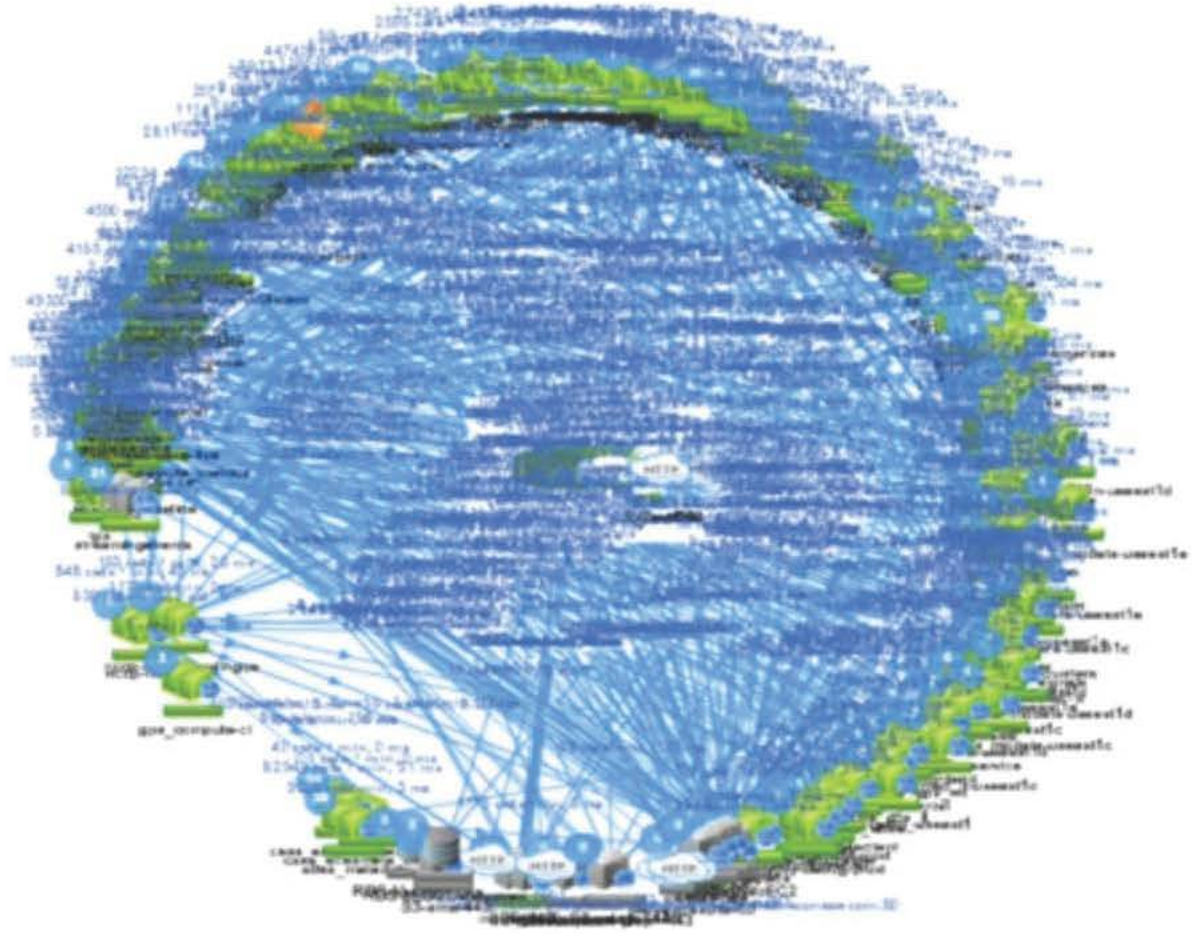
IoT: Cloud Services

IoT pushing the limits of SOA to microservices

IoT and Microservices

- IoT is gaining a lot of attention and the Cloud platform has more requirements.
- Big Data became commonplace and the world started moving towards the API economy.



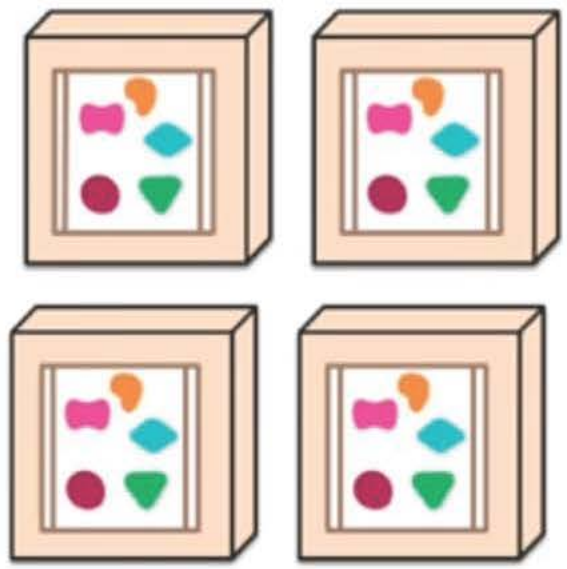
Classic SOA

- This is where the Classic SOA started showing problems.
    - too complicated, with hundreds of interfaces and impossible to define granularity.
- The Microservices architecture adds agility to SOA and brings the much needed speed and efficiency when it comes to deployment and modification of systems.
- Microservices can define the size of a service and the manner in which they talk to other services.
- Microservice architecture brings in smaller services and lightweight protocols.
- The principle of the Microservices architecture is akin to the Unix principle “Do one thing and do it well”.

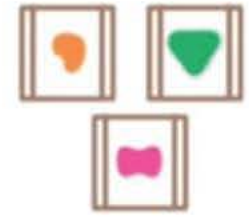
Microservice style vs Monolyth

A monolithic application puts all its functionality into a single process...

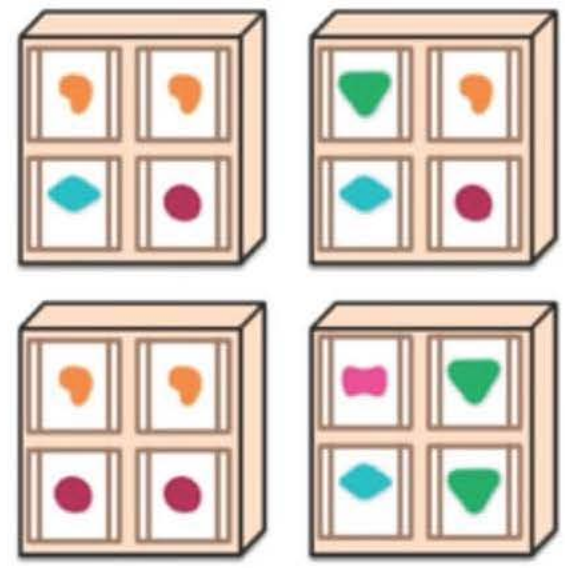
... and scales by replicating the monolith on multiple servers



A microservices architecture puts each element of functionality into a separate service...

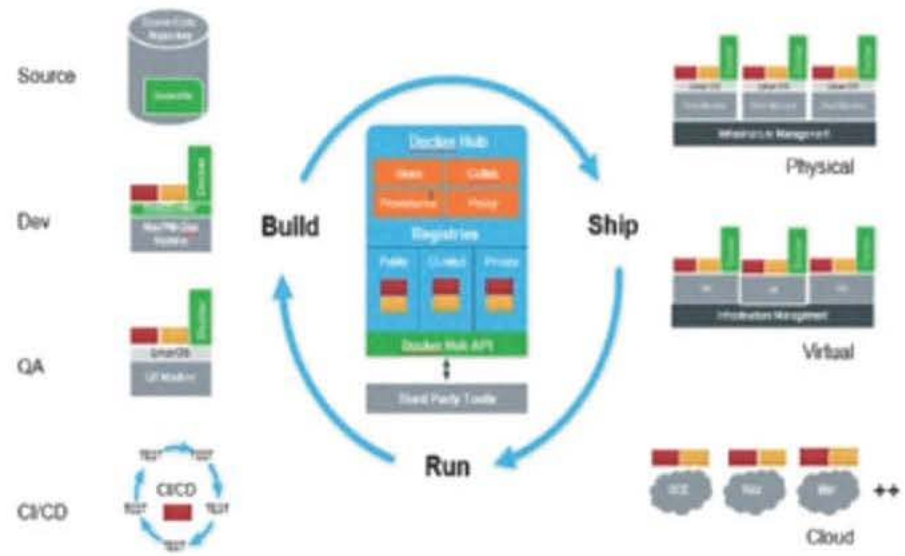


... and scales by distributing these services across servers, replicating as needed.

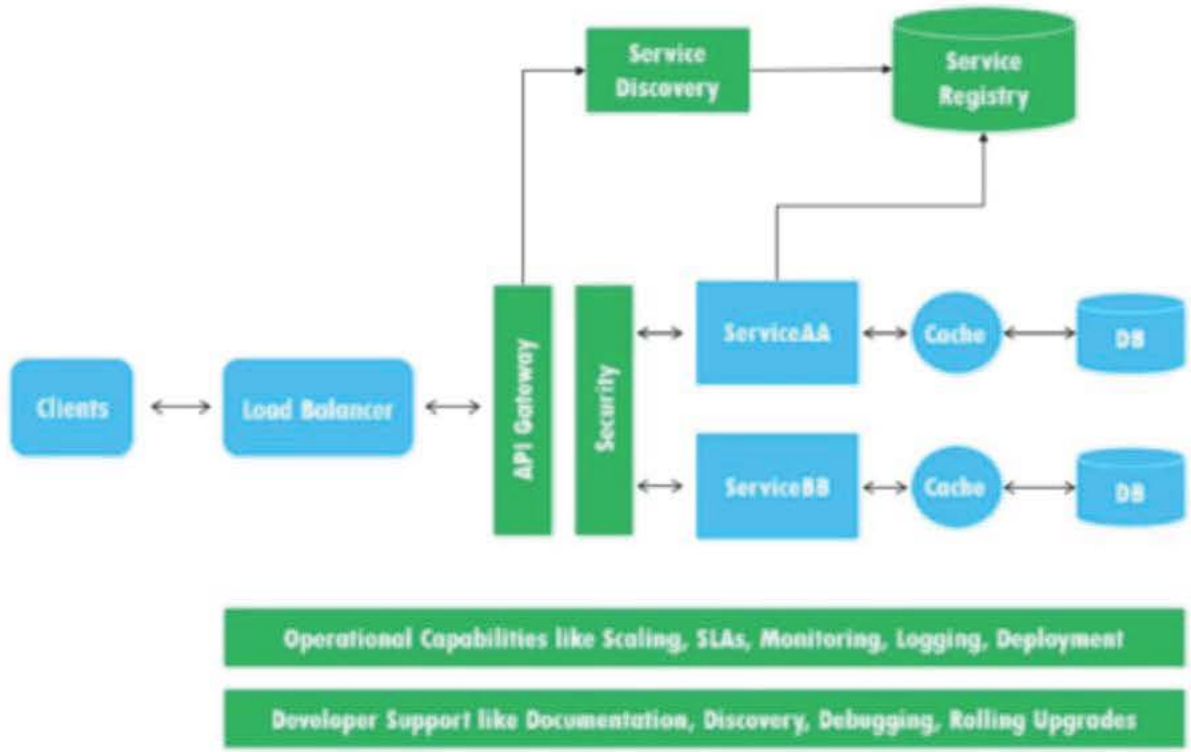


IoT & Microservices

- **Microservice architectural style:** developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, e.g. REST API.
- Microservices also offer a way of scaling the infrastructure both horizontally and vertically giving long term benefits to the IoT deployments. Each of the services can scale based on the needs.
- Given the dynamism of deployment and scalability expectations which comes with IoT, Microservices need to become an important part of the overall IoT Strategy.



Microservices Challenges



- **Distributed application logic:** with microservices the logic is spread across the services and, more importantly, embedded in the control and data flow between those services.
- **Diverse technology stack**: the system may be comprised of in house developed services, open source software, and external libraries.
- **Hard to test and debug**: there might be thousands of interactions between the constituent services
  - Equivalent to the “butterfly effect” (a minor change of service, could potentially be catastrophic)