Arsitektur Perangkat Lunak Modern

Architectural Patterns and Styles

Husni

Husni.trunojoyo.ac.id

husni@trunojoyo.ac.id

Daftar Isi

Pendahuluan

Ragam & Pola Arsitektural

- Ragam (*style*) arsitektural adalah suatu **kekhususan** dari jenis **elemen** dan **relasi**, bersama dengan himpunan pembatasan **bagaimana** semua itu **dapat digunakan**.
- Pada sisi lain, pola (*pattern*) arsitektural mengekspresikan suatu skema organisasi struktural yang fundamental bagi sistem software.
  - Menghadirkan himpunan sub-sistem yang telah ditetapkan, menentukan tanggung jawab mereka, dan termasuk aturan dan pedoman untuk mengatur hubungan antar mereka.

- Clements et al (2011)

Ragam & Pola Arsitektural

- Dalam beberapa kasus, elemen-elemen arsitektur disusun mengikuti cara pemecahan masalah tertentu.
- Komposisi tersebut telah terbukti berguna dari waktu ke waktu, pada banyak domain berbeda, sehingga telah didokumentasikan dan disebarluaskan.
- Bentuk komposisi elemen arsitektural ini, dinamakan ***architectural patterns***, menyediakan strategi paketan untuk penyelesaian beberapa masalah di dalam sistem.

- (Bass et al, 2013)

Perbedaan antara *Patterns* & *Styles*

- Dalam Clements et al. (2011) dapat dilihat diskusi lebih panjang mengenai perbedaan antara pola dan gaya arsitektur.
- Ia berpendapat bahwa **Pattern** adalah suatu **triple** ***context-problem-solution***; sedangkan **Style** hanyalah sebuah **kondensasi** yang berfokus paling berat pada bagian ***solution***.

- (Bass et al, 2013)

Perbedaan antara *Patterns* & *Styles*

- Bagian penting dari suatu pola arsitektur adalah fokusnya pada masalah dan konteks juga bagaimana memecahkan masalah di dalam konteks itu.
- *Style* arsitektur fokus pada pendekatan arsitektur, dengan pedoman lebih ringan mengenai kapan suatu *style* tertentu dapat atau tidak dapat berguna.
- Secara sangat informal, dapat disederhanakan dengan:
  - **Pola** arsitektur: **{ problem, context } → pendekatan arsitektur**;
  - **Gaya** arsitektur: **pendekatan arsitektur**.

- Clements et al (2011)

Mengapa *Patterns* & *Styles*?

- **Meskipun tidak ada pandangan tepat untuk setiap sistem, panduan secara luas dapat membantu kita memperoleh pijakan.**
- Arsitek perlu memikirkan softwarenya dalam tiga cara secara simultan:
  1. Bagaimana software distrukturkan sebagai himpunan unit implementasi;
  2. Bagaimana software distrukturkan sebagai himpunan dari elemen yang mempunyai perilaku dan interaksi *runtime*;
  3. Bagaimana software berhubungan ke struktur non-software di dalam lingkungannya.

- Clements et al (2011)

Client-Server

Client-Server

- Komponen-komponen pada *style* client-Server berinteraksi dengan meminta layanan dari komponen lain.
- Requester (yang meminta layanan) disebut sebagai **client**, dan *service provider* disebut sebagai **server**, yang menyediakan sejumlah layanan melalui satu atau lebih portnya.
- **Client menginisiasi interaksi**, meminta layanan yang dibutuhkan dari server dan menunggu hasil dari permintaan tersebut.

- Clements et al (2011)

Client-Server: Keterbatasan

- Client terkoneksi ke server melalui konektor ***request/response***;
- **Komponen Server dapat menjadi client bagi server-server lain**;
- Spesialisasi dapat memberlakukan pembatasan :
  - Jumlah sambungan ke port yang diberikan;
  - Hubungan yang dimungkinkan antar server-server.
- Komponen-komponen dapat diatur dalam beberapa ***tier-tier***.

- Clements et al (2011)

Client-Server: Contoh

Contoh umum dari sistem ber-gaya client-server adalah

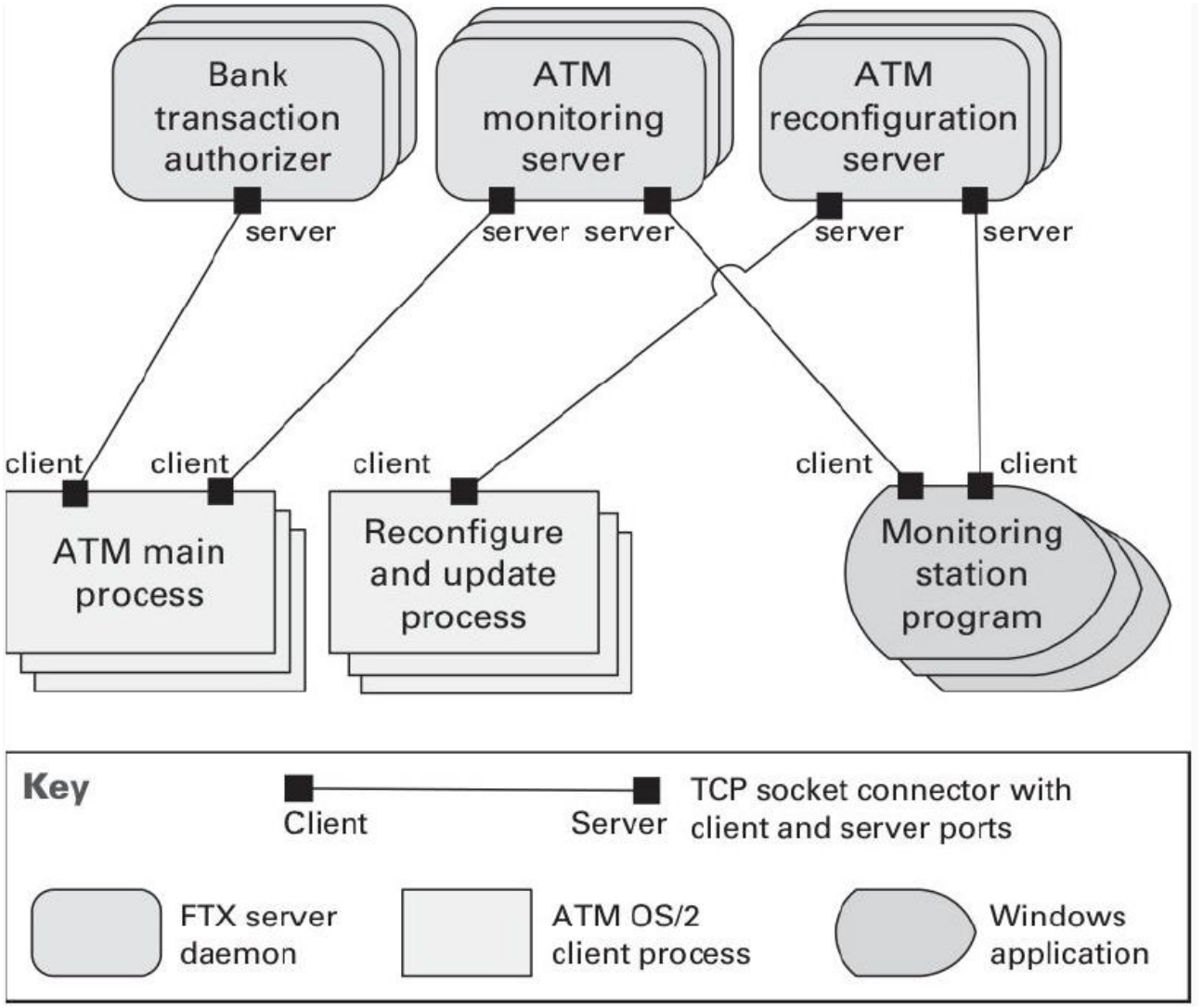
- Sistem informasi yang berjalan pada jaringan lokal, di mana kliennya adalah aplikasi GUI (seperti Visual Basic) dan server adalah sistem manajemen database (seperti Oracle);
- Aplikasi berbasis web di mana klien berjalan pada browser web dan server adalah komponen yang berjalan pada server web (seperti Tomcat atau Apache).

- Clements et al (2011)

Client-Server: Contoh ATM

- Sistem perbankan *Automated Teller Machine* (ATM) yang dikembangkan di awal 1990-an.
- Pada saat itu, arsitektur client-server adalah alternatif modern untuk sistem berbasis mainframe.
- (aplikasi server J2EE dan .NET belum ada. Arsitektur *multitier* belum tampil sebagai sebuah gaya.)

- Clements et al (2011)



Arsitektur client-server ATM sistem perbankan, menggunakan notasi resmi (Clements et al, 2011).

Service-Oriented Architecture (SOA)

SOA

- Service-Oriented Architecture (SOA) terdiri dari kumpulan komponen terdistribusi yang **menyediakan** dan / atau **mengkonsumsi** jasa (***service***).
- Dalam gaya ini, komponen penyedia layanan dan komponen konsumen (pelanggan) layanan **dapat menggunakan bahasa implementasi dan platform yang berbeda**. Sebagian besar layanan bersifat mandiri.
- Komputasi dicapai dengan satu set komponen yang bekerja sama untuk menyediakan atau mengkonsumsi layanan melalui jaringan.

- Clements et al (2011)

SOA: Elemen-elemen

- **Penyedia** (pemasok, provider) layanan: menyediakan satu atau lebih layanan melalui antarmuka yang diterbitkan (*publish*).
- **Properti-properti** dapat berbeda berdasarkan teknologi terapannya (seperti EJB atau ASP.NET) tetapi mencakup kinerja, kendala otorisasi, ketersediaan, dan biaya;
- **Pelanggan** layanan (konsumen, *customer*): memanggil layanan langsung atau melalui perantara.

- Clements et al (2011)

SOA: Elemen-elemen

- *Simple Object Access Protocol* (**SOAP**) **connector**, yang menggunakan protokol SOAP untuk komunikasi sinkron antar Web services, biasanya memanfaatkan **HTTP**.
- **Port** dari komponen yang menggunakan SOAP sering dideskripsikan dalam **WSDL**;
- Konektor *REpresentational State Transfer* (**REST**), yang bersandar pada operasi *request/response* dasar dari protokol **HTTP**.

- Clements et al (2011)

SOA: Ketentuan Khusus

- Konsumen layanan terhubung ke penyedia layanan, namun komponen perantara (seperti ESB, registry, atau server BPEL) dapat pula digunakan;
- ESB mengakibatkan hadirnya topologi *hub-and-spoke*;
- **Penyedia Service juga dapat menjadi pelanggan dari service**;
- Pola SOA spesifik menghadirkan ketentuan khusus tambahan.

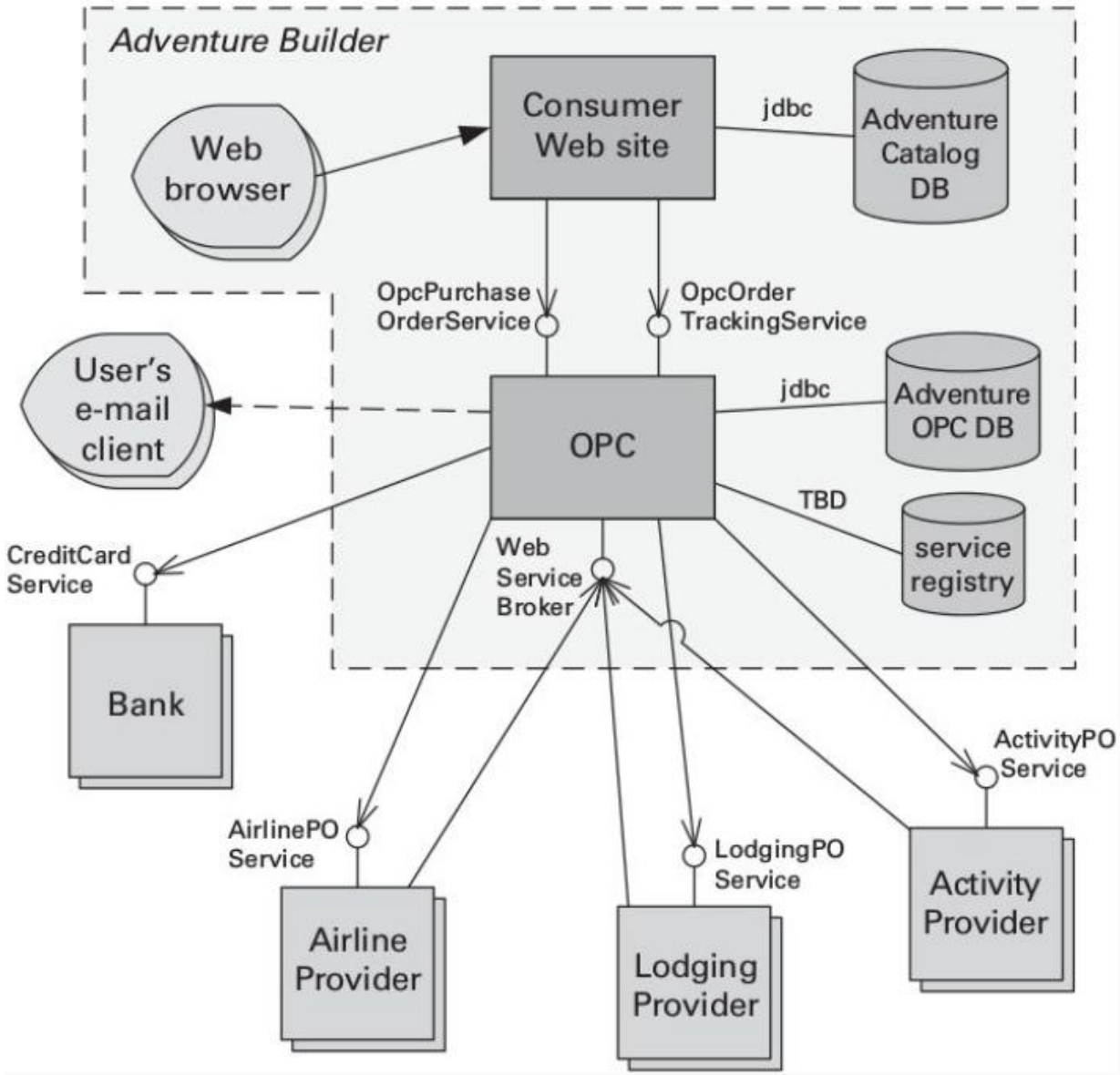
- Clements et al (2011)

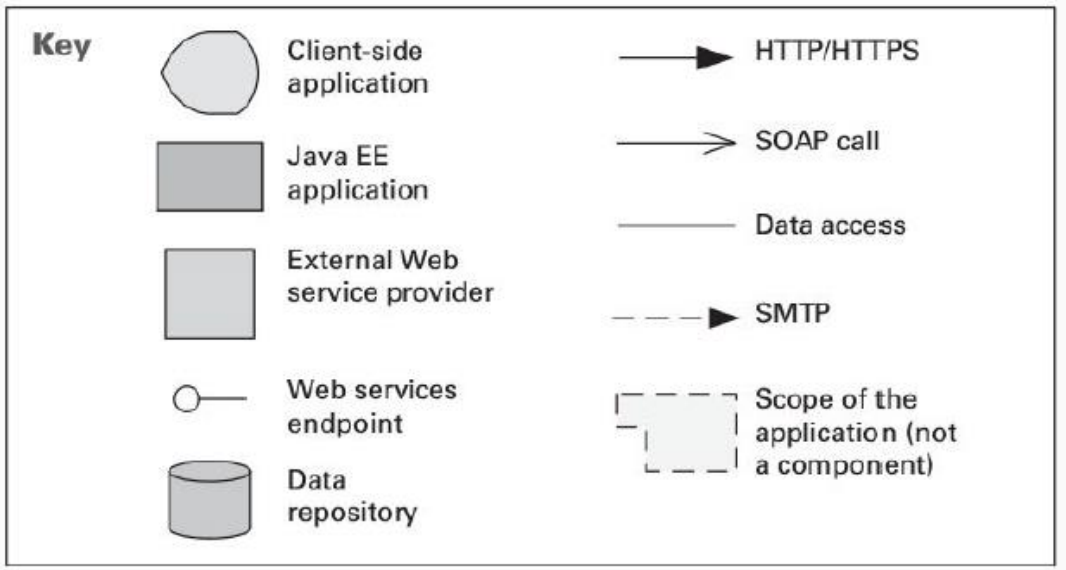
SOA: Contoh *Adventure Builder 2010*

- Sistem ini diambil dari dokumen arsitektur perangkat lunak contoh yang menyertai buku ini secara online, di wiki.sei.cmu.edu/sad.
- Sistem Adventure Builder (Adventure Builder 2010) berinteraksi melalui layanan Web SOAP dengan beberapa penyedia layanan eksternal.
- Penyedia service eksternal dapat berupa mainframe, Java, atau .NET - sifat komponen eksternal ini transparan karena konektor SOAP menyediakan interoperabilitas yang diperlukan.

- Clements et al (2011)

Diagram pandangan SOA dari Adventure Builder 2010, menggunakan notasi resmi (Clements et al, 2011)





REpresentational State Transfer (REST)

REST

- ***RE**presentational **S**tate **T**ransfer* (REST) adalah *architectural style* perangkat lunak dari **World Wide Web**.
- REST merupakan suatu *architectural style* bagi sistem hypermedia terdistribusi, menggambarkan prinsip-prinsip **software engineering** yang **mengarahkan** REST dan **ketentuan interaksi terpilih** untuk mempertahankan prinsip-prinsip tersebut, sekaligus membedakan mereka dari ketentuan ***architectural styles*** lain.

- Clements et al (2011) dan Fielding (2000) 23

REST

- Sejak 1994, REST telah digunakan untuk memandu rancangan dan pengembangan dari arsitektur web modern.
- Dikerjakan sebagai bagian kerja dari Internet Engineering Task Force (**IETF**) dan World Wide Web Consortium (**W3C**) untuk mendefinisikan standard arsitektural bagi Web: Hypertext Transfer Protocol (**HTTP**), Uniform Resource Identifier (**URI**), dan Hyper Text Markup Language (**HTML**).

- Fielding (2000)

REST: Ketentuan Khusus (*constraints*)

1. **Null Style**;
2. Client-Server = **Client-Server**;
3. Stateless = Client-**Stateless**-Server;
4. Cache = Client-**Cache**-Stateless-Server;
5. Uniform Interface = **Uniform**-Client-Cache-Stateless-Server;
6. Layered System = Uniform-**Layered**-Client-Cache-Stateless-Server;
7. Code-On-Demand = **REST**.

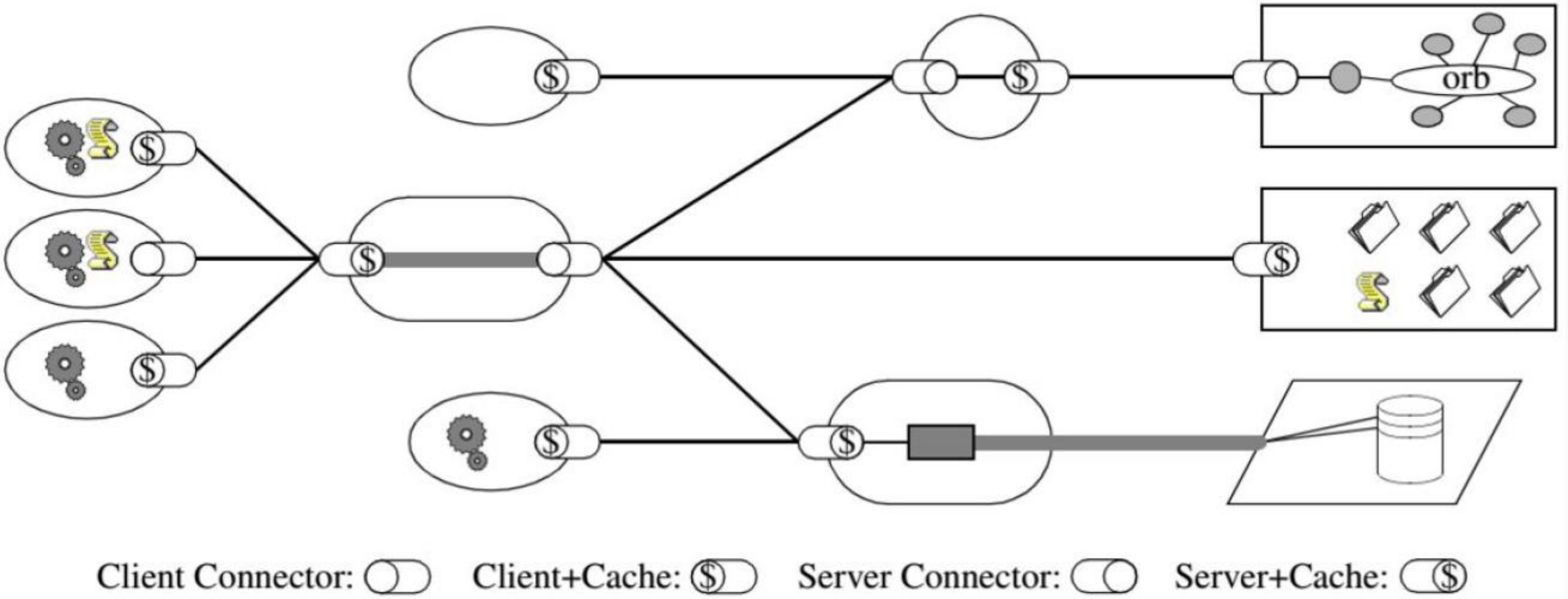
- Fielding (2000)

REST: RESTful

- **Aplikasi** yang **dibangun mengikuti ketentuan REST** disebut **RESTful**.
- Sistem RESTful biasanya berkomunikasi di atas **HTTP** dengan metode sama (**GET**, **POST**, **PUT**, **DELETE**, dll) yang digunakan web browser untuk me-retrieve halaman web dan mengirimkan data ke server *remote*.
- Sistem REST menyediakan antarmuka dengan sistem eksternal sebagai sumber daya web yang **diidentifikasi** dengan **URI**. Contoh: Operasi dapat menggunakan metode standard adalah seperti **GET /people/1** atau **DELETE /people/1**.

- Fielding (2000)

REST Style (Fielding, 2000)



Microservices

Microservices: Konsep Inti

- Mungkin konsep yang paling penting untuk dipahami dengan pola ini adalah gagasan dari komponen layanan (*service component*).
- **Komponen** layanan **mengandung satu** atau **lebih modul** (mis. kelas Java) yang mewakili suatu fungsi bertujuan tunggal (mis. memberikan informasi cuaca untuk kota/daerah tertentu) atau bagian terpisah dari aplikasi bisnis besar (mis. Informasi posisi perdagangan saham atau penentuan tarif asuransi mobil).

- Richards (2015)

Microservices: Unit-unit diserahkan terpisah

- Aplikasi yang dibangun menggunakan pola arsitektur layanan mikro (*microservices*) umumnya lebih kuat, menyediakan skalabilitas yang lebih baik dan dapat lebih mudah mendukung pengiriman berkelanjutan (*continuous delivery*).

- Richards (2015)

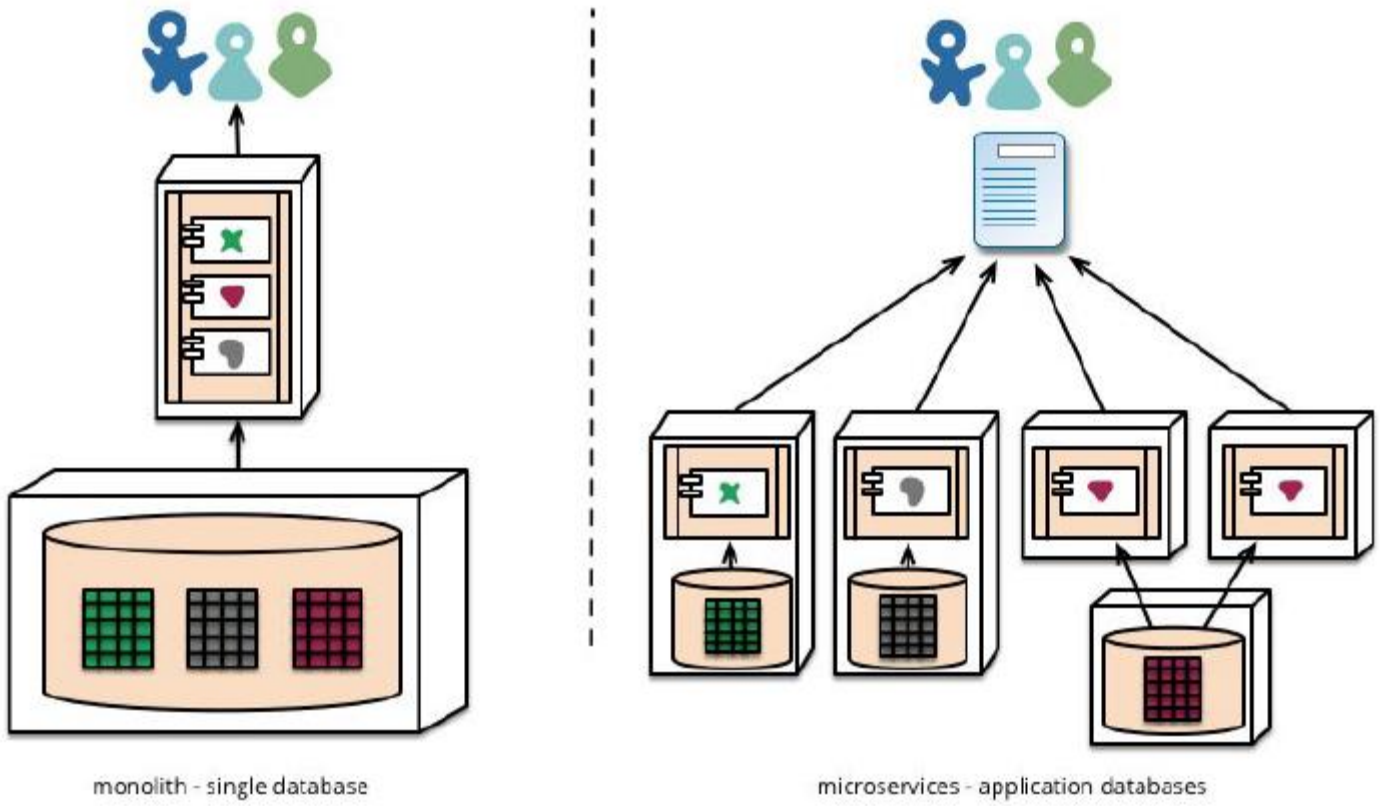
Microservices: Arsitektur Terdistribusi

- Konsep kunci lain dalam pola arsitektur *microservices* adalah **arsitekturnya terdistribusi**, berarti bahwa semua **komponen dalam arsitektur sepenuhnya terpisah** (*decoupled*) dari satu sama lain dan diakses melalui protokol akses remote (misalnya, JMS, AMQP, REST, SOAP, RMI, dll).

  Tujuan "distribusi" dari pola arsitektur ini adalah untuk memperoleh skalabilitas dan karakteristik penyebaran yang tinggi (*superior*).

- Richards (2015)

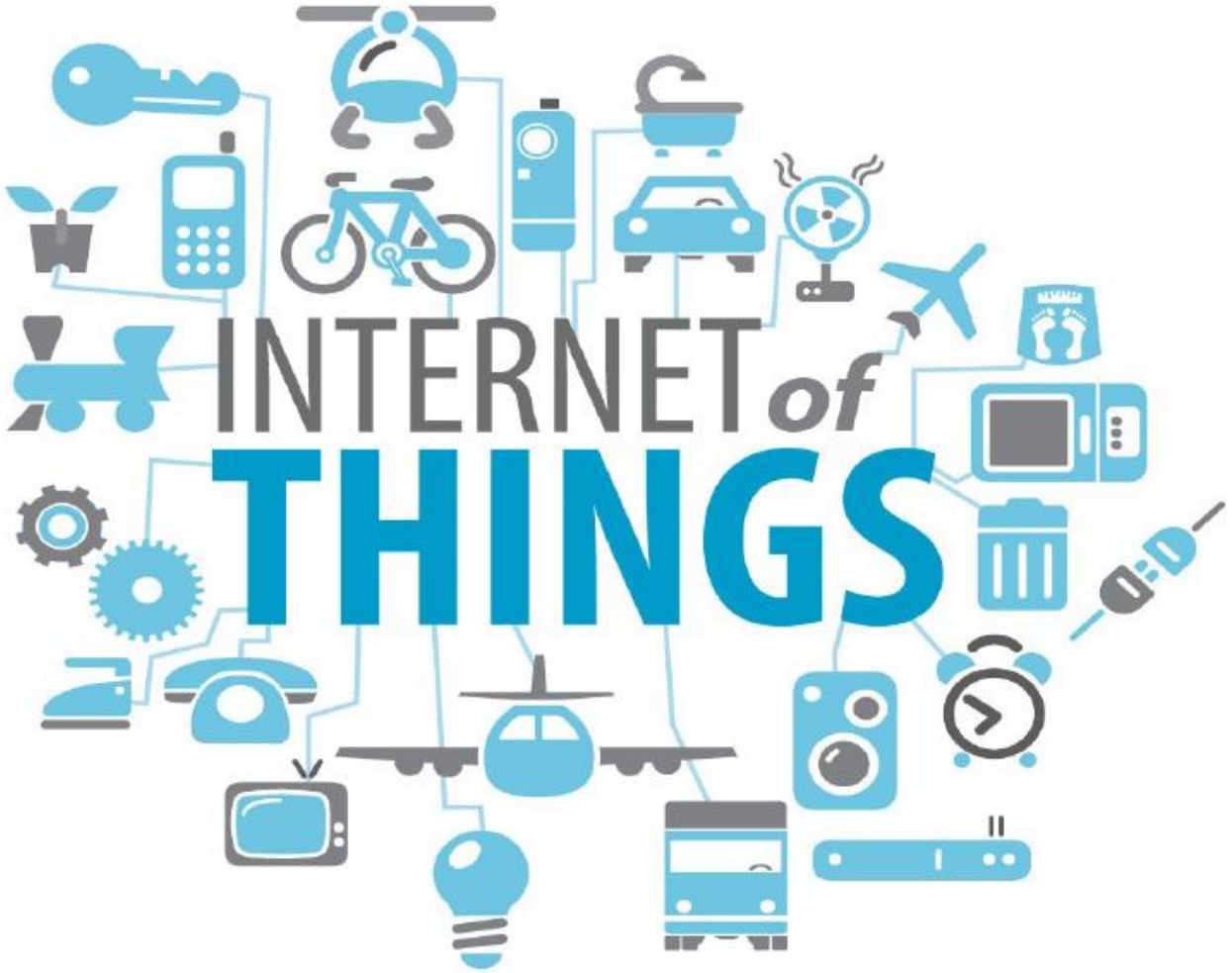
Microservices: Contoh



- Martin Fowler (2014)

Internet of Things (IoT)

IoT telah menjadi teknologi pelopor utama yang mengubah software – dan masyarakat (Weyrich and Ebert, 2016)



Sejarah IoT

- Istilah "Internet of Things" (IOT) pertama kali digunakan pada tahun 1999 oleh Kevin Ashton yang bekerja pada standar penandaan (*tagging*) objek menggunakan RFID untuk aplikasi logistik.
- Namun, ide komputasi di mana-mana (*ubiquitous computing*) sudah ada sejak akhir 1980-an.
- Sejak itu, para peneliti telah bekerja pada banyak sistem yang fokus pada *tag* dan sensor, *middleware* dan teknologi awan, dan jaringan komunikasi.

- Weyrich and Ebert (2016)

Apa itu IoT?

- Internet of Things (IOT) berurusan dengan fungsi inovatif dan produktivitas yang lebih baik dengan menghubungkan perangkat-perangkat dengan lancar (*seamlessly*).
- Akan meningkatkan sejumlah besar inovasi, efisiensi, dan kualitas.
- Menghubungkan bidang produksi, kesehatan, otomotif, atau sistem transportasi dengan sistem TI. Informasi bisnis penting akan memberikan nilai yang sangat besar untuk organisasi.
- Perusahaan IT besar seperti Cisco dan SAP memperkirakan ada miliaran perangkat jaringan dan layanan bisnis alam semesta berbasis TI, dengan ekspektasi triliun dolar.

- Weyrich dan Ebert (2016)

IoT: Marketplace

- Sebuah tonggak yang sangat terlihat dicapai ketika IETF merilis IPv6, protokol yang memungkinkan IOT.
- Baru-baru ini, IOT telah menerima usulan keterlibatan dan kerja komersial pada referensi arsitekturnya. Ini didukung oleh industri-industri besar :
  - **Google** mengumumkan **Brillo** sebagai **OS** untuk perangkat IoT dalam rumah pintar **(*smart home*)**;
  - Banyak perangkat komersial yang tersedia telah menggunakan standard komunikasi ***machine-to-machine*** (**M2M**) seperti **Bluetooth**, **ZigBee** dan **low-power Wi-Fi**.

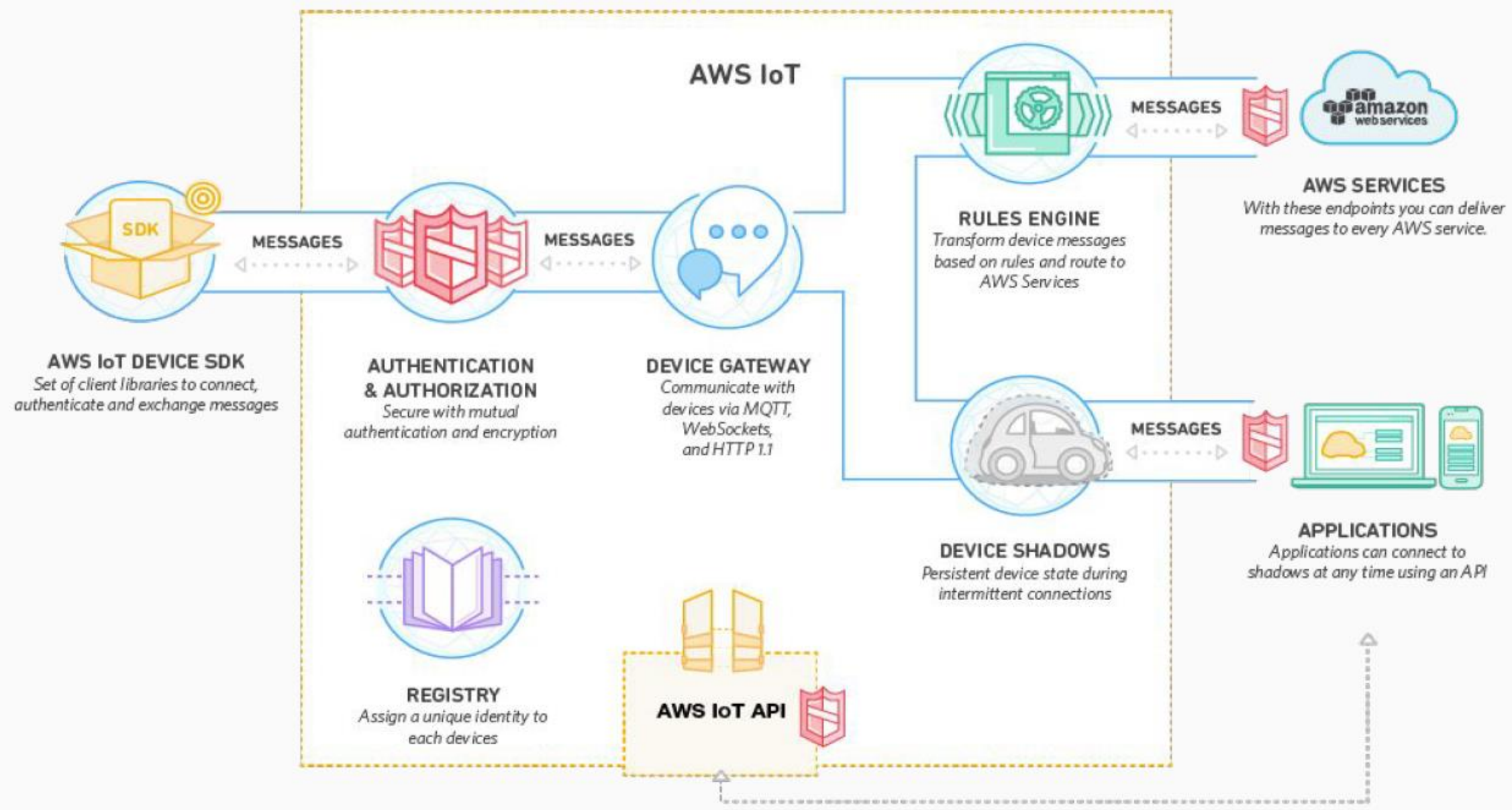
- Weyrich dan Ebert (2016)

IoT: Marketplace

- Microsoft telah mengumumkan bahwa Windows 10 akan mendukung sistem *embedded* untuk mikro-kontroler yang beredar luas seperti Raspberry Pi 2 dan 3;
- Samsung dan perusahaan lainnya telah mengumumkan generasi baru chip mereka untuk perangkat pintar;
- Telah banyak laporan implementasi menjelaskan bagaimana mikro-kontroler jaringan yang difungsikan sebagai hub untuk sensor, aktuator, dan penandaan (*tagging*).
- Pada tahun 2015 Amazon merilis layanan cloud untuk IOT, AWS IOT.

- Weyrich dan Ebert (2016)

AWS IoT


AWS IoT
SDK
MESSAGES
MESSAGES
MESSAGES
MESSAGES
amazon web services
AWS IoT DEVICE SDK
Set of client libraries to connect, authenticate and exchange messages
AUTHENTICATION & AUTHORIZATION
Secure with mutual authentication and encryption
DEVICE GATEWAY
Communicate with devices via MQTT, WebSockets, and HTTP 1.1
RULES ENGINE
Transform device messages based on rules and route to AWS Services
AWS SERVICES
With these endpoints you can deliver messages to every AWS service.
DEVICE SHADOWS
Persistent device state during intermittent connections
APPLICATIONS
Applications can connect to shadows at any time using an API
REGISTRY
Assign a unique identity to each devices
AWS IoT API

Referensi

Referensi

- **Bass** L, Clements P, and Kazman R. 2013. **Software Architecture in Practice**. Addison-Wesley, Boston, MA.
- **Clements** P, Bachmann F, Bass L, Garlan D, Ivers J, Little R, Merson P, Nord R, and Stafford J. 2010. **Documenting Software Architectures: Views and Beyond**. Addison-Wesley, Boston, MA.
- **Fielding** R. 2000. **Architectural Styles and the Design of Network-Based Software Architectures**. Ph.D. Dissertation. University of California, Irvine, CA.
- **Richards** M. 2015. **Software Architectural Patterns**. O'Reilly Media, Sebastopol, CA.
- **Fowler**, M. **Microservices: A definition of this new architectural term**. 2014. Available in: http://martinfowler.com/articles/microservices.html.
- **Weyrich** M and **Ebert** C. 2016. **Reference Architectures for the Internet of Things**. IEEE Software, vol. 33, no. 1, pp. 112-116.

Terima kasih ☺

- husni@trunojoyo.ac.id
- Husni.trunojoyo.ac.id