11. Microservices

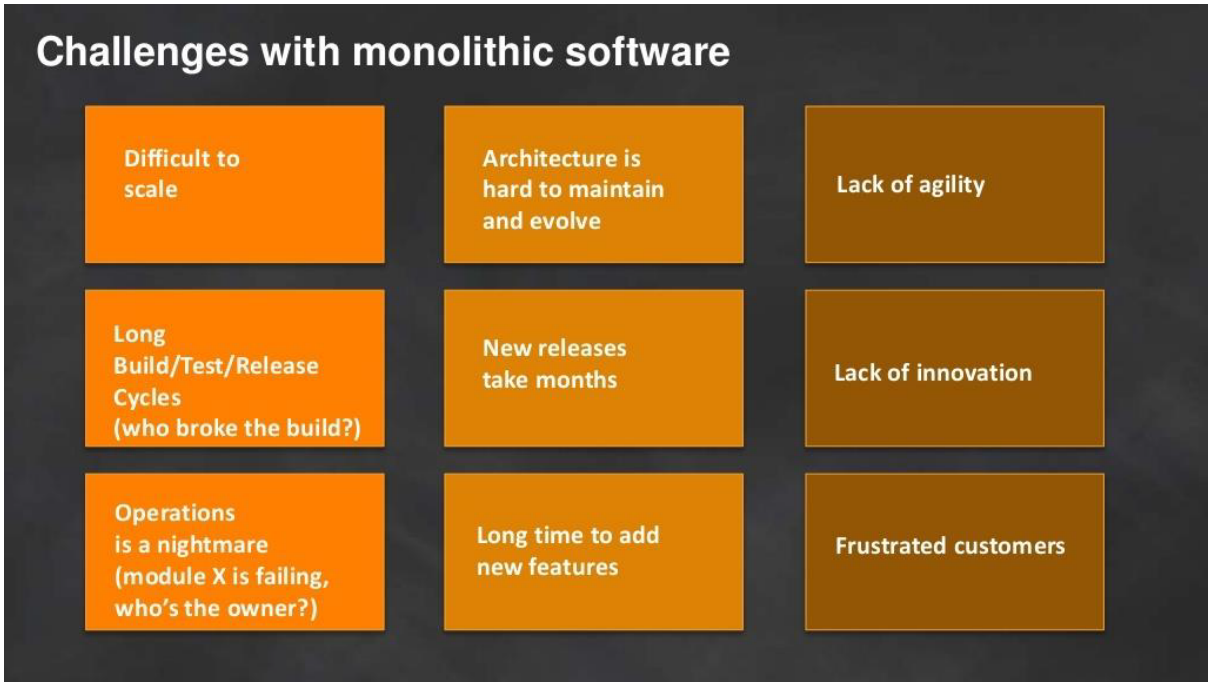
Current approach for service oriented architecture (SOA)

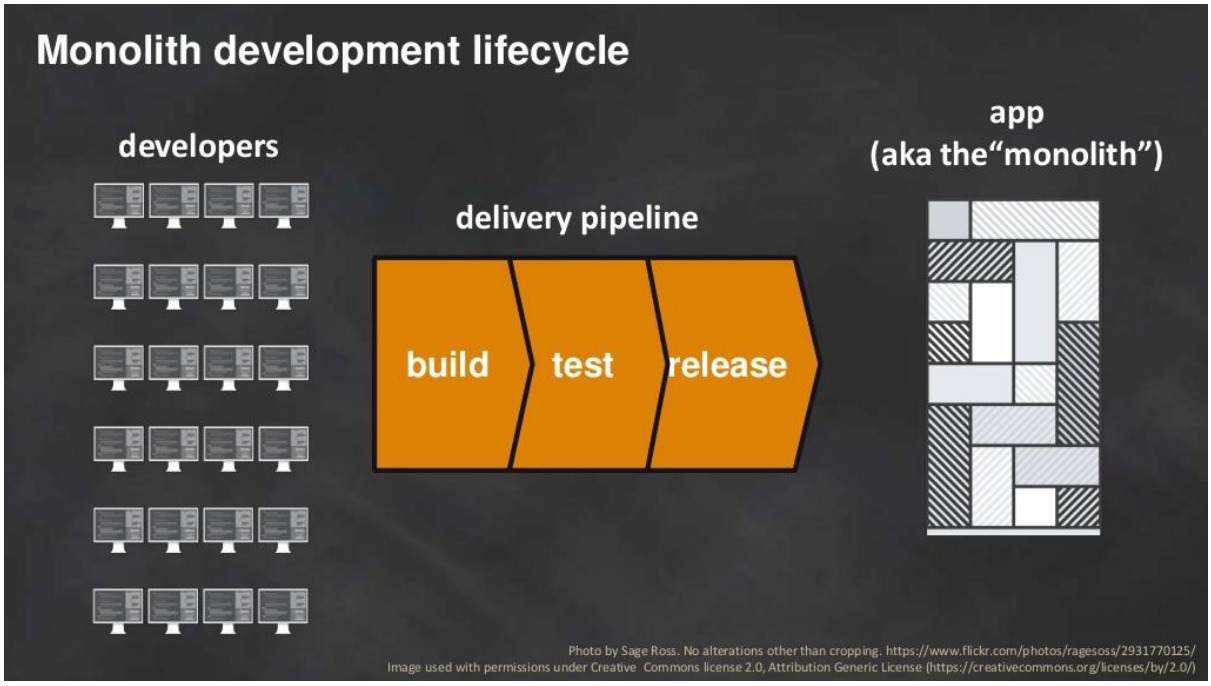
11.1 Monoliths to microservices

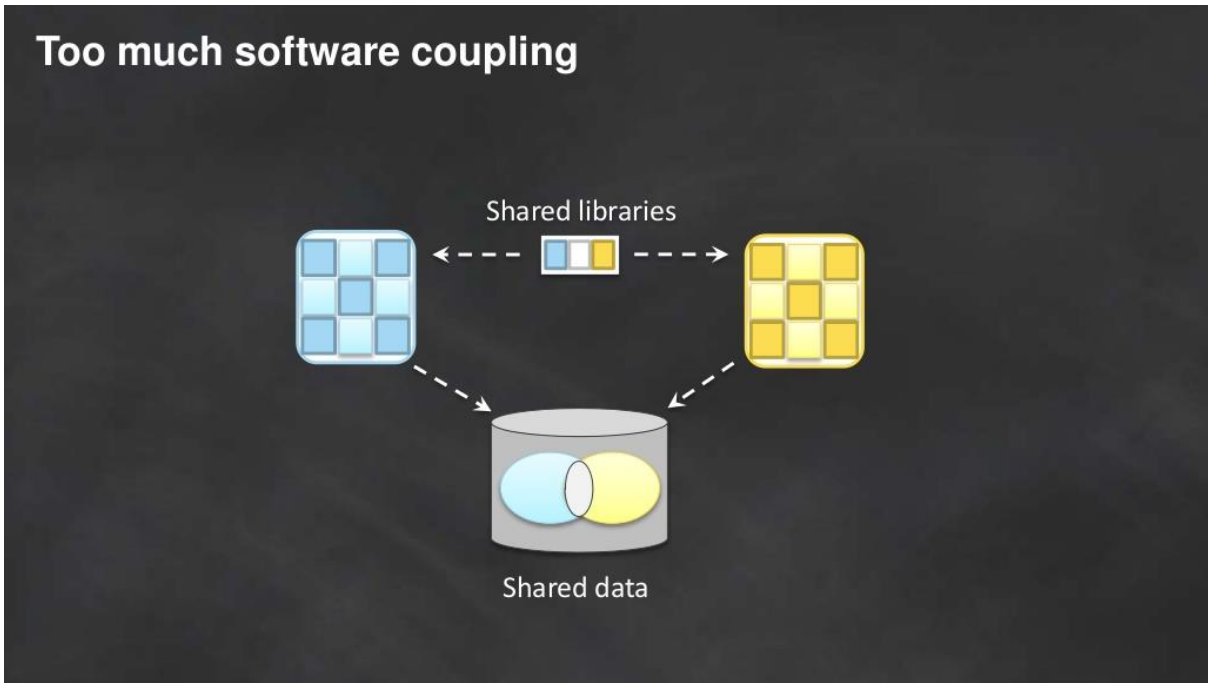
Components are packed toghether → monolith

Even if functionalities are divided, everything is in a block

Disadvantages:







You have tens of developers that are developing a single module, then the delivery pipeline, and the final app (the monolith).

In microservices are divided in modules indpendent on its own.

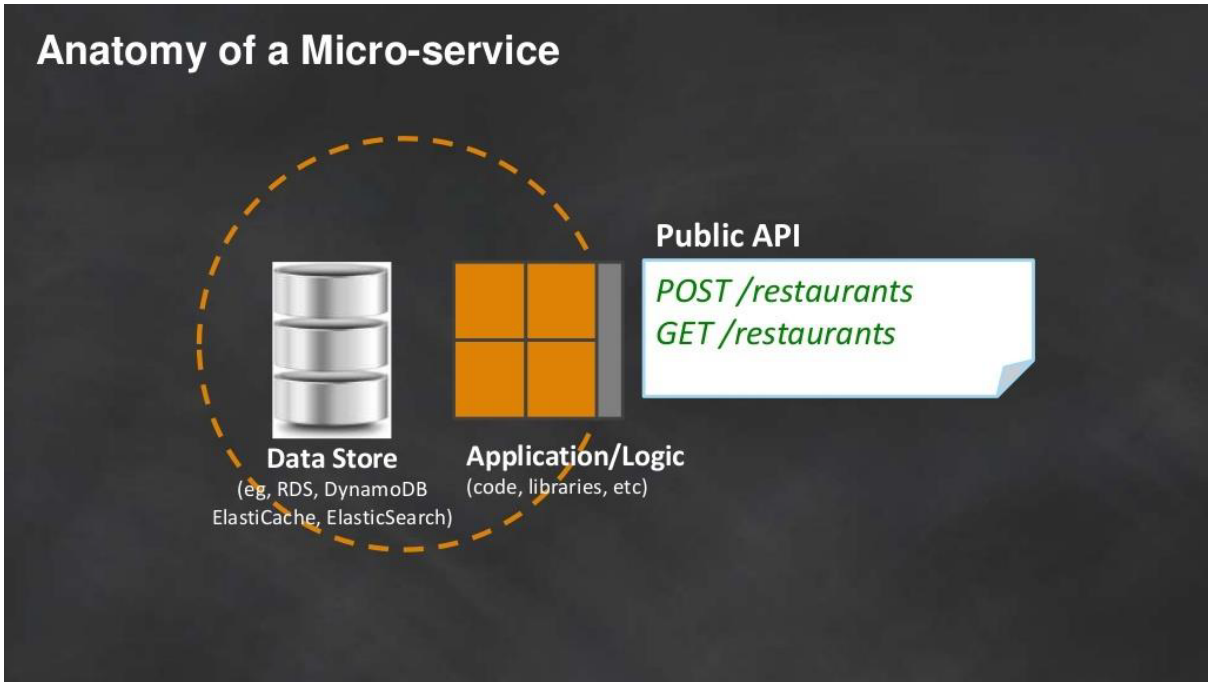
11.2 Microservices

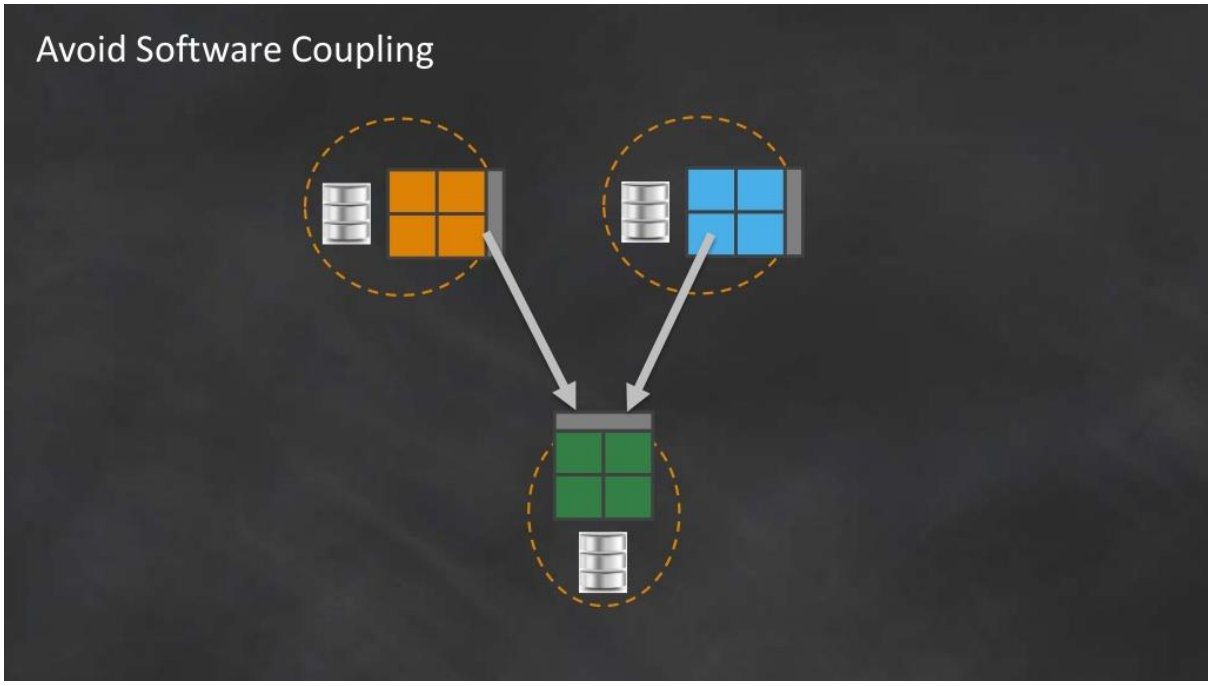
Microservices (def.): concrete and modern interpretation of SOA used to build distributed sw systems.

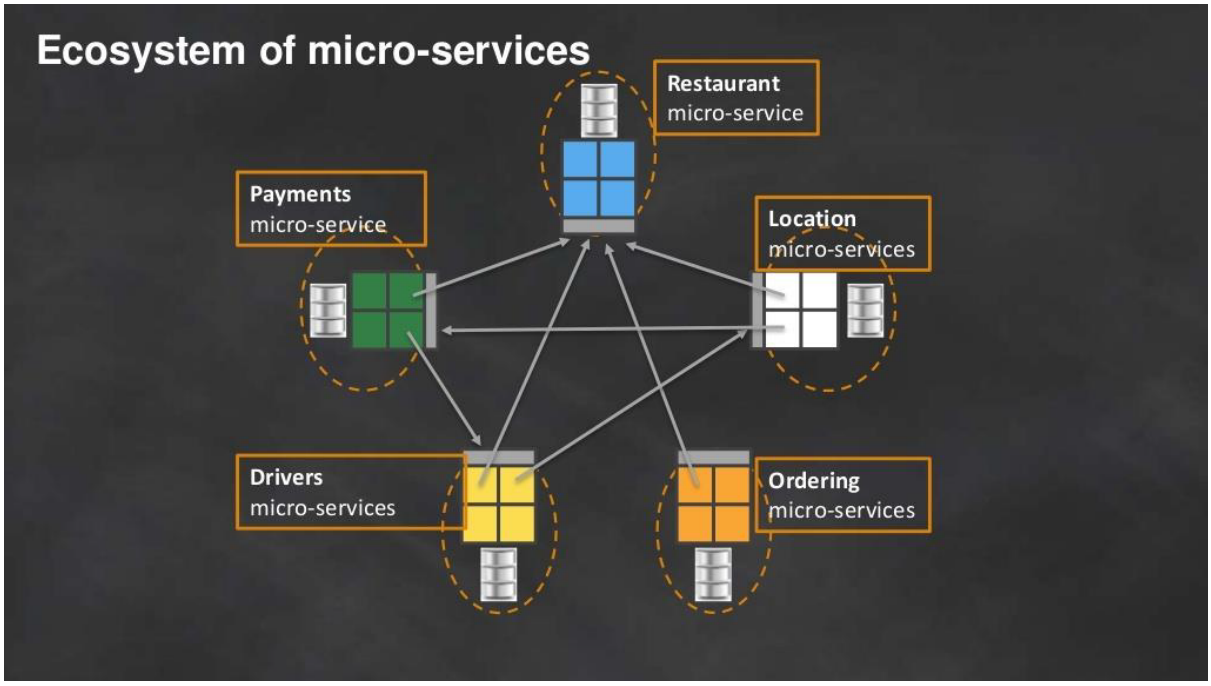
def.: service-oriented architecture (services communicate with each other over the network) composed of loosely coupled elements (you can update the services independently, updating one service doesn't require changing any other services) that have bounded contexts (self-contained: you can update the code without knowing anything about the internals of other microservices).

11.3 Anatomy of a micro-service

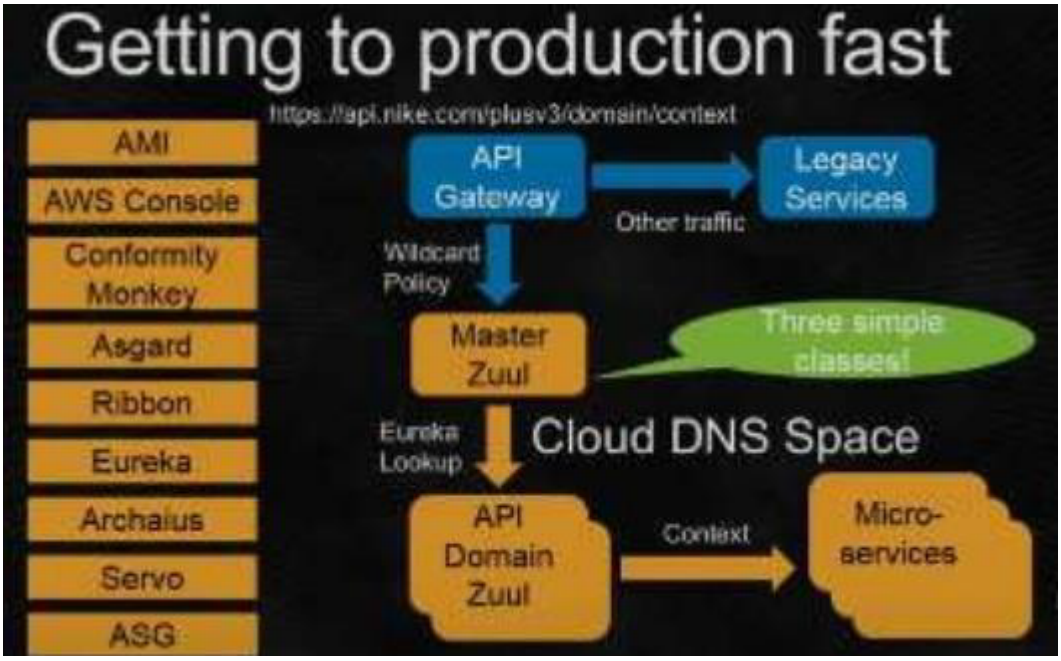
Avoid SW coupling → implies the creation of a ecosystem of micro-services





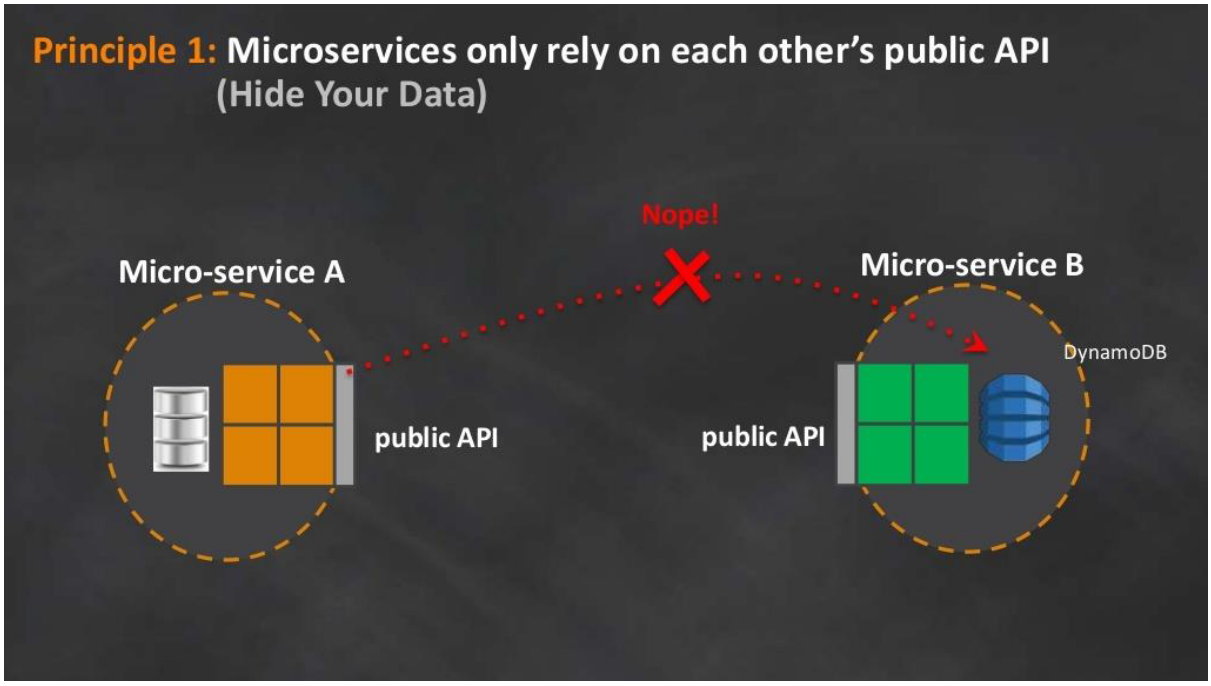


Example (Nike's journey to microservices)

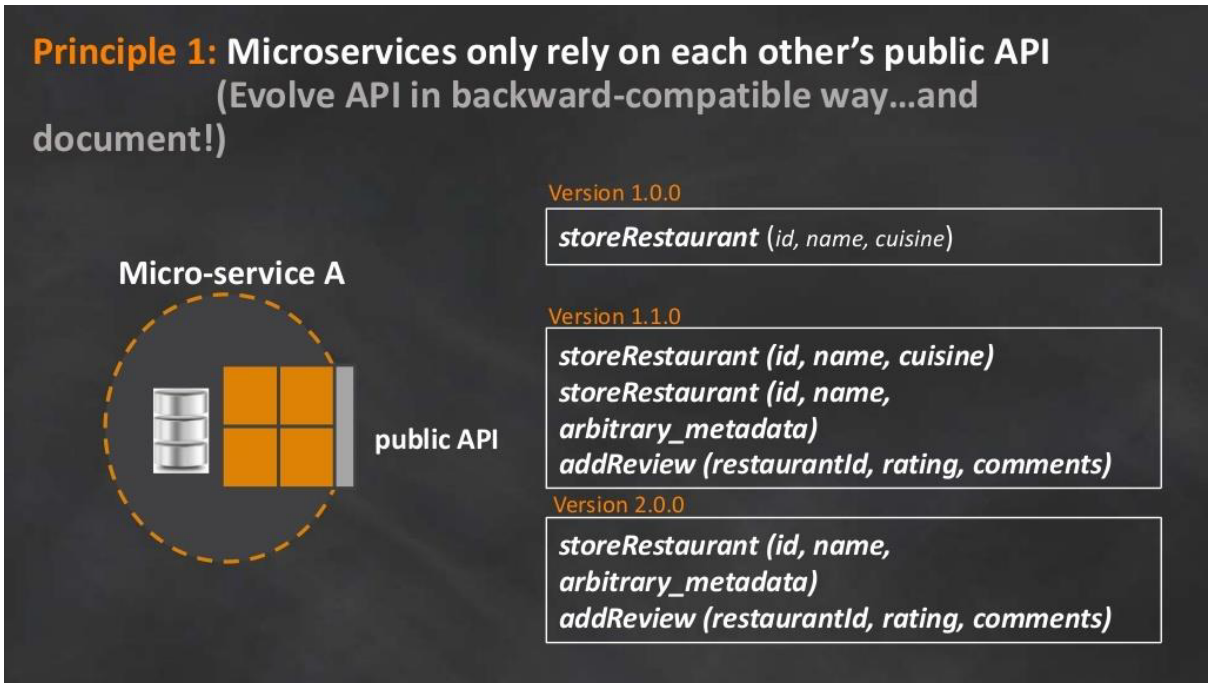


11.4 Principle 1

Micro-services only rely on each other's public API

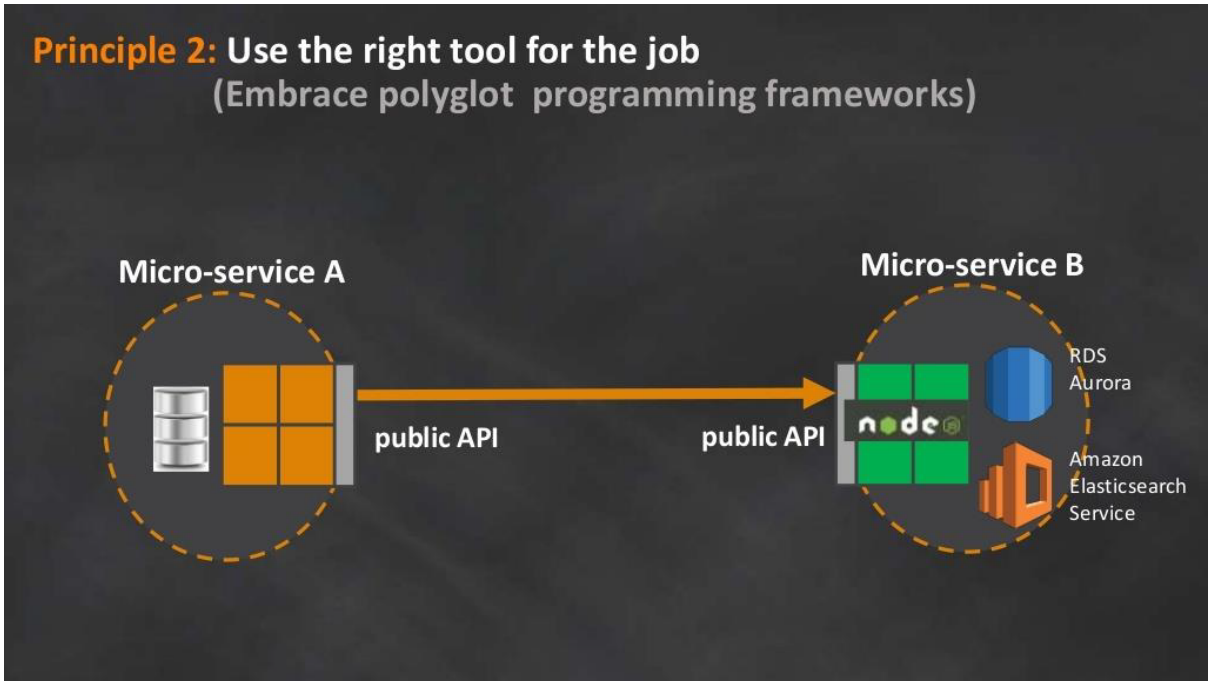


You can't access another microservices, but just interpolate their public API, so the content of the microservices are hidden from the external.



(you have to evolve API in backward-compatible way and document it)

11.5 Principle 2



Different microservices can use different technology, so:

every microservice can use its language, functionalities and paradigm

look for differences between sql and nosql.

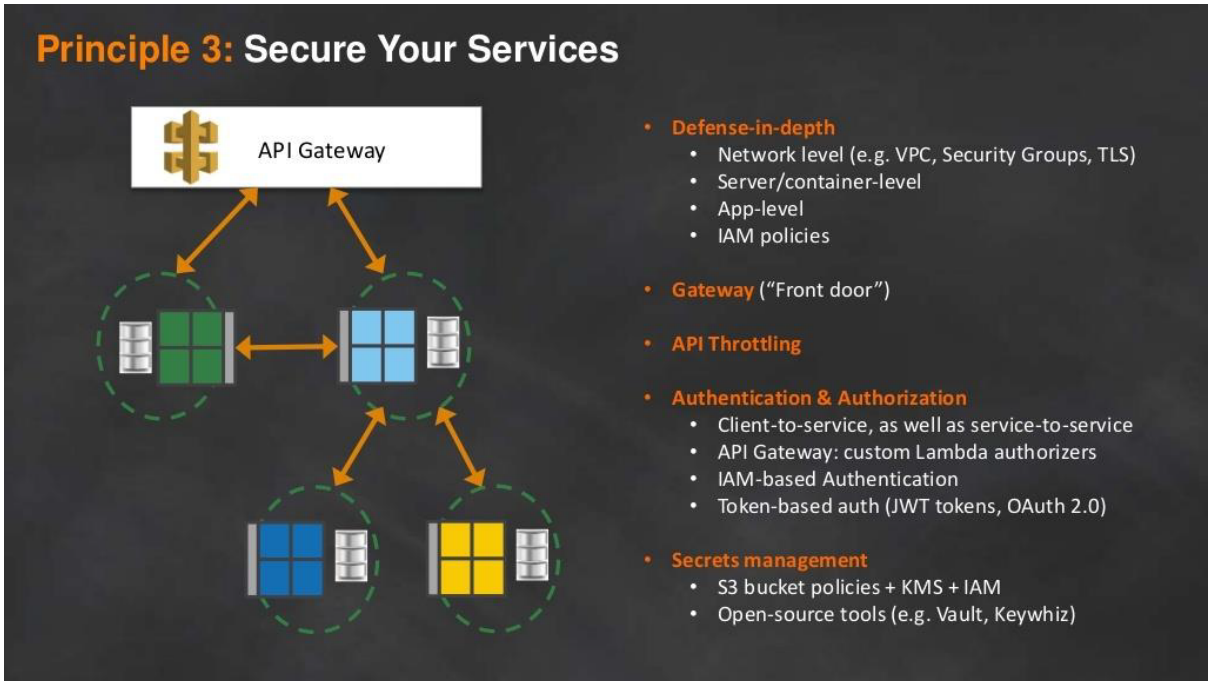
SEE EXAMPLE FROM P. 57

11.6 Principle 3

Secure your Services

when build complex, adopt defense-in-depth

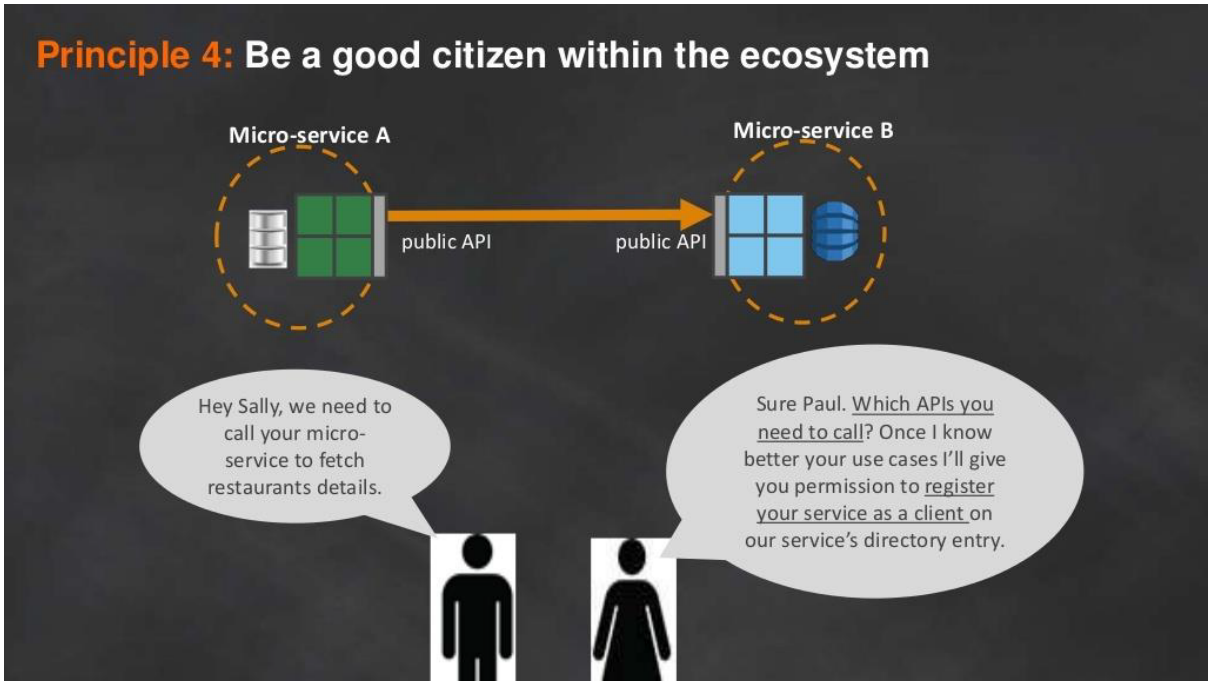
gateway (filter request from the external)

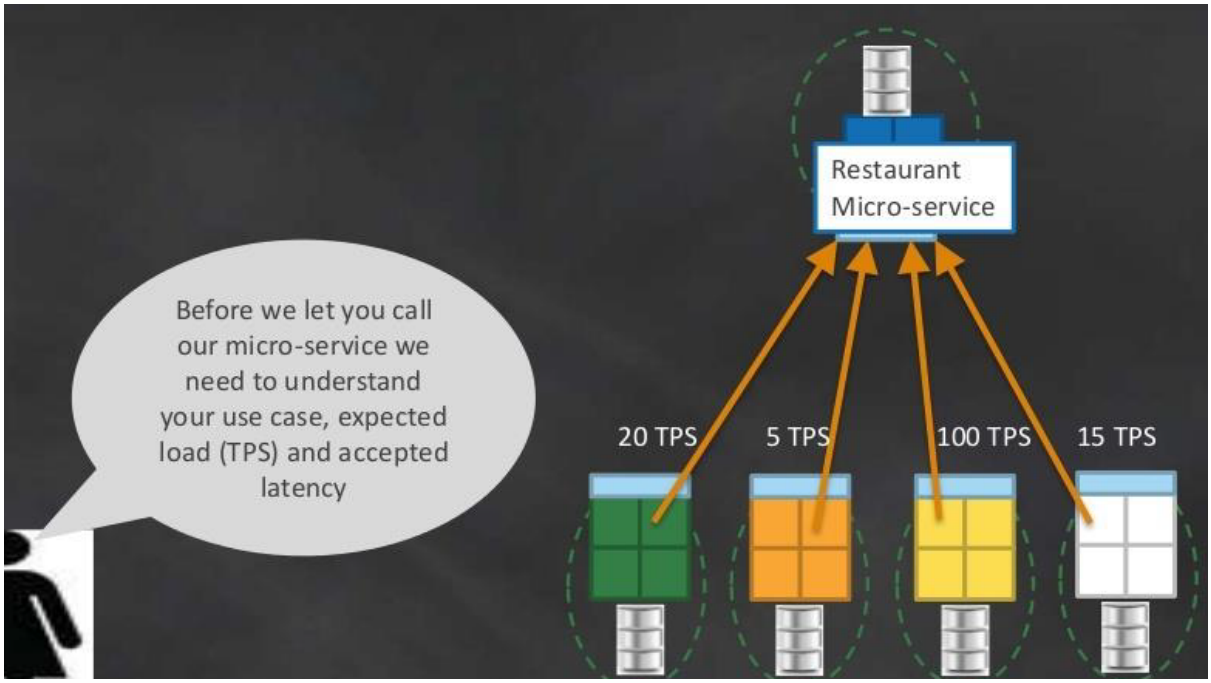


11.7 Principle 4

Be a good citizen within the ecosystem

Give SLAs to client who need resources, first we need to understand use case, expected load (TPS) and latency





11.8 Principle 5

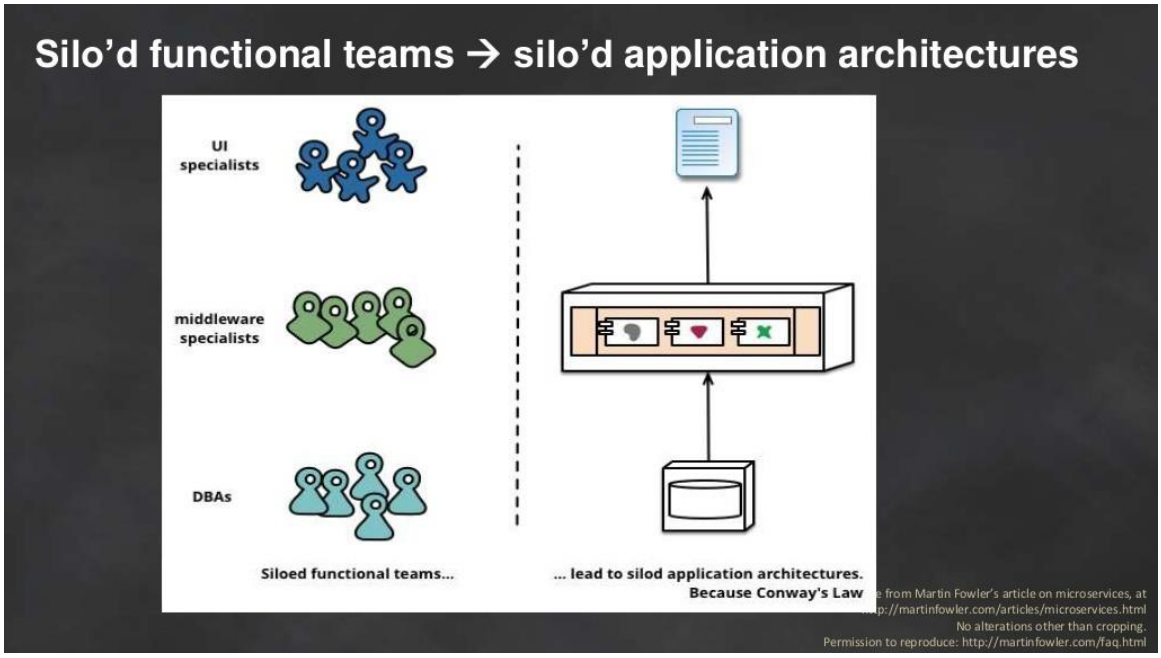
More than just technology transformation

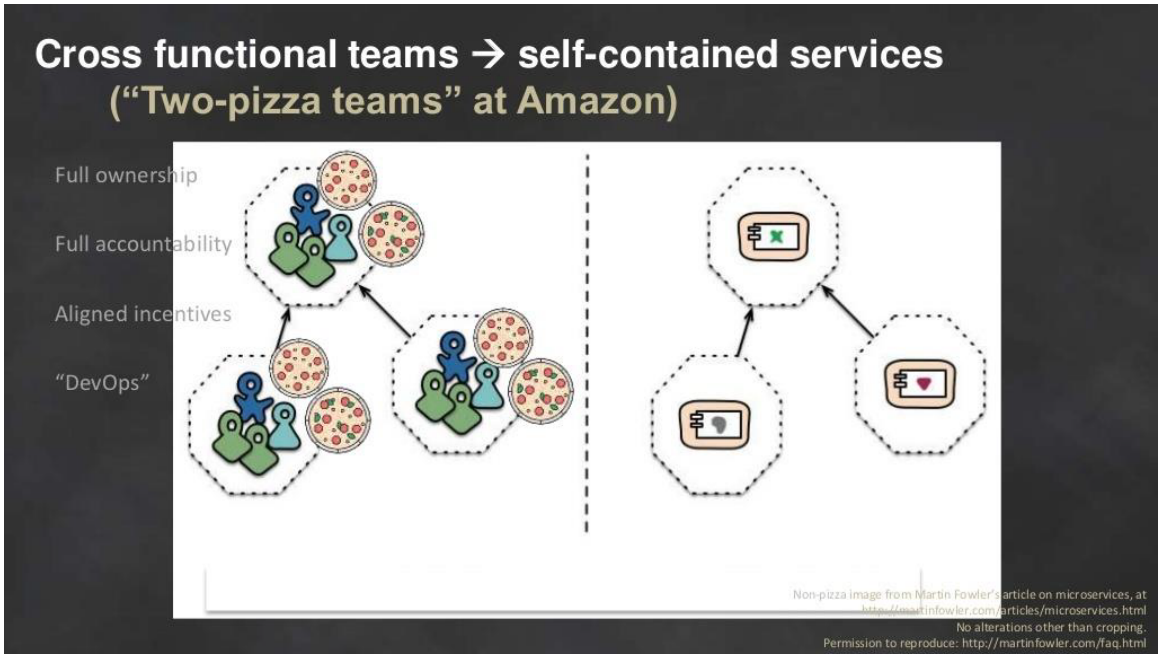
Conway's law

> any organization that designs a sytem will inevitably produce a design whose structure is a copy of the organiation's

> communication structure

Decentralize governance and data management

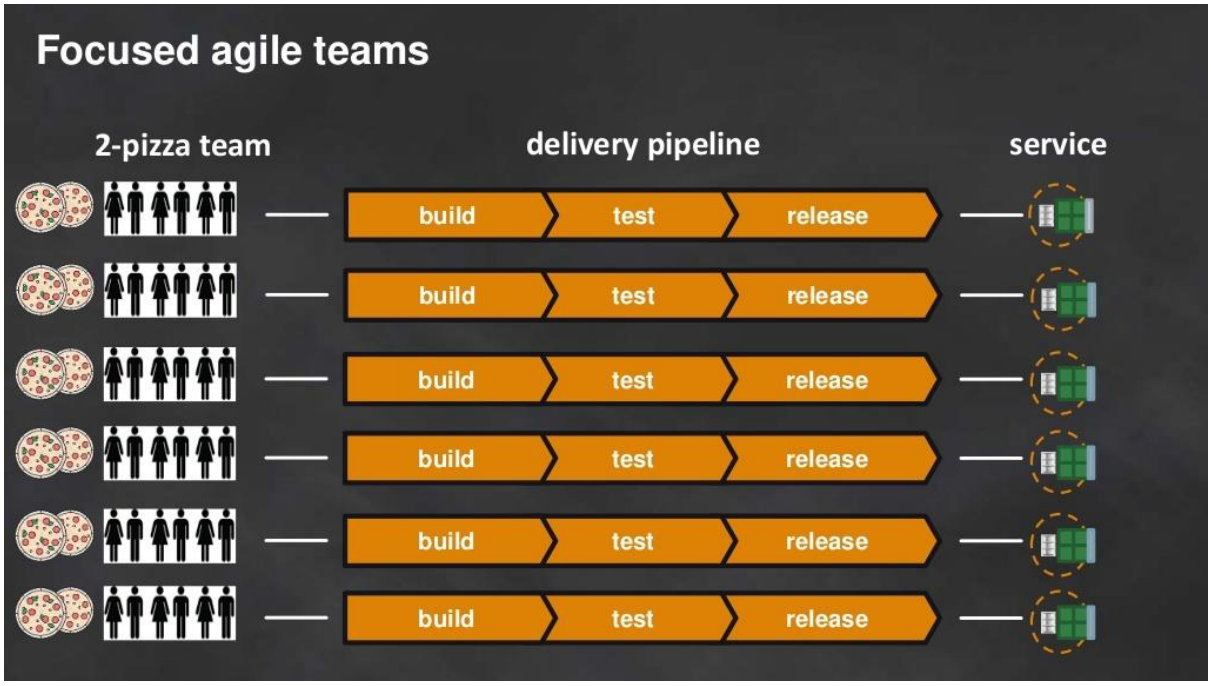




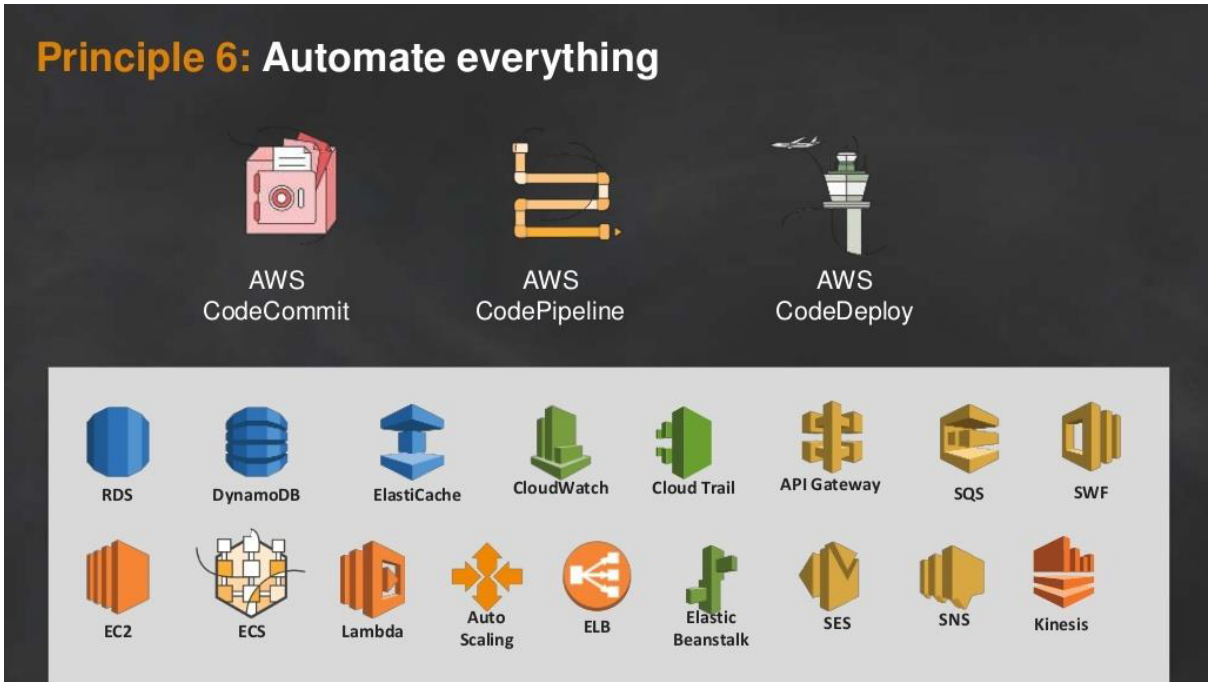
Have 2-pizza teams

11.9 Principle 6

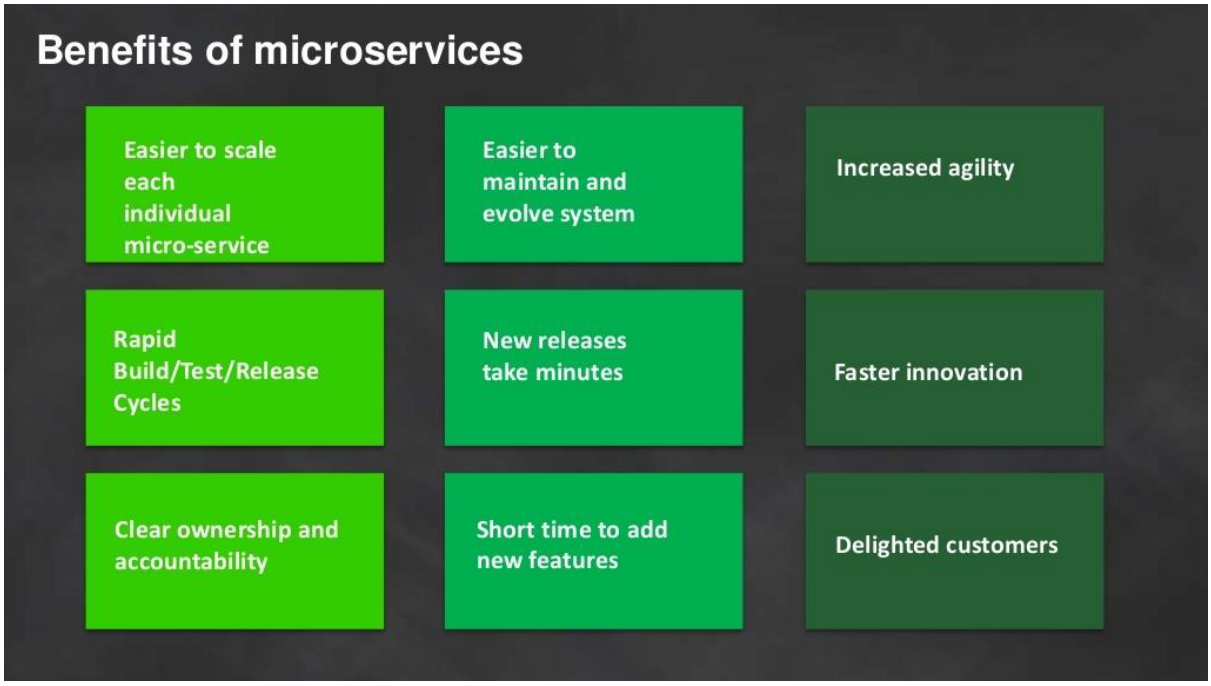
Automate everything



Focused agile teams



11.10 Benefits of microservices



See EXAMPLE in slides