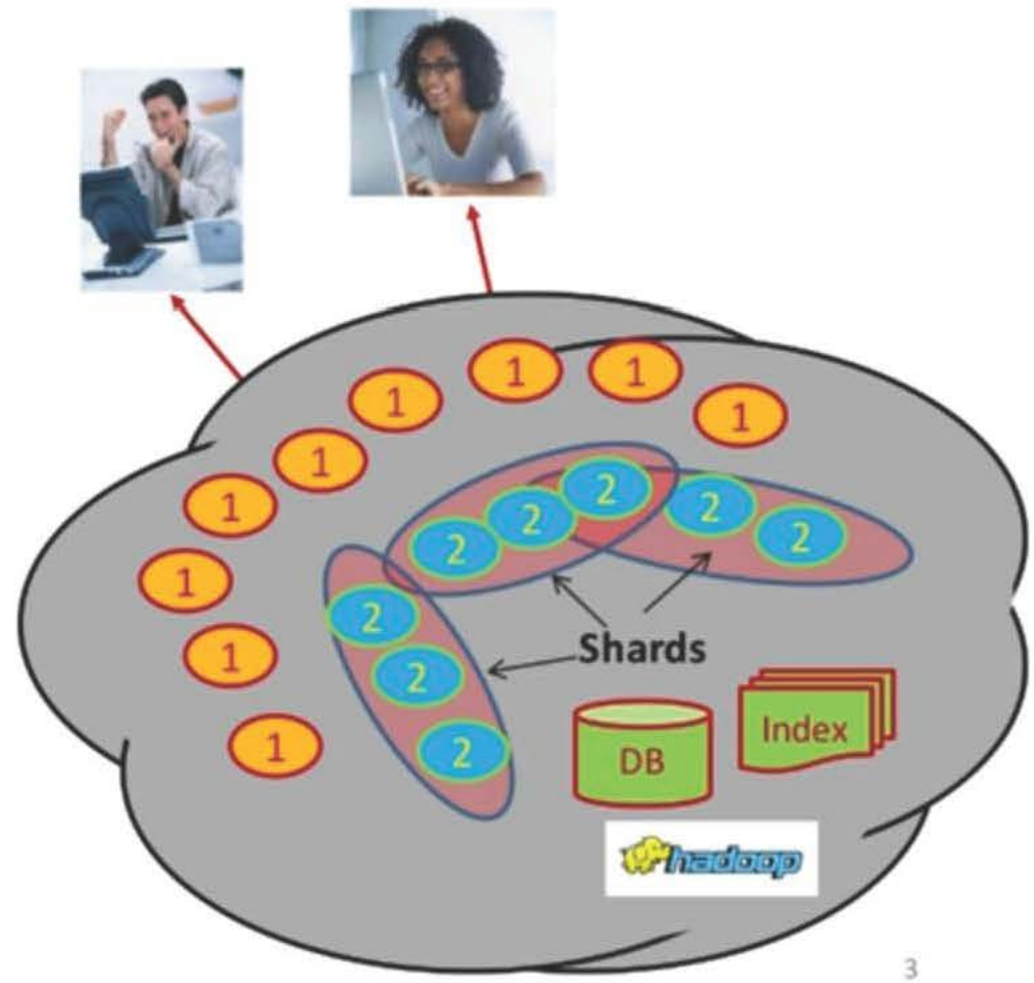
IoT: Cloud Services

Cloud Edge Services

IoT Apps talking to Cloud

- The modern cloud is comprised by a number of services called micro-services
  - many times in a multi-tiered architecture
- Client devices (IoT sensors, smartphones, etc) talk to the Edge of the Cloud
  - The **edge services** are simple, lightweight, and nimble



IoT Cloud Edge

- Cloud services for IoT must provide per-device authentication and access control.

- Cloud Edge must route messages to a microservice endpoint based on message properties.

- We will talk later about micro services and IoT messages.

- Routing rules give the flexibility to send messages where they need to go without the need to stand up additional services to process messages or to write additional code.

Edge services

- Near the edge of the cloud focus is on serving a vast number of clients in the fastest possible way
  - Caching content at different layers helps
  - Stateless

- Inside we find high volume services that operate in a pipelined manner, asynchronously

- Deep inside the cloud we see a world of virtual computer clusters that are scheduled to share resources and on which applications like MapReduce/Hadoop/Spark are very popular
  - Applications: A/B testing

- In the bottom of the Cloud is where the data are being stored

Cloud Edge evolution

- The Cloud API is the way the IoT services interact with the cloud. Hence background upgrades/updates should not affect the communication.
- This can be achieved with a common well-defined and extensible Cloud API
- A robust edge service must enable
  - rapid development
  - great flexibility
  - expansive insights
  - resiliency

Cloud Edge Features to support IoT

- Authentication
- Insights
- Stress Testing
- Canary Testing
- Dynamic Routing
- Load Shedding
- Security
- Static Response handling
- Multi-Region Resiliency

Cloud Edge Insights

- A service that allows us to shed and prioritize traffic when issues occur.

- Detailed information into network performance and errors, as well as handles software load balancing for even load distribution.

- Fine-grained metrics in real-time so that we can quickly observe and react to problems.

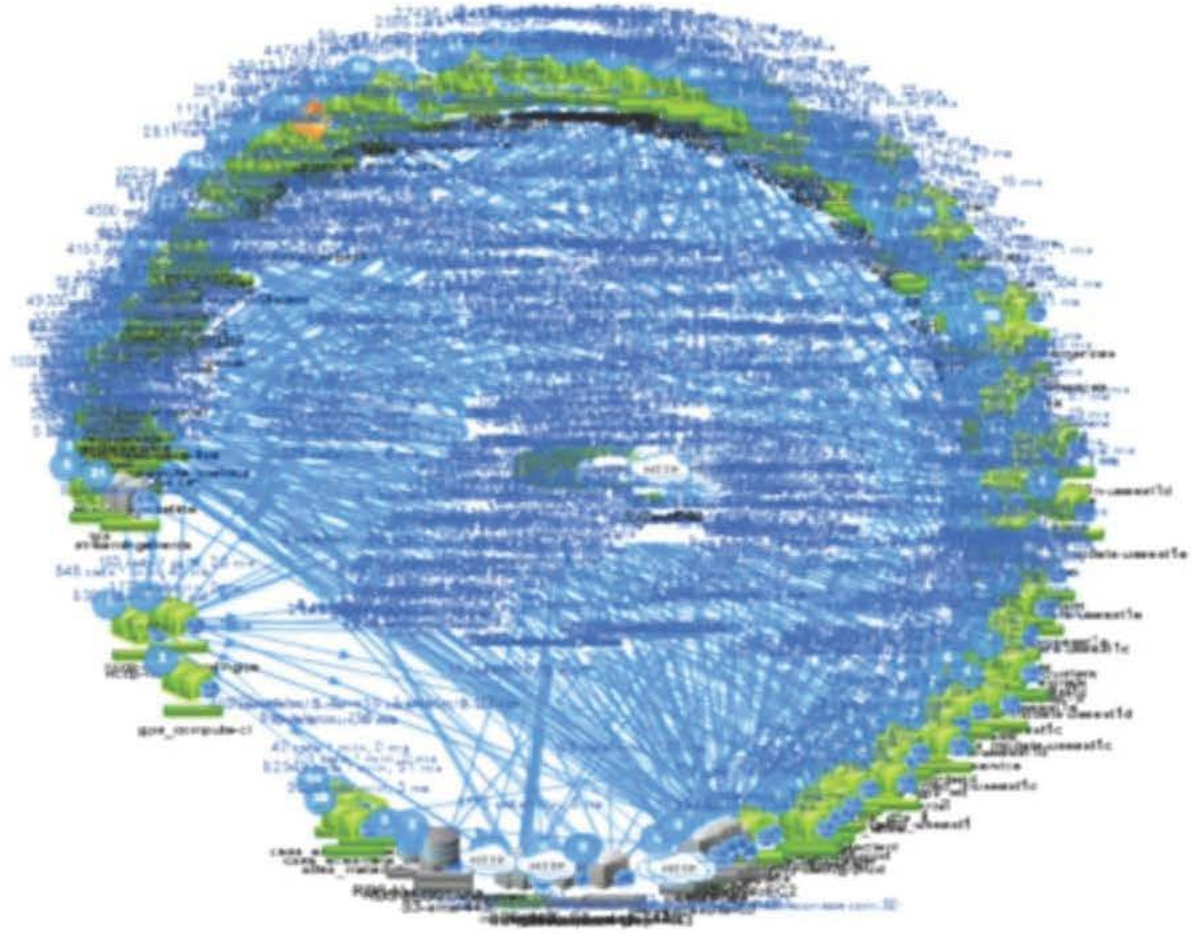
- Dynamically change properties

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.



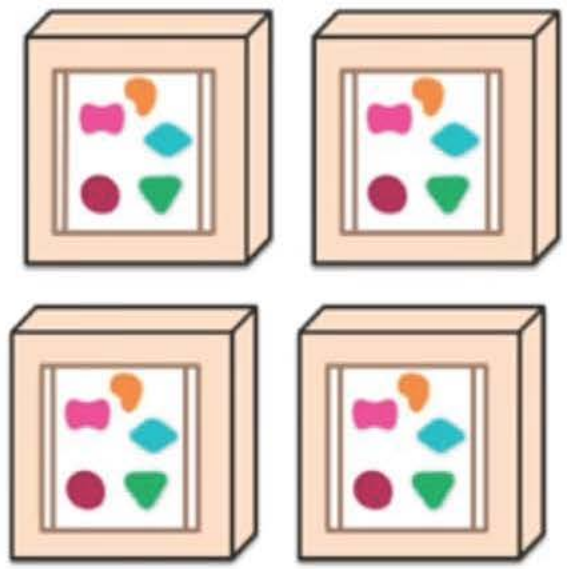
Classic SOA

- 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”.

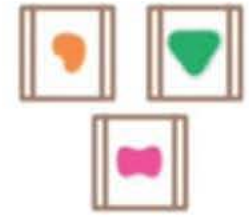
Microservice style vs Monolyth

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

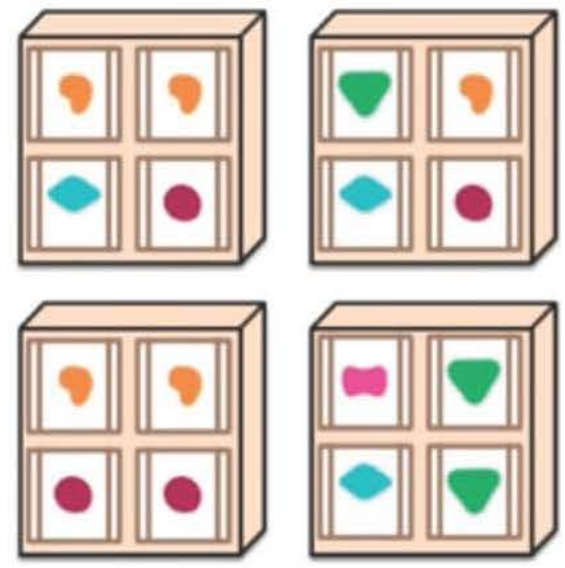
... and scales by replicating the monolith on multiple servers



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

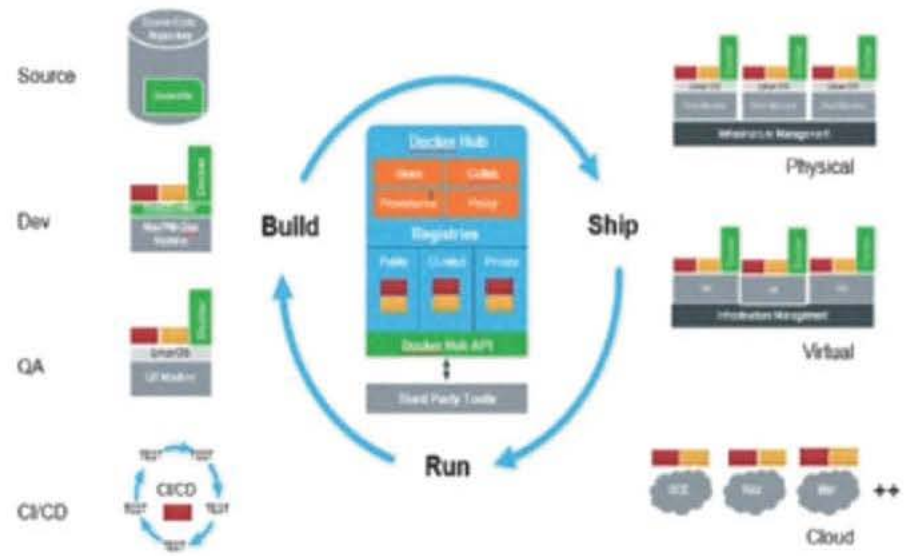


... and scales by distributing these services across servers, replicating as needed.

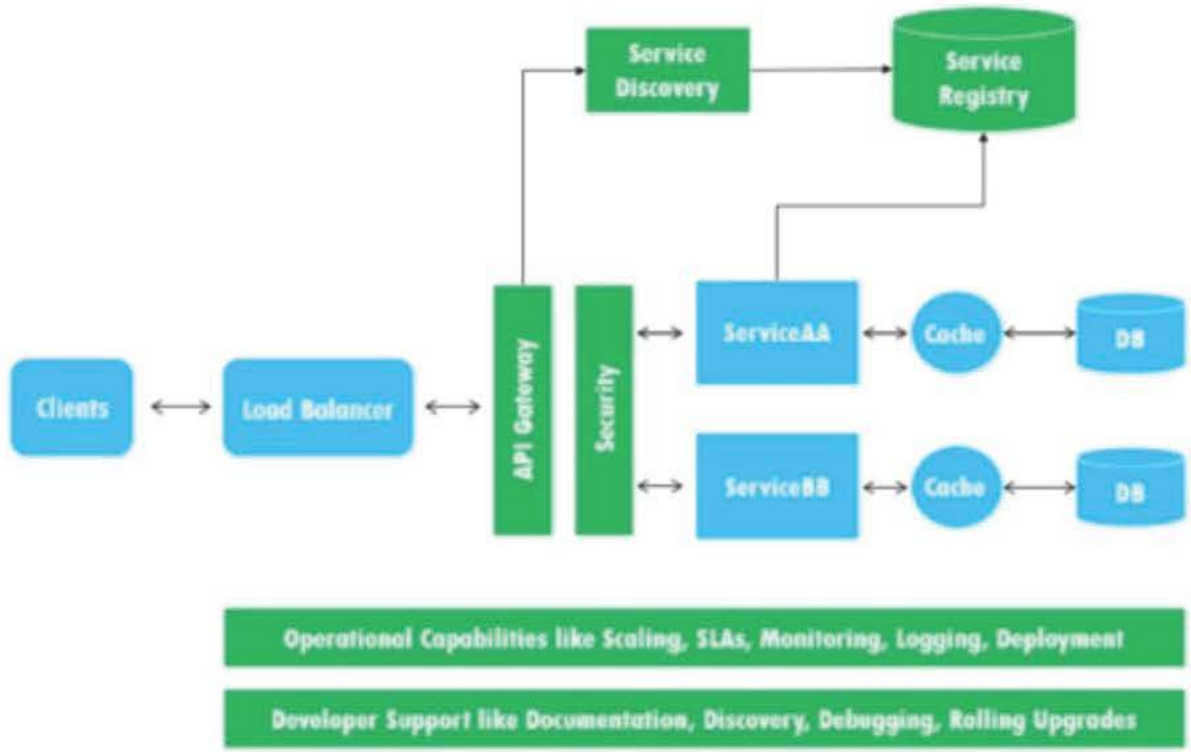


IoT & Microservices

- **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)

IoT: Cloud Services

Why Microservices?

So why Microservices?

- **Fast deployment:** Code changes for each service can be made independently (as long as the API contract with other services isn't violated) and therefore, build+test+deploy cycles speed up dramatically.
  - Netflix, for example, deploys code a hundred times in a single day thanks to the early adoption of the microservices architecture!
- **Efficient scaling:** Each microservice can be scaled independently,
  - a much more efficient way of scaling an application, because not every part of an application experiences the same amount of load and needs to be scaled equally.
- **Design autonomy:** Developers get the freedom to employ different technologies, frameworks, and design patterns to design and implement each microservice,
  - pursuing a *horses for the courses* strategy as necessary.

How can IoT exploit Microservices

- The monolithic IT architectures do not align with an environment where every device has a computer and wireless connection.

- Example:

- In my own house, I may have 7 light bulbs and 10 meters of light strips that all have their own processors, plus an Apple TV and an XBox.

- And how I want to interact with them is probably different than what you would want to do.

- This requires some level of decoupling: I want devices announcing themselves and reacting to the actions of other devices

- A fuzzy problem, a domain ideally suited for Microservices.

IoT/Microservices

- In the previous example:
    - I envision a MicroService that simply indicates whether I am home or away (probably via my iPhone and it geo-location services).
        - Another Microservice reacts to that and, based on the time of day, turns lights on or off (via Apple HomeKit and my Philips Hue controller).
- With Microservices, I can keep adding a bit more sophistication through additional services without waiting for one of the big vendors to build an application with that feature.

Challenges of IoT and Microservices

- **Interoperability**. How do I get devices from various vendors working with each other?

- **Security**. How do I protect access to systems in my house from malicious strangers (or hacker acquaintances, in my case)?

Dealing with the challenges

- **Interoperability:** there are a couple of standards emerging, and key vendors recognize that if their hub supports multiple interoperability standards, it is more likely to be used than a competitor's.

- **Security**: best addressed by using a locked, wireless network, and following best practices to secure it. Then you only need to protect the externally-facing facade from assault.

    - Apple TV, Microsoft XBox, and Amazon Echo all seem to be competing to be that facade.

Open APIs for IoT

- A successful vendor in this space will recognize the need for interoperability, and rather than solving it by successive features in proprietary products, will open up API's to allow rich communication to their hubs and devices over standard protocols (like RESTful interfaces with http protocols).
- A rich open source model is emerging
  - Much like some the toy robot and drone markets.
  - Market share is the reward for getting there first with open protocols.

Vendor Direction

- Recently, Amazon is pushing hard to be that integration vendor (see Amazon to flex internet of things).
- Philips just reversed a decision to exclude external devices from their Hub; the negative community reaction to exclusion was so vehement that Philips understood they would lose market share by such a move (see Philips Hue is getting back its third party smart bulb support). The industry seems to understand what they need to do.