

11. Microservices

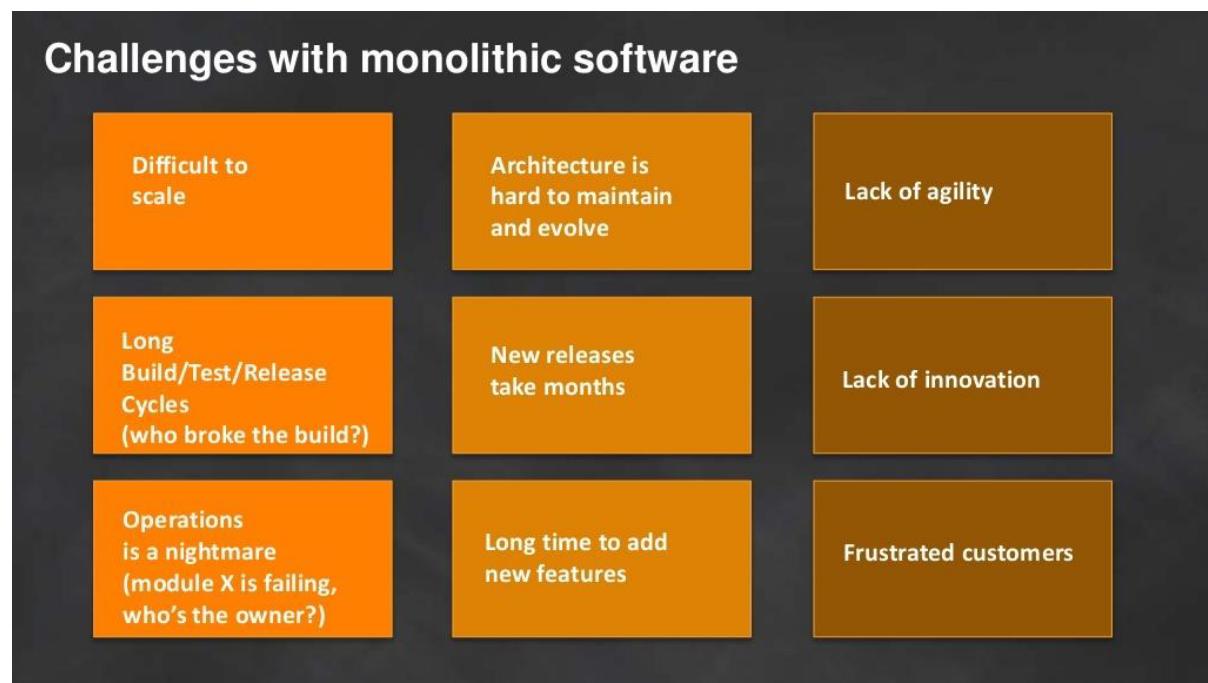
Current approach for service oriented architecture (SOA)

11.1 Monoliths to microservices

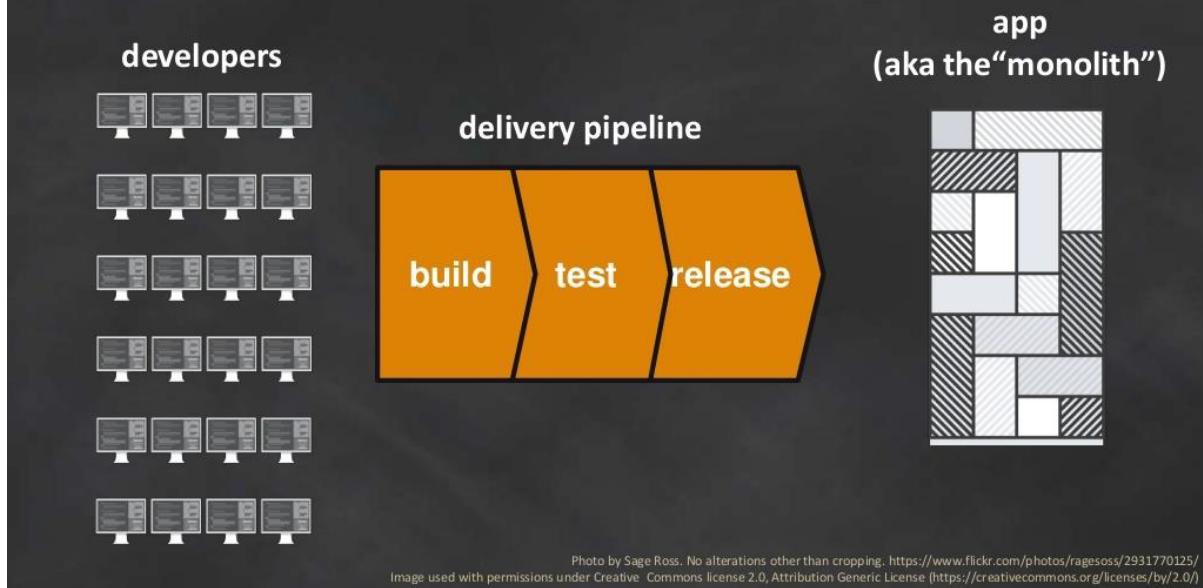
Components are packed together → monolith

Even if functionalities are divided, everything is in a block

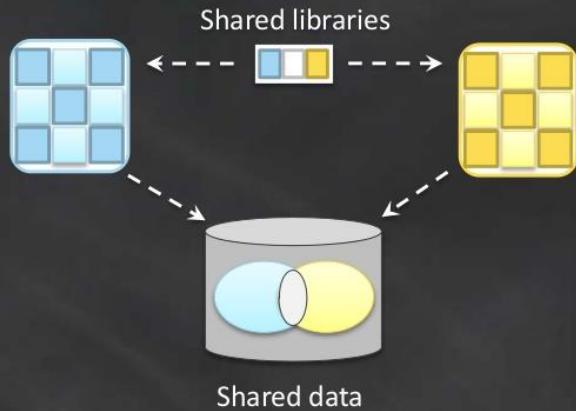
Disadvantages:



Monolith development lifecycle



Too much software coupling



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 independent on its own.

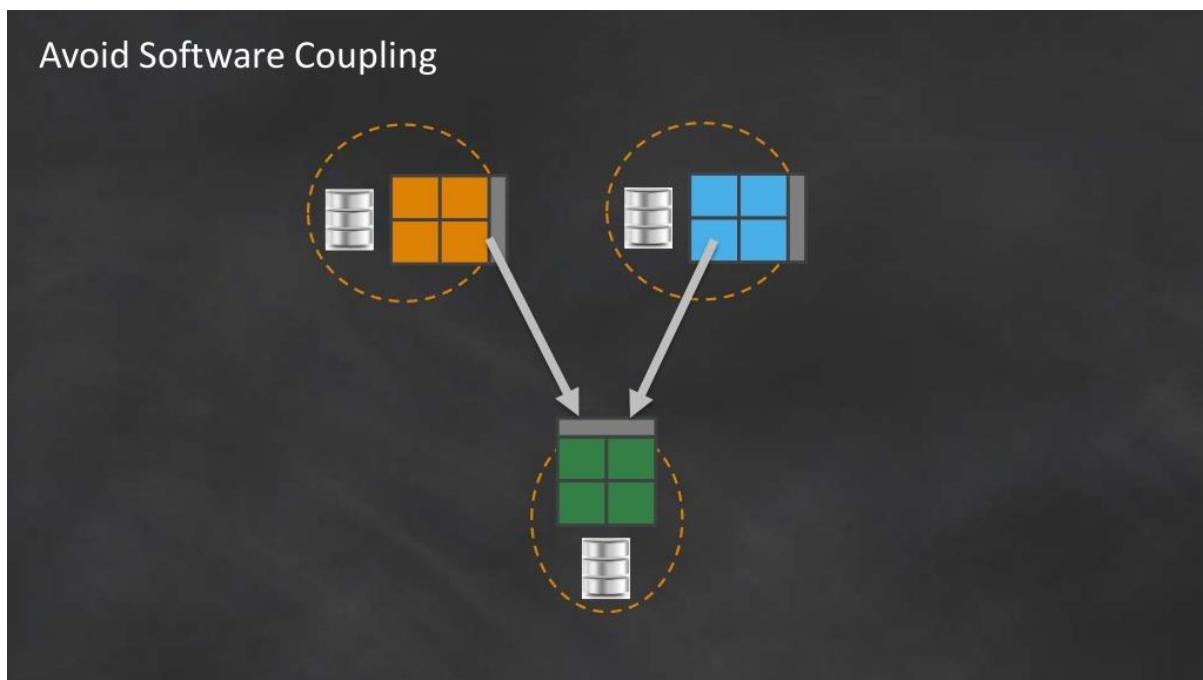
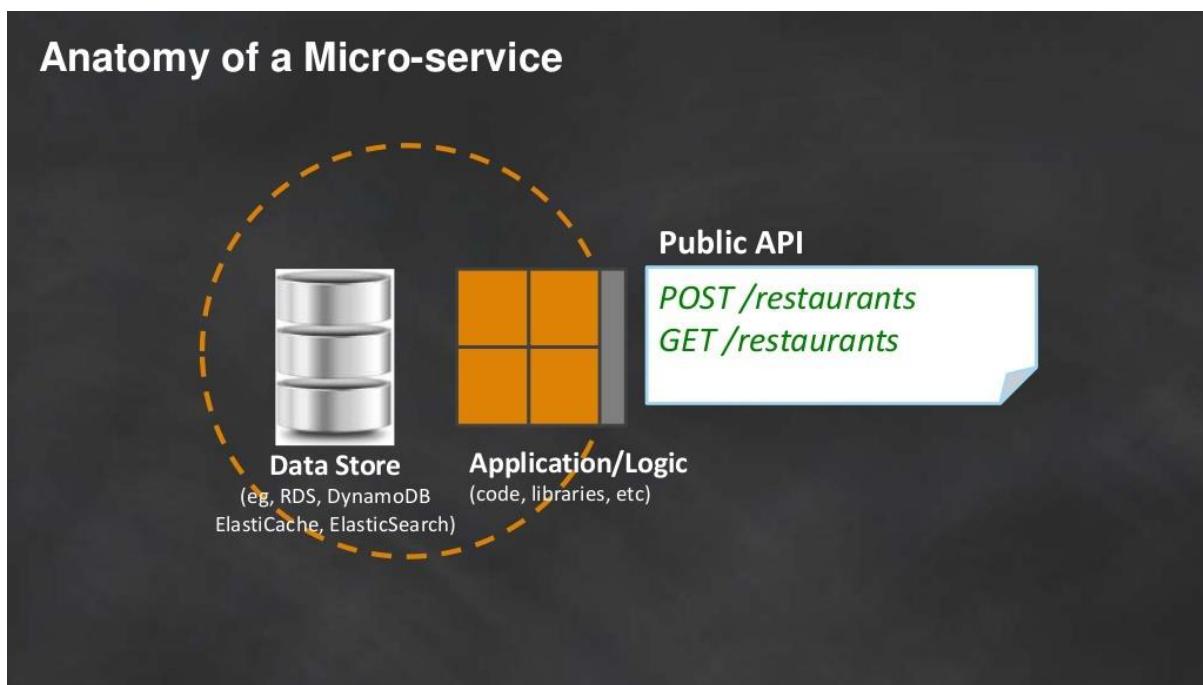
11.2 Microservices

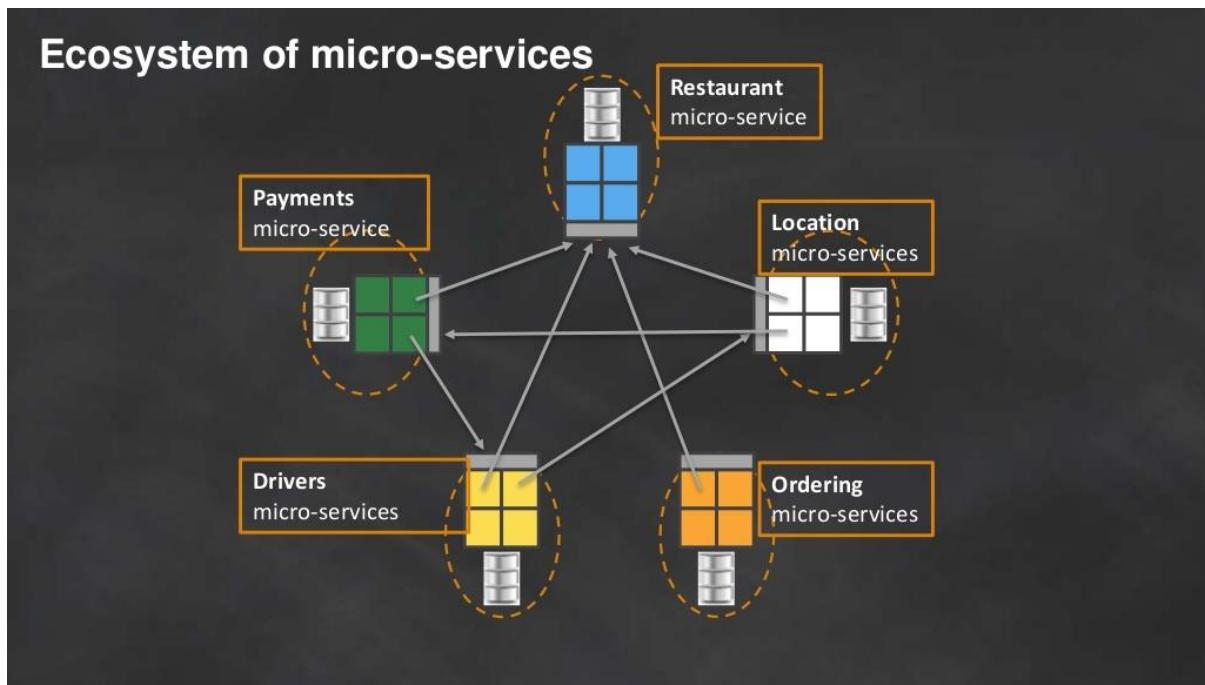
Microservices (def.): concrete and modern interpretation of SOA used to build distributed sw systems.

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).

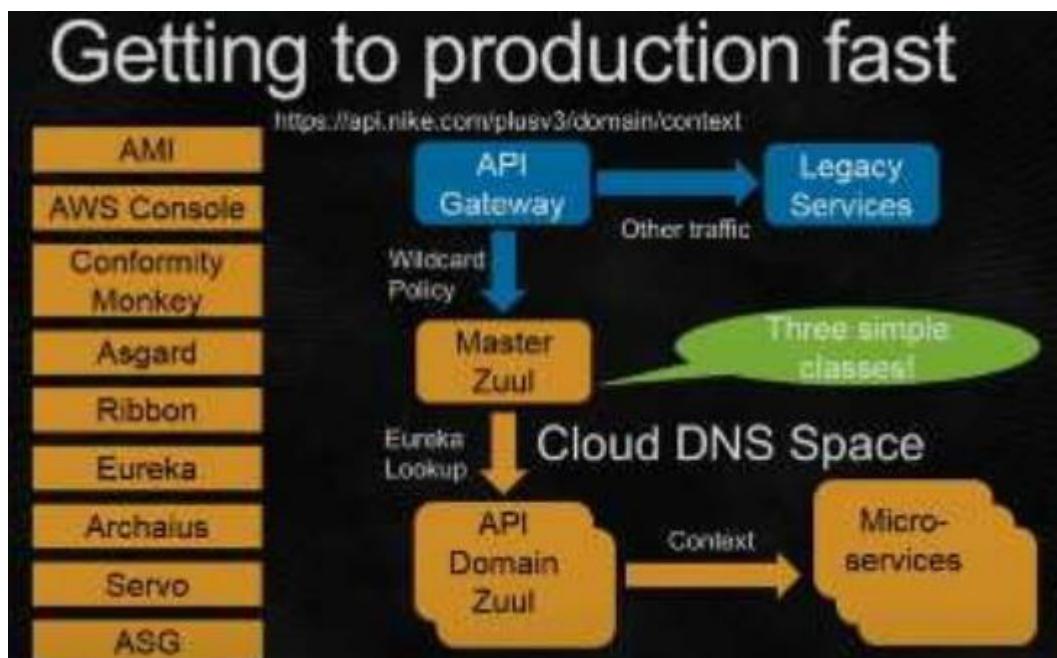
11.3 Anatomy of a micro-service

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





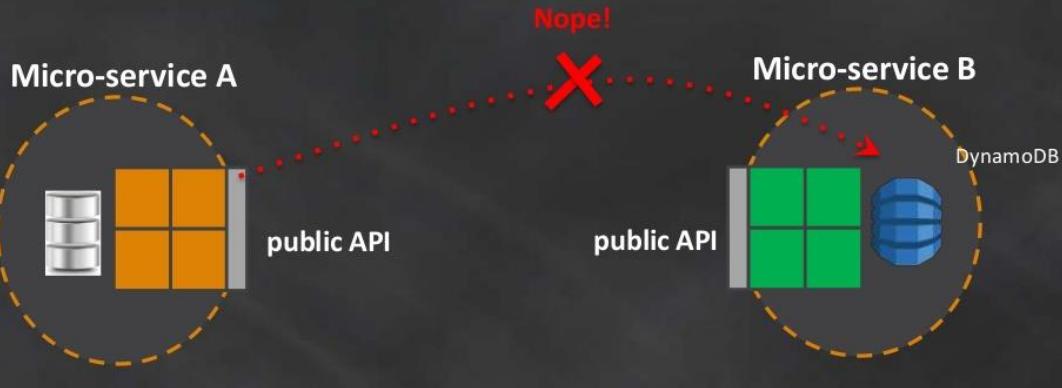
Example (Nike's journey to microservices)



11.4 Principle 1

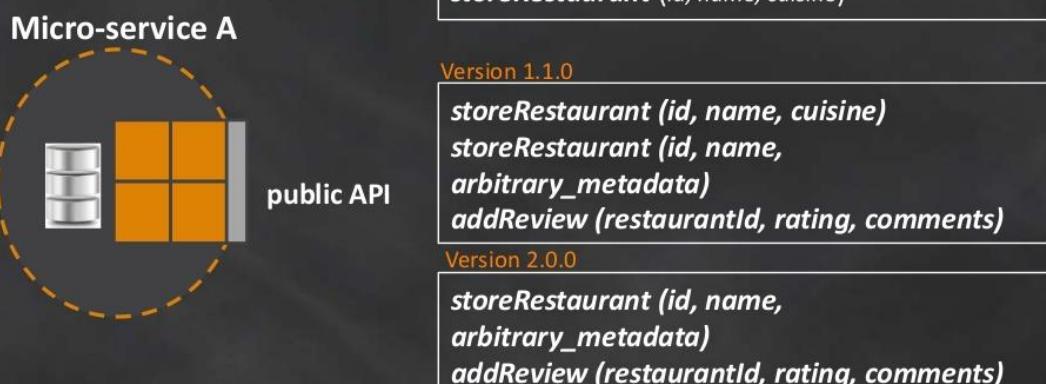
Micro-services only rely on each other's public API

Principle 1: Microservices only rely on each other's public API (Hide Your Data)



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

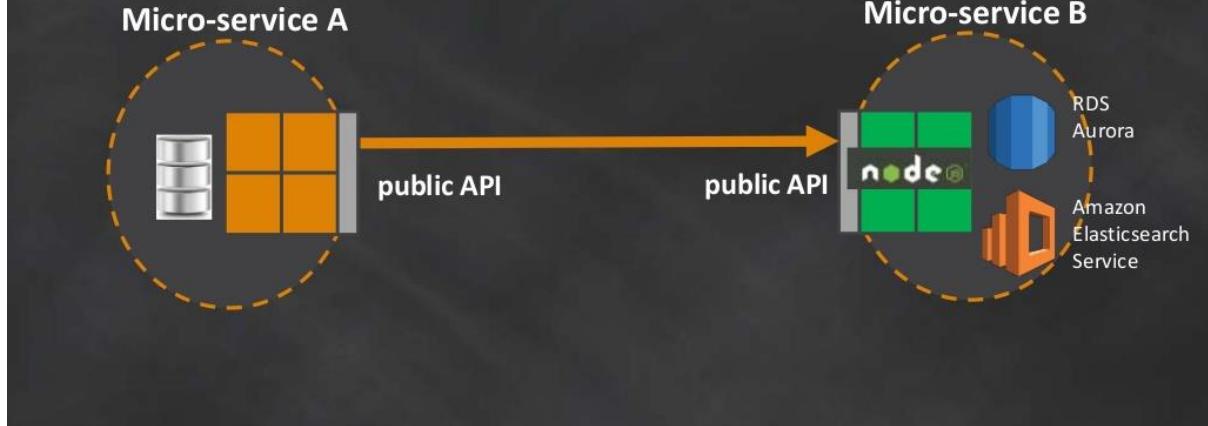
Principle 1: Microservices only rely on each other's public API (Evolve API in backward-compatible way...and document!)



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

11.5 Principle 2

Principle 2: Use the right tool for the job (Embrace polyglot programming frameworks)



Different microservices can use different technology, so:
every microservice can use its language, functionalities and paradigm
look for differences between sql and nosql.

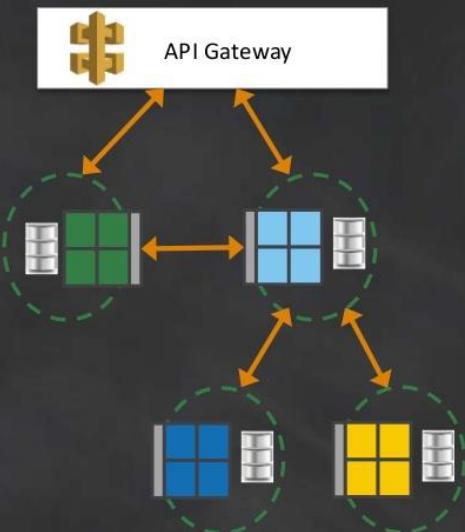
SEE EXAMPLE FROM P. 57

11.6 Principle 3

Secure your Services

when build complex, adopt defense-in-depth
gateway (filter request from the external)

Principle 3: Secure Your Services



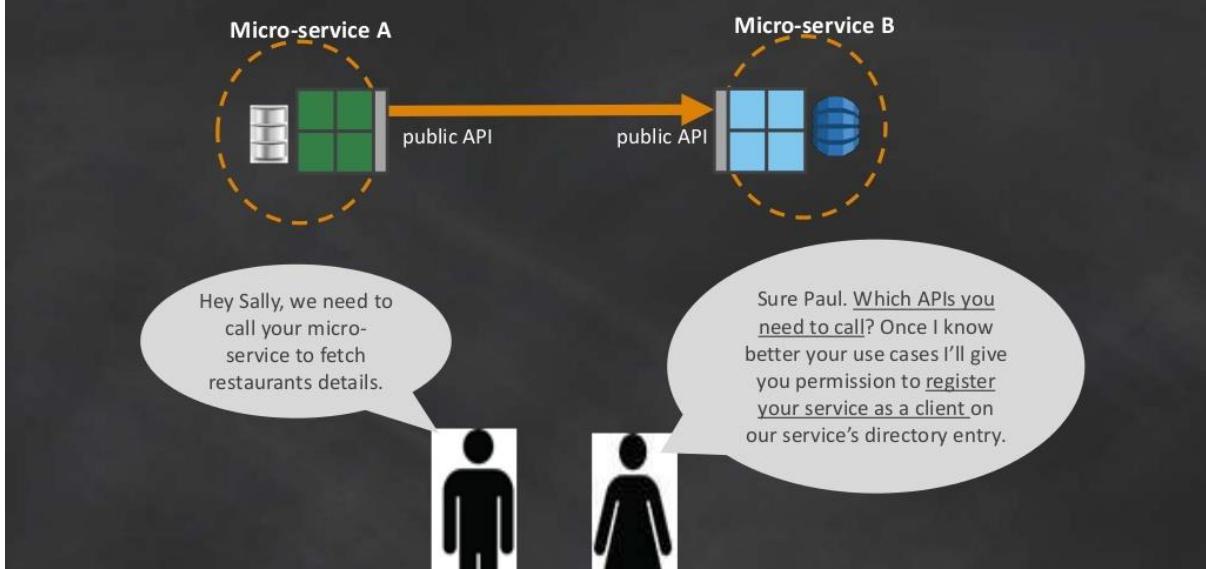
- **Defense-in-depth**
 - Network level (e.g. VPC, Security Groups, TLS)
 - Server/container-level
 - App-level
 - IAM policies
- **Gateway** ("Front door")
- **API Throttling**
- **Authentication & Authorization**
 - Client-to-service, as well as service-to-service
 - API Gateway: custom Lambda authorizers
 - IAM-based Authentication
 - Token-based auth (JWT tokens, OAuth 2.0)
- **Secrets management**
 - S3 bucket policies + KMS + IAM
 - Open-source tools (e.g. Vault, Keywhiz)

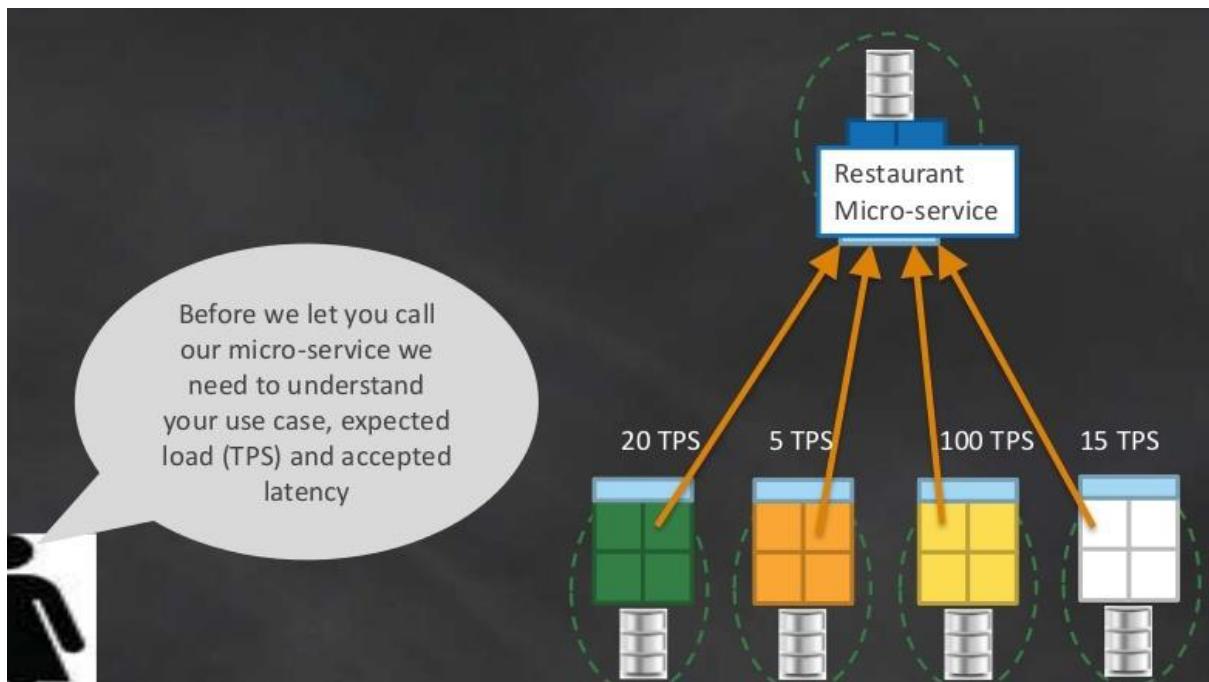
11.7 Principle 4

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

Principle 4: Be a good citizen within the ecosystem





Distributed monitoring and tracing

- “Is the service meeting its SLA?”
- “Which services were involved in a request?”
- “How did downstream dependencies perform?”

Shared metrics

- e.g. request time, time to first byte

Distributed tracing

- e.g. Zipkin, OpenTracing

User-experience metrics

11.8 Principle 5

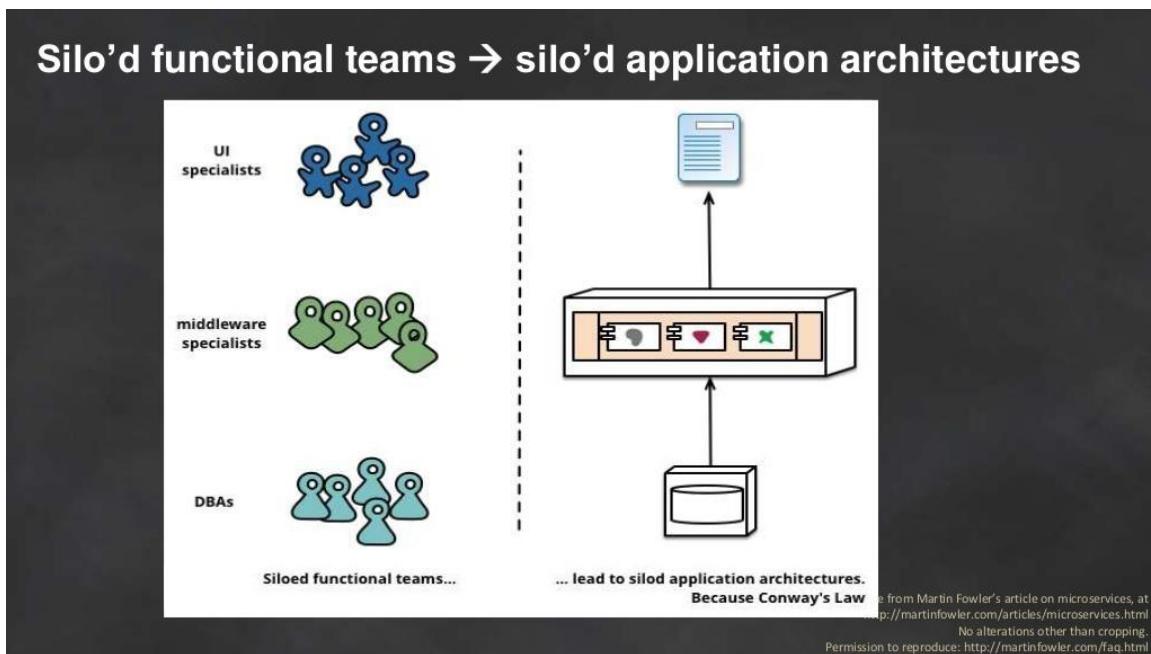
More than just technology transformation

Conway's law

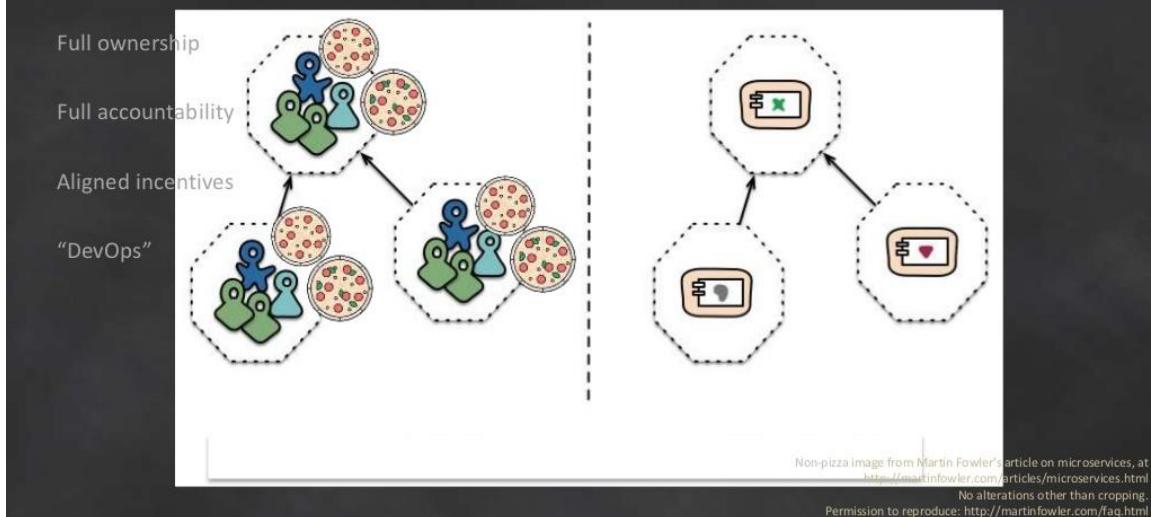
any organization that designs a system will inevitably produce a design whose structure is a copy of the organisation's

communication structure

Decentralize governance and data management



Cross functional teams → self-contained services ("Two-pizza teams" at Amazon)

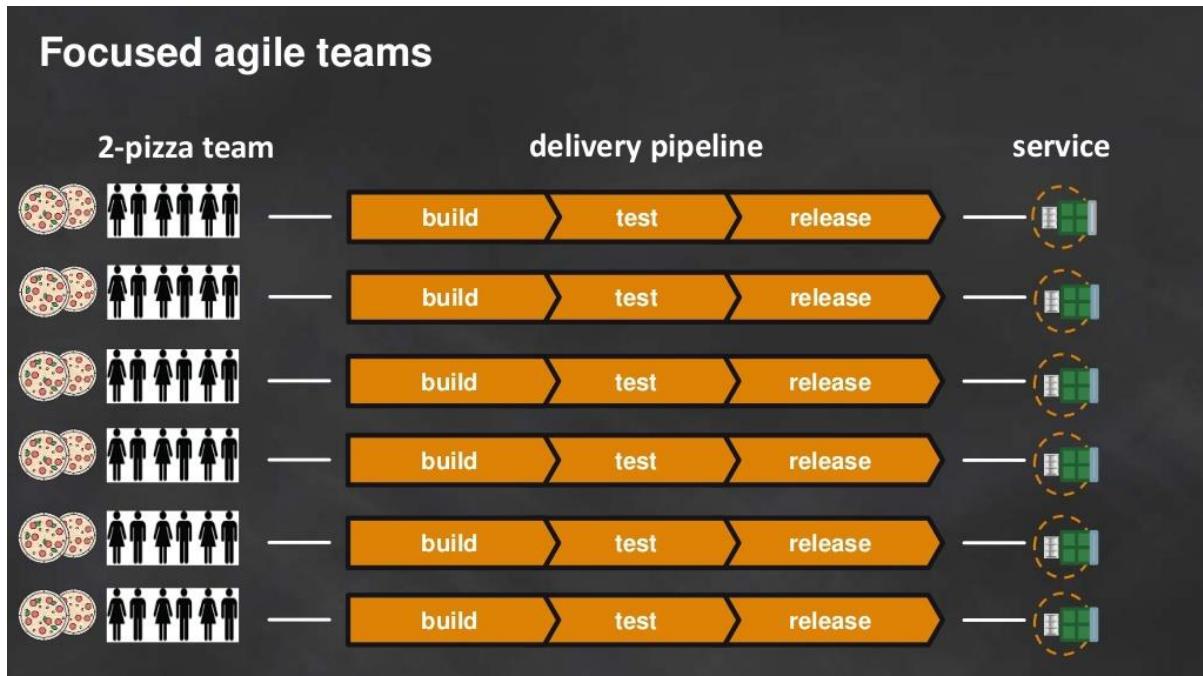


Have 2-pizza teams

11.9 Principle 6

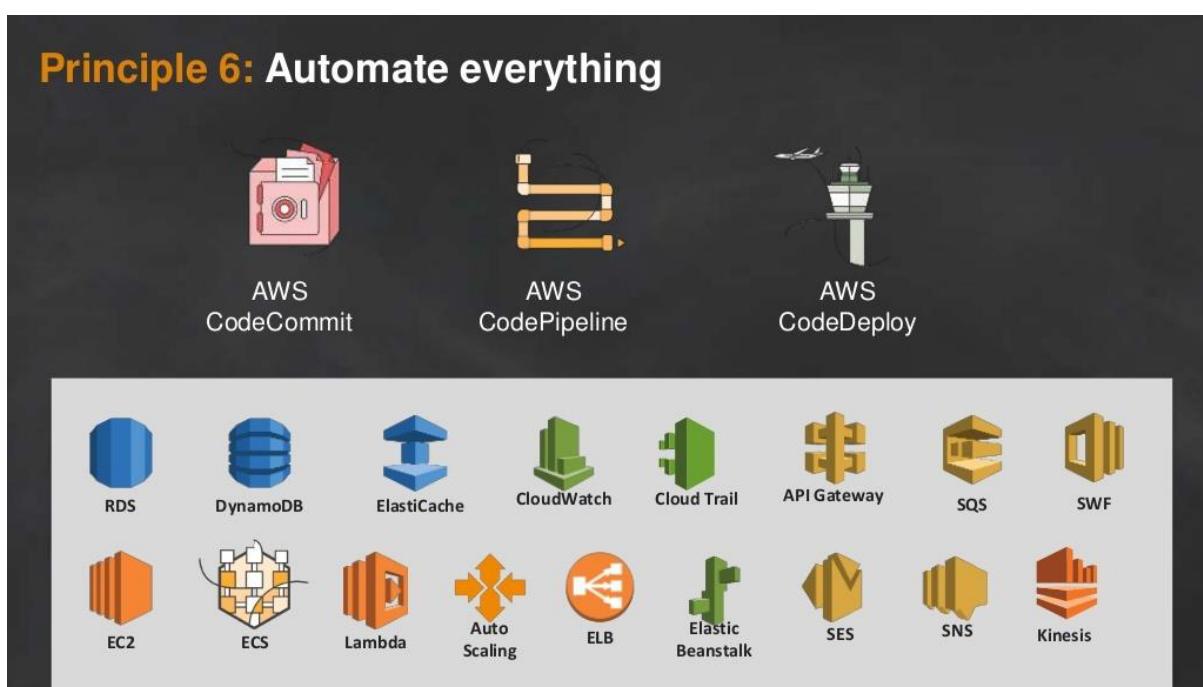
Automate everything

Focused agile teams



Focused agile teams

Principle 6: Automate everything



11.10 Benefits of microservices

Benefits of microservices

Easier to scale each individual micro-service

Easier to maintain and evolve system

Increased agility

Rapid Build/Test/Release Cycles

New releases take minutes

Faster innovation

Clear ownership and accountability

Short time to add new features

Delighted customers

See EXAMPLE in slides