasilkan sistem yang lebih kokoh dan lebih sulit untuk membuat seluruh aplikasi down.
- **Scalability** — Independensi *microservices* memudahkan *service* tersebut untuk di-*scale* secara horizontal dan *on-demand*. Hal ini akan memudahkan sistem untuk menangani permintaan yang meningkat pada waktu tertentu, atau juga pada waktu yang tidak diantisipasi. Tetapi selain itu, perlu diperhatikan bahwa pemecahan *service* yang terlalu kecil akan menyebabkan biaya yang signifikan, bahkan kadang eksponensial oleh karena perlunya banyaknya hal yang perlu disetup pada setiap *service*.

Question 2

Jelaskan apa yang dimaksud dengan SOA (*System-oriented architecture*), kemudian jelaskan menggunakan kalimat anda sendiri mengenai perbedaan SOA dengan *microservice*.

Answer 2. *System-oriented architecture* adalah sebuah penggambaran dari bagaimana caranya sebuah sistem dibangun menjadi komponen kecil yang dapat dipakai berulang kali di *service* yang berbeda. Hal ini mungkin mirip seperti *microservice*, tetapi *microservice* lebih terfokuskan pada pemecahan sistem monolitik menjadi banya *service* yang saling kecil terpisah. Sedangkan SOA lebih terfokus pada pemecahan interface service agar dapat dipakai berulang kali di *service* yang berbeda. Mungkin dapat dibilang bahwa SOA adalah dasar dari *microservice*.

Question 3

Buatlah tabel perbandingan yang membahas kekurangan serta kelebihan dari SOA dan *microservice*.

Answer 3. Berikut adalah tabel perbandingan kekurangan dan kelebihan dari SOA dan *microservice*.

	SOA	Microservices
Code Reusaibility		**Kelebihan**
Kekurangan	**Kekurangan**	**Kekurangan**

Table 1: Perbandingan SOA dan *microservice*

Question 4

Tuliskan 5 layanan dari AWS atau Azure dan lakukan hal berikut:

a. Tentukan apakah layanan tersebut masuk dalam kategori Iaas, PaaS, atau SaaS. Berikan alasannya

b. Jelaskan tarif yang dikenakan jika menggunakan layanan tersebut.

c. Deskripsikan use case penggunaan layanan tersebut.

Answer 4. Berikut adalah beberapa layanan dari AWS dan penjelasan mengenai layanan tersebut.

EC2. Elastic Compute Cloud — Sebuah IaaS yang memberikan pelanggan sebuah *Barebone VPS* (*Virtual Private Server*) yang ditujukan untuk membuat / hosting aplikasi enterprise.

Berdasarkan spesifikasi yang dibutuhkan, pelanggan dapat memilih apakah mereka mau dikenakan biaya termuka (*upfront*) dengan *savings plan* atau per-jam yang di-charge per bulan. Biaya per-jam ini akan berbeda-beda tergantung dari jenis instance yang digunakan.

Beberapa use-case dari EC2 adalah hosting aplikasi enterprise, Big Data, distributed workload (Spark dan Hadoop) dan juga migrasi dari situasi on-premise ke cloud.

Lightsail. Mirip seperti EC2, Amazon Lightsail adalah sebuah IaaS yang memberikan pelanggan sebuah *Barebone VPS* (*Virtual Private Server*) yang ditujukan untuk membuat / hosting aplikasi personal. Service ini lebih ditujukan untuk pengguna personal yang akan men-deploy aplikasi berskala simpel.

Seperti EC2, pricing yang ditawarkan berbeda tergantung spesifikasi yang dibutuhkan. Namun, Lightsail tidak memiliki opsi pembayaran *upfront* cost, sehingga pelanggan hanya akan dikenakan biaya per-bulan. Juga, opsi instance yang ditawarkan oleh Lightsail cenderung lebih terbatas (kurang fleksibel) dibandingkan EC2. Tetapi, hal ini memungkinkan Lightsail untuk memiliki biaya yang lebih murah dibandingkan EC2.

Beberapa use-case dari Lightsail meliputi hosting aplikasi personal, seperti blog, portfolio, dan juga situs personal, aplikasi web yang tidak membutuhkan banyak resource, pembelajaran cloud computing, dan juga untuk menjadi environmen development / testing.

S3. Simple Storage Service — Sebuah PaaS yang memberikan pelanggan sebuah *Object Storage* yang dapat digunakan untuk menyimpan data secara aman dan dapat diakses dari mana saja. PaaS karena pelanggan tidak perlu mengurus infrastruktur storage, namun hanya perlu mengurus bagaimana cara data disimpan / diakses.

Biaya penggunaan S3 umumnya dikenakan berdasarkan volume data yang disimpan (Standard plan per gigabyte) yang berbeda di setiap datacenter. Plan standard mendapatkan bandwidth network gratis untuk menimpan KE S3, dan gratis mengambil 100GB data pertama per bulan.

Use-case S3 meliputi *Object Storage* dari sebuah aplikasi yang mungkin berbentuk user data, profile picture, gambar, video, dll.

Lambda. Sebuah PaaS yang memberikan pelanggan sebuah *Function as a Service* yang dapat digunakan untuk menjalankan kode tanpa harus mengurus infrastruktur server (*serverless*). Pelanggan hanya perlu mengurus bagaimana cara kode tersebut dijalankan.

Biaya penggunaan Lambda dikenakan berdasarkan jumlah invokasi yang dilakukan. Pelanggan akan mendapatkan 1 juta invokasi secara gratis. Setelah itu, pelanggan akan dikenakan biaya per-juta invokasi.

Contoh use-case dari Lambda adalah untuk menjalankan kode / function yang tidak lama untuk dijalankan, tetapi sering di-invokasi namun tidak membutuhkan infrastruktur server yang besar. *image processing*,

ECS. Sebuah SaaS yang memberikan pelanggan sebuah *Container as a Service* yang dapat digunakan untuk menjalankan container tanpa harus mengurus infrastruktur server. Pelanggan hanya perlu mengurus environment didalam container tersebut dijalankan.

Biaya penggunaan ECS dikenakan berdasarkan jumlah container yang dijalankan dalam instansi EC2. Penghargaannya juga berbeda tergantung dari jenis instansi EC2 yang digunakan.

Hampir semua server dapat didesain untuk dijalankan di sebuah container melalui Docker atau Kubernetes. Sehingga, use-case dari ECS adalah untuk menjalankan container yang dibuat menggunakan Docker atau Kubernetes.

Question 5

Jelaskan apa itu ubiquitous computing dan bagaimana teknologi tersebut berperan dalam sistem cloud computing?

Answer 5. Ubiquitous Computing adalah konsep / prinsip dimana teknologi digital computing dapat digunakan untuk mempermudah kehidupan manusia dimanaput. Teknologi ini dapat berupa hardware, software, ataupun juga service yang sering digunakan sehari hari. Banyak intisari dari prinsip ini yang terselubung dengan konsep IoT dan Cloud Computing oleh karena kedua hal tersebut dapat diakses dari mana saja (hanya butuh koneksi ke satu sama lain).

Question 6

Jelaskan apa itu internet of thing (IoT) dan berikan satu contoh layanan dari cloud provider yang mendukung IoT beserta use case penggunaannya.

Answer 6. Internet of Things adalah semua benda fisik yang memiliki sensor, kemampuan untuk memproses data dari sensor tersebut, dan mengirimkan hasil proses data tersebut ke device lainnya (*IoT Hub*), maupun cloud.

Salah satu layanan dari cloud provider yang mendukung IoT adalah Tuya. Tuya adalah platform IoT yang memfokusan developer produk fisik untuk membangun produk IoT. Mereka menyediakan SDK dan API untuk platform low-power, dan juga menyediakan platform tool untuk banyak tipe appliance IoT (Sensor temperature, sensor kelembapan, lighting, camera, smart lock, cleaning robot, dan lain lain)

Tuya juga akan dapat terhubung secara otomatis ke Apple HomeKit atau Google Home agar pengguna dapat mengakses sensor dan mengantar perintah ke device IoT melalui aplikasi tersebut.

Case Study

Suatu perusahaan Fintech berencana melakukan migrasi arsitektur TI berbasis monolithic yang digunakan saat ini. Berdasarkan keterangan dari tim system analyst, arsitektur tersebut, sulit mengakomodir traffic yang meningkat dalam beberapa bulan terakhir. Arsitektur monolithic yang digunakan terlihat pada gambar 1. Anda diminta untuk melakukan migrasi arsitektur TI tersebut dengan mendesain arsitektur berbasis microservice berdasarkan arsitektur monolithic yang ada.

a. Gambarkan rancangan arsitektur microservice dari perusahaan Fintech tersebut dengan jelas (silahkan gunakan gaya anda sendiri dalam menggambarkan desain arsitektur).

b. Berikan **dua** rekomendasi layanan dari cloud provider yang dapat digunakan pada arsitektur microservice yang dirancang. Ceritakan bagaimana service tersebut digunakan dan berperan dalam arsitektur microservice tersebut.

Case Study: Answer 1a. Berikut adalah potensi arkitektur fintek yang menggunakan microservice:

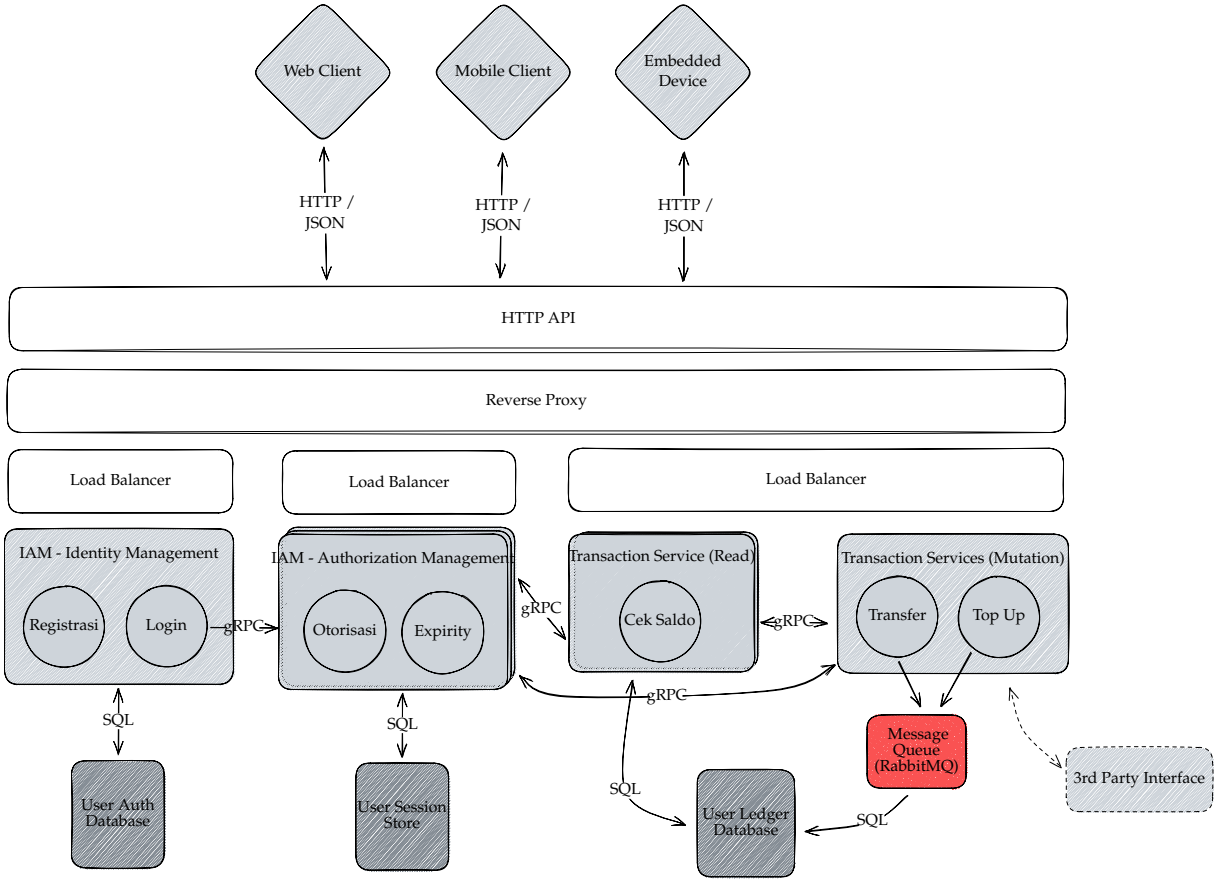

Figure 1: Potensi arsitektur fintech menggunakan microservice

Case Study: Answer 1b. Banyak layanan cloud provider yang dapat menampung arsitektur microservice. Oleh karena perkembangan teknologi *containerization* yang pesat, hampir semua layanan cloud provider dapat digunakan untuk arsitektur microservice. Semua cloud provider yang dapat menjalankan sebuan container (Docker / K8s) dapat digunakan untuk arsitektur microservice. Berikut adalah beberapa layanan cloud provider yang memiliki support untuk digunakan dalam arsitektur microservice:

- **AWS ECS, Load Balancer dan VPC** — AWS ECS dapat digunakan untuk menjalankan container yang dibuat menggunakan Docker. Setelah berjalan, Load Balancer dapat dipasang agar semua routing kepada API dapat tersebar secara merata ke semua instansi. Juga, load balacer dapat mereplikasi jumlah container yang ada agar sistem tetap available. Agar service dapat diakses secara cepat secara internal, Amazon VPC dapat digunakan untuk mendesignasi addressing / subnet yang terdapat pada container.
- **Render.com Services** — Render adalah layanan cloud provider PaaS yang berfokuskan pada memudahkan developer untuk mendeploy aplikasinya secara *reliable*. Layanan ini menyediakan layanan untuk menjalankan container Docker yang dapat scalable secara otomatis. Mereka juga menyediakan layanan untuk hosting database secara managed agar dapat dipakai oleh container tersebut. Semua container akan terhubung secara otomatis pada satu subnet yang sama sehingga data dapat diakses dengan cepat secara internal. Render ini juga menyediakan monitoring container otamatis yang dapat mengumpulkan telemetri.