Carnegie Mellon University Qatar

Microservices

17-313 Fall 2023

Inspirations:
Martin Fowler (http://martinfowler.com/articles/microservices.html)
Josh Evans @ Netflix (https://www.youtube.com/watch?v=CZ3wIuvmHeM)
Matt Ranney @ Uber (https://www.youtube.com/watch?v=kb-m2fasdDY)
Christopher Meiklejohn & Filibuster (http://filibuster.cloud)
Sam Newman (https://samnewman.io/books/building_microservices/)

S3D
Software and Societal Systems Department

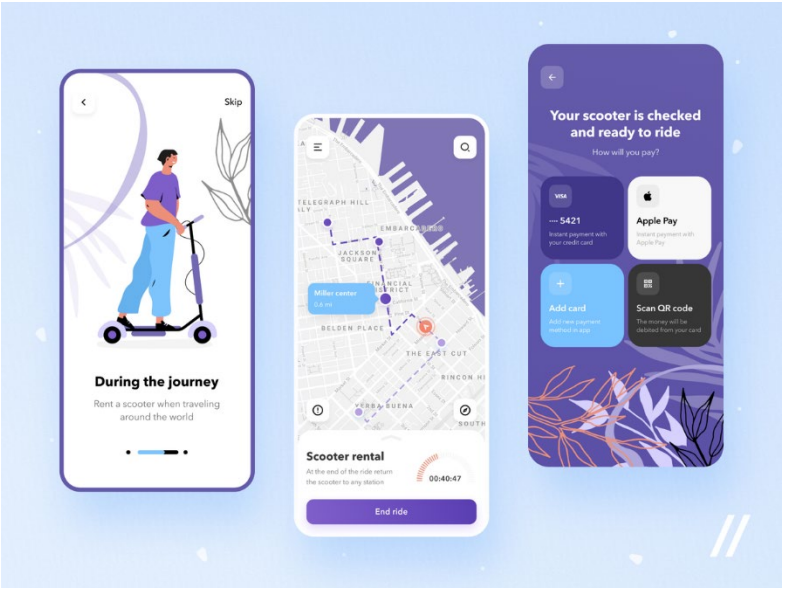
Learning Goals

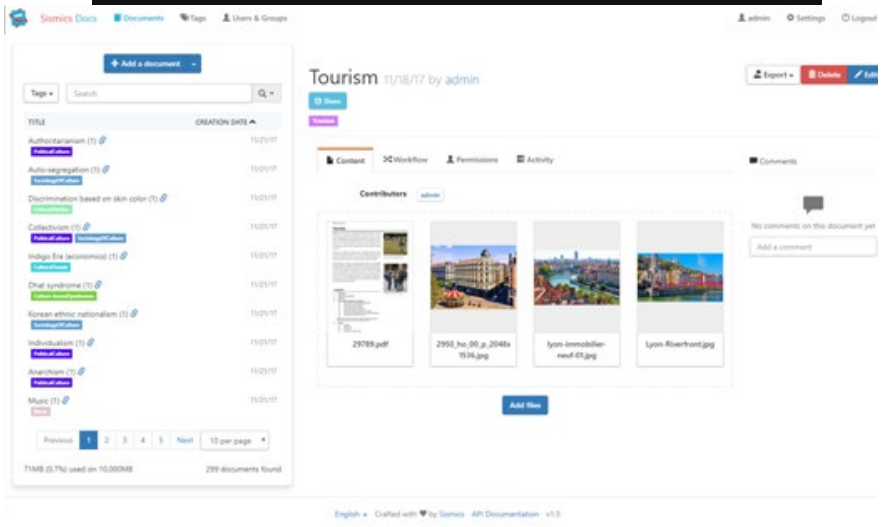
- Contrast the monolithic application design with a modular design based on microservices.
- Principles of Microservices
- Reason about tradeoffs of Microservices architectures.

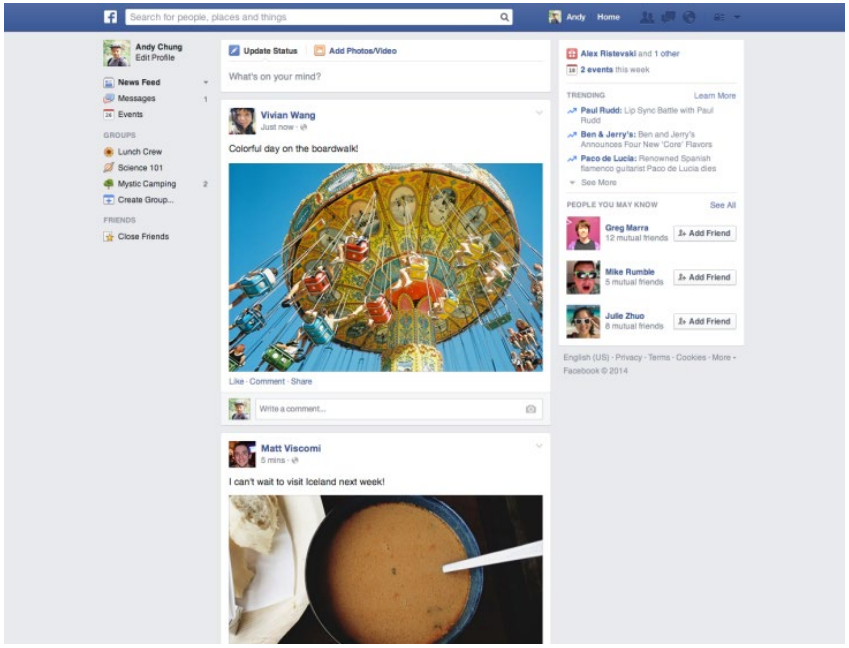
Before we get to microservices…

S3D

How might these apps be architected?

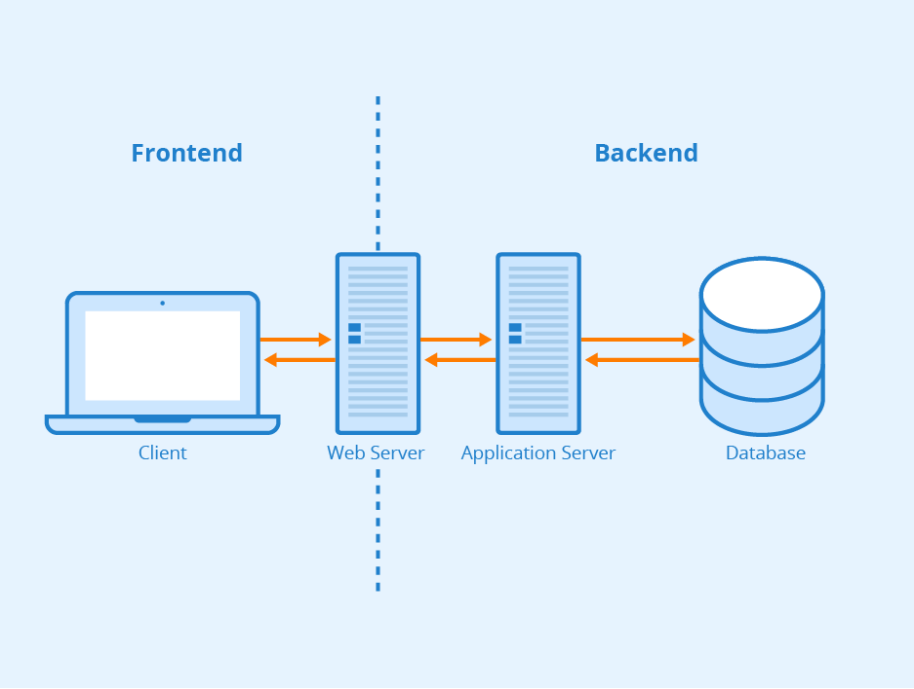




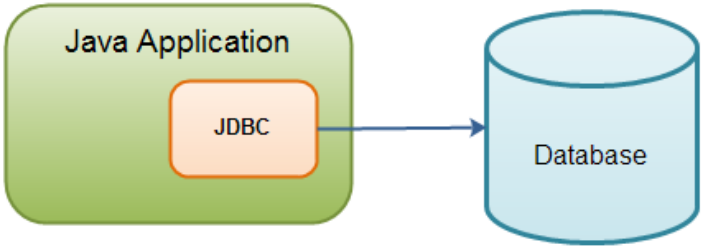


MONOLITHS

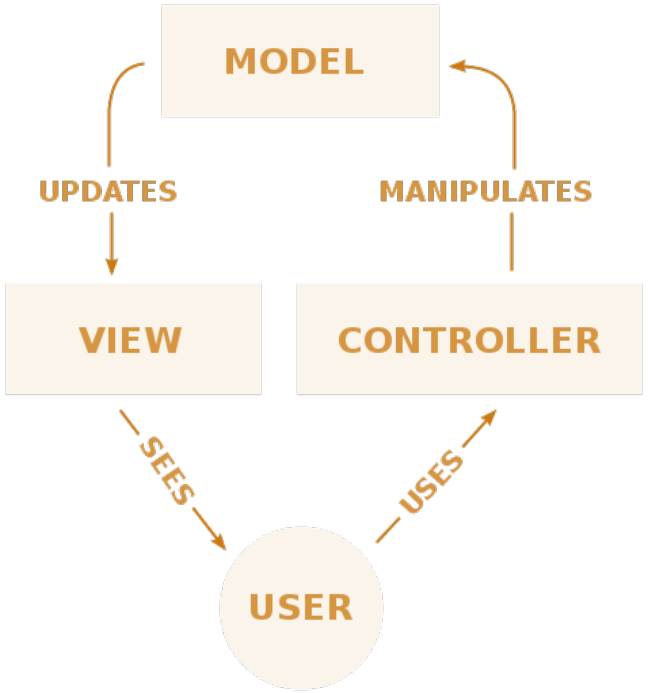
Monolithic styles



Source: https://www.seobility.net (CC BY-SA 4.0)



Monolithic styles: MVC Pattern (e.g., NodeBB)

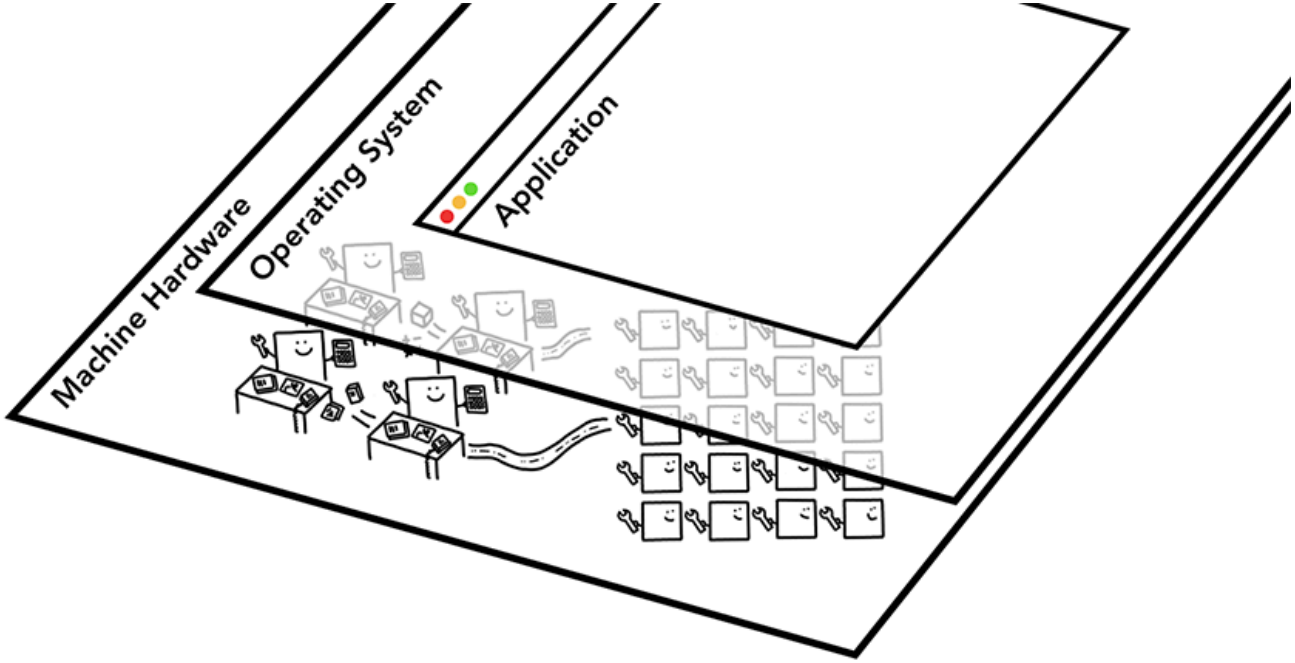


Separation of concerns

SERVICE-BASED ARCHITECTURE

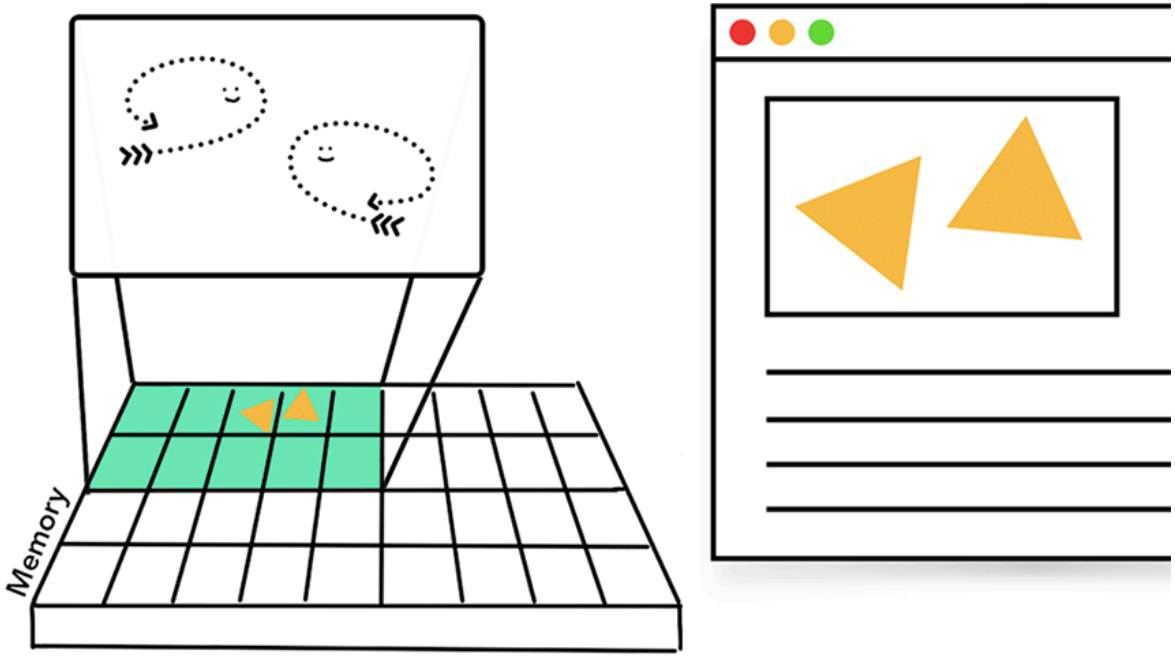
Chrome

Web Browsers



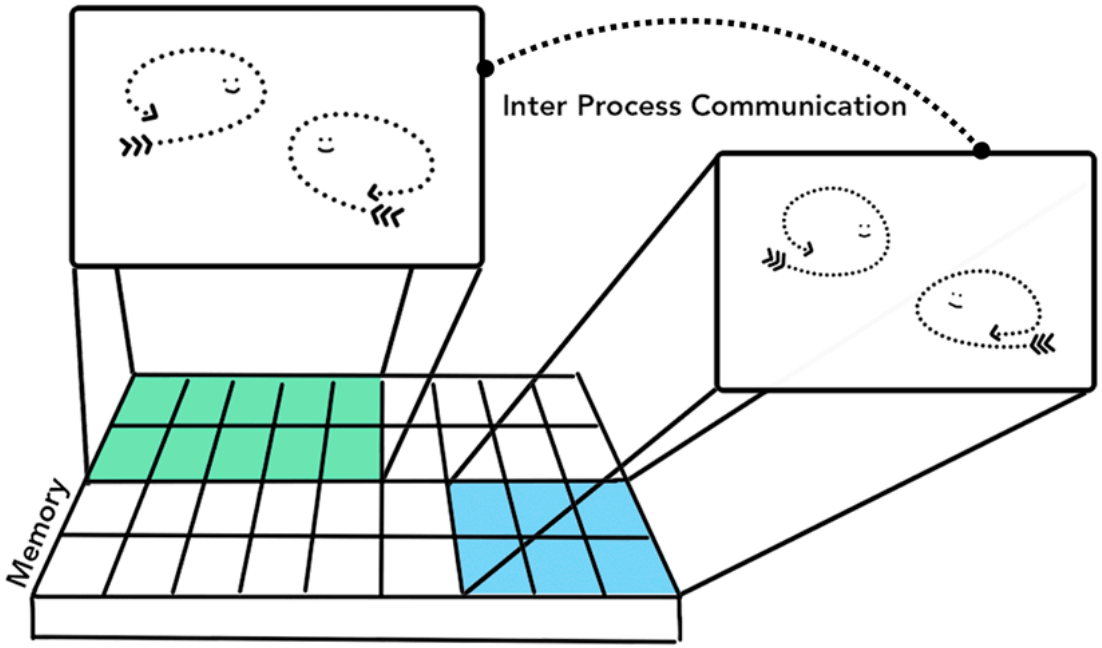
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Browser: A multi-threaded process



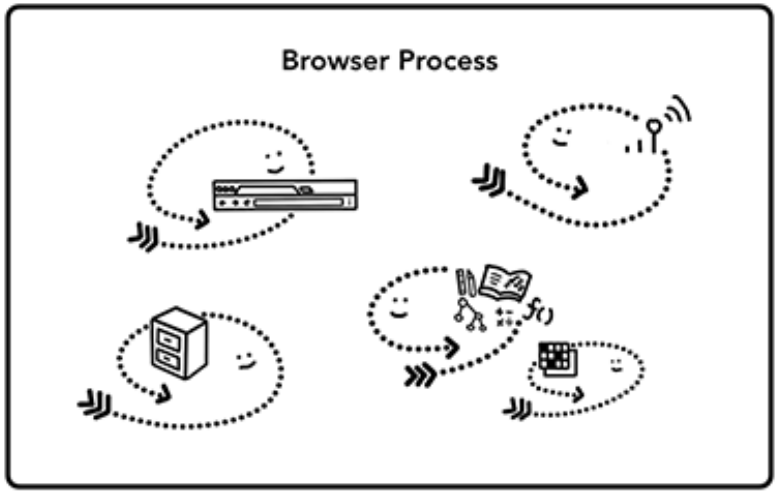
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

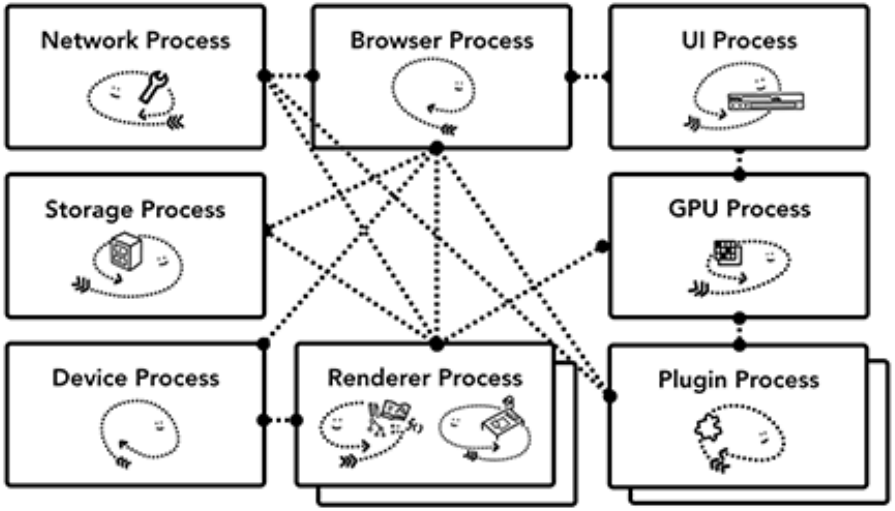
Multi-process browser with IPC



Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

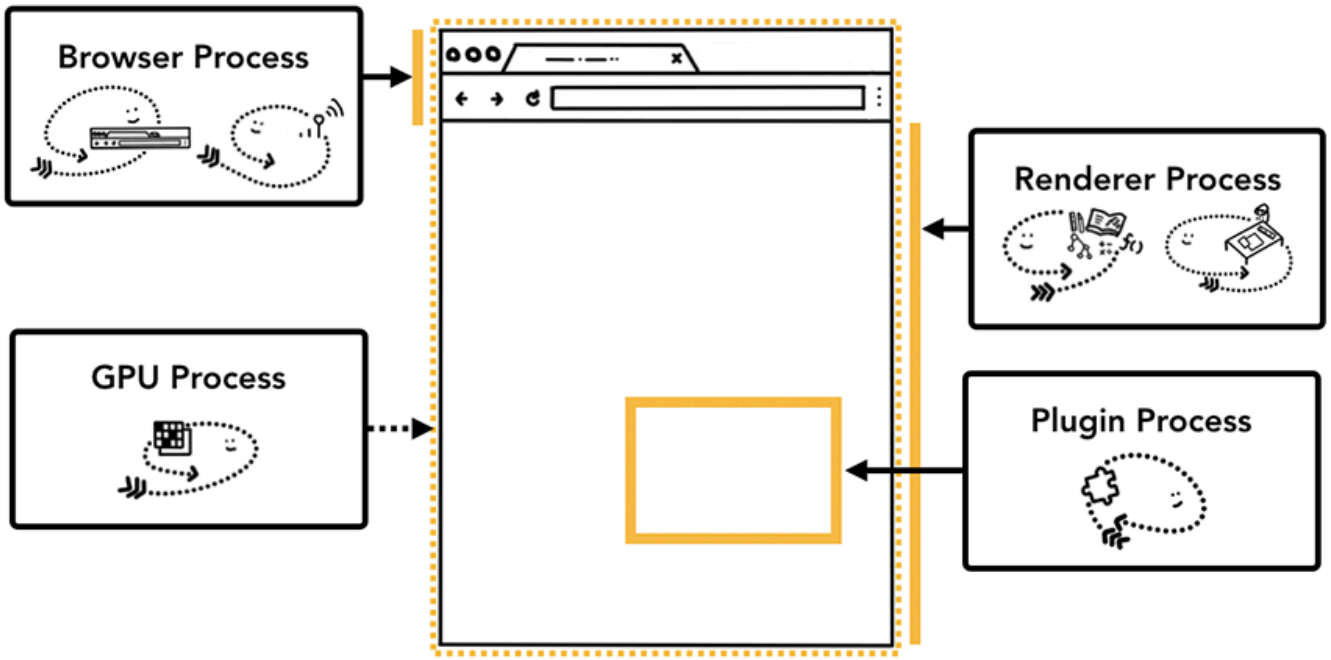
Browser Architectures





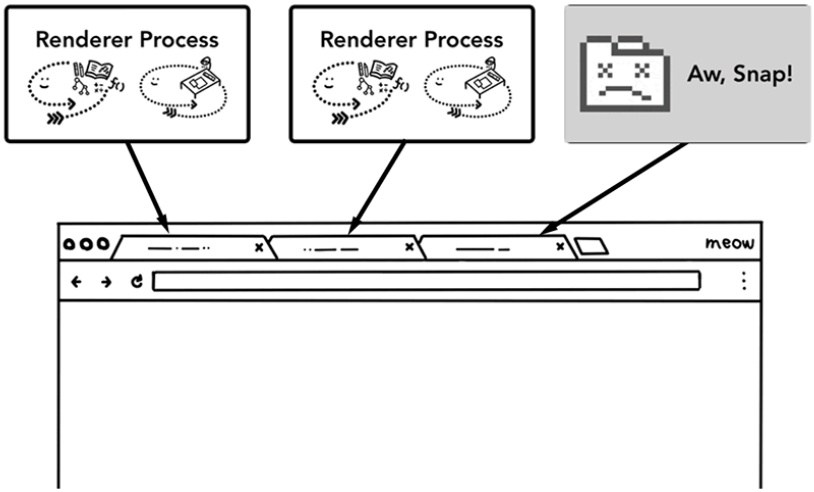
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

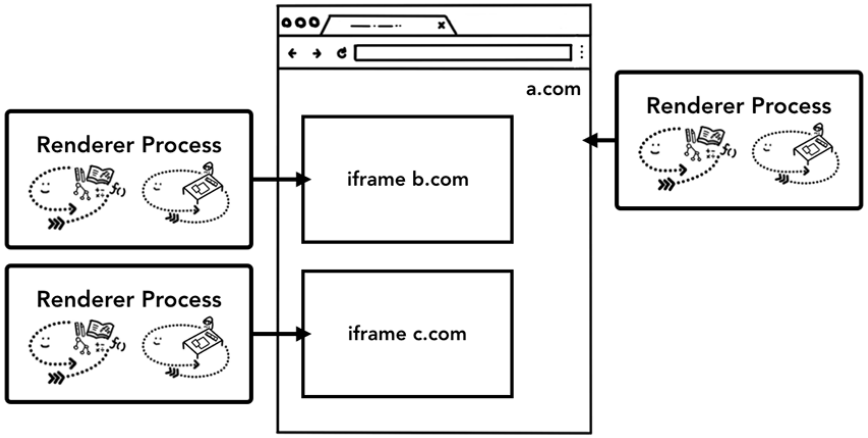
Service-based browser architecture



Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Service-based browser architecture





Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Service Oriented Architecture

- Ability to change components independently
- Independent processes
- Focusing on doing one thing well

MICROSERVICES


Microservices
Microservices Everywhere


WHAT ARE
MICROSERVICES?
SERVICES FOR ANTS???!
makeameme.org

"*Small autonomous services that work well together*"

Sam Newman

S3D

Microservices

COMCAST

NETFLIX

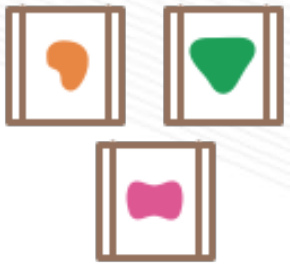
ebay

UBER

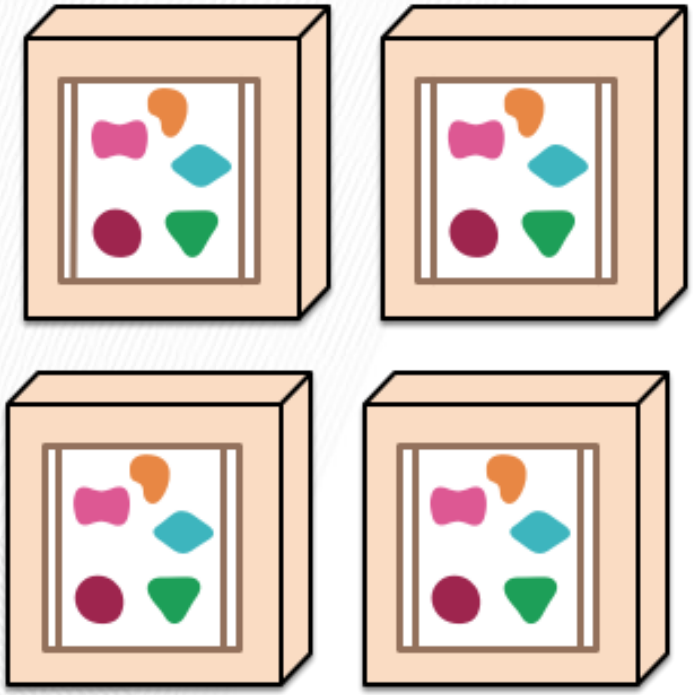
GROUPON®

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

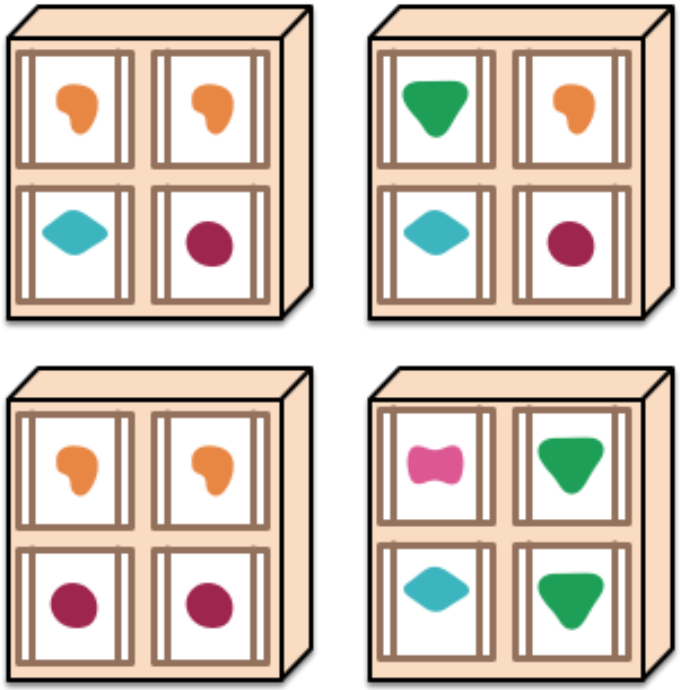
A microservices architecture puts each element of functionality into a separate service...



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



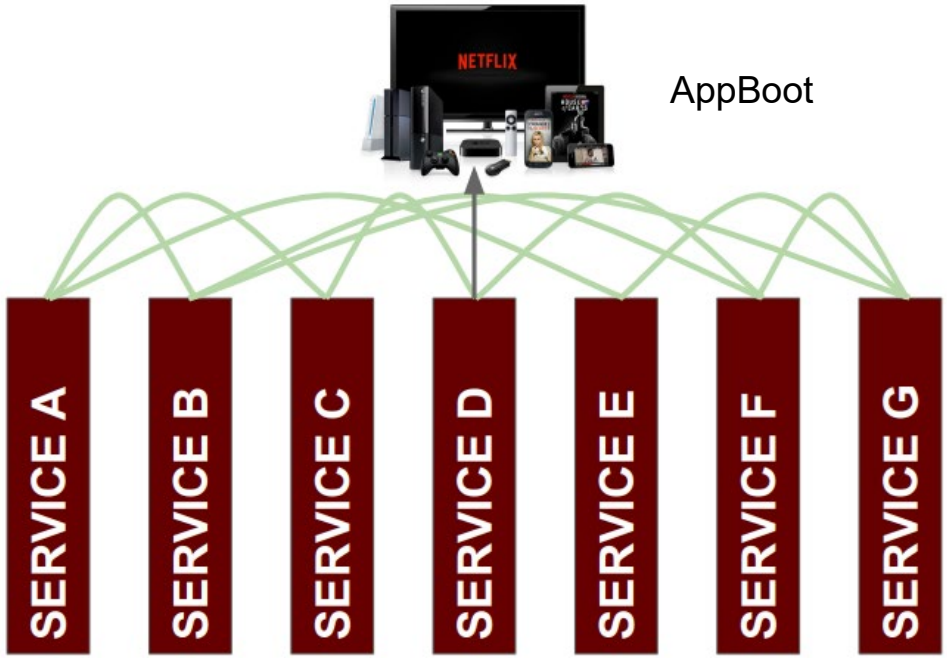
... and scales by distributing these services across servers, replicating as needed.



source: http://martinfowler.com/articles/microservices.html

S3D

Netflix Discussion



AppBoot

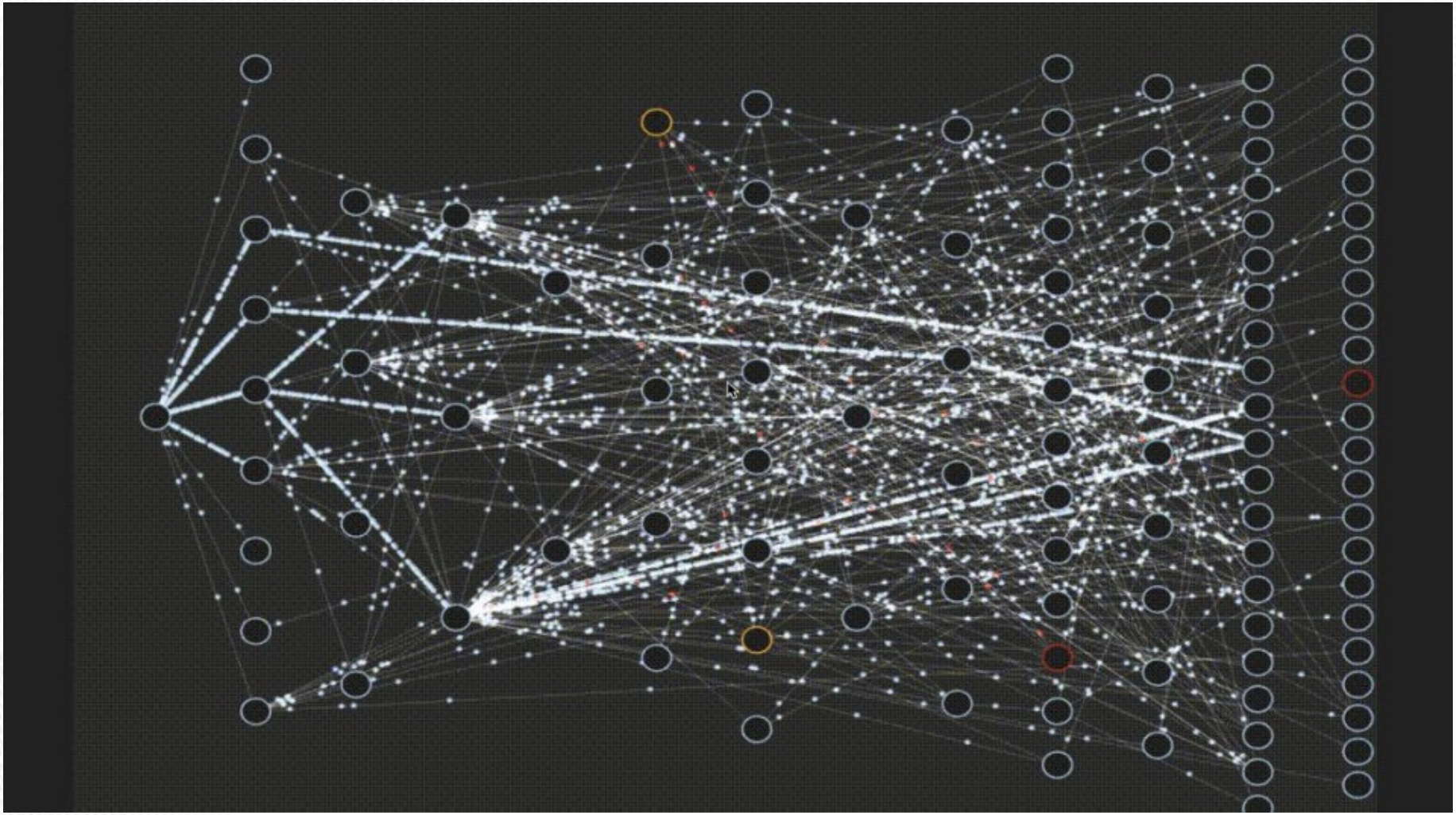
Bookmarks

Recommendations

My List

Metrics

(as of 2016)



Monoliths vs Microservices

What are the consequences of this architecture? On:

- Scalability
- Reliability
- Performance
- Development
- Maintainability
- Evolution
- Testability
- Ownership
- Data Consistency

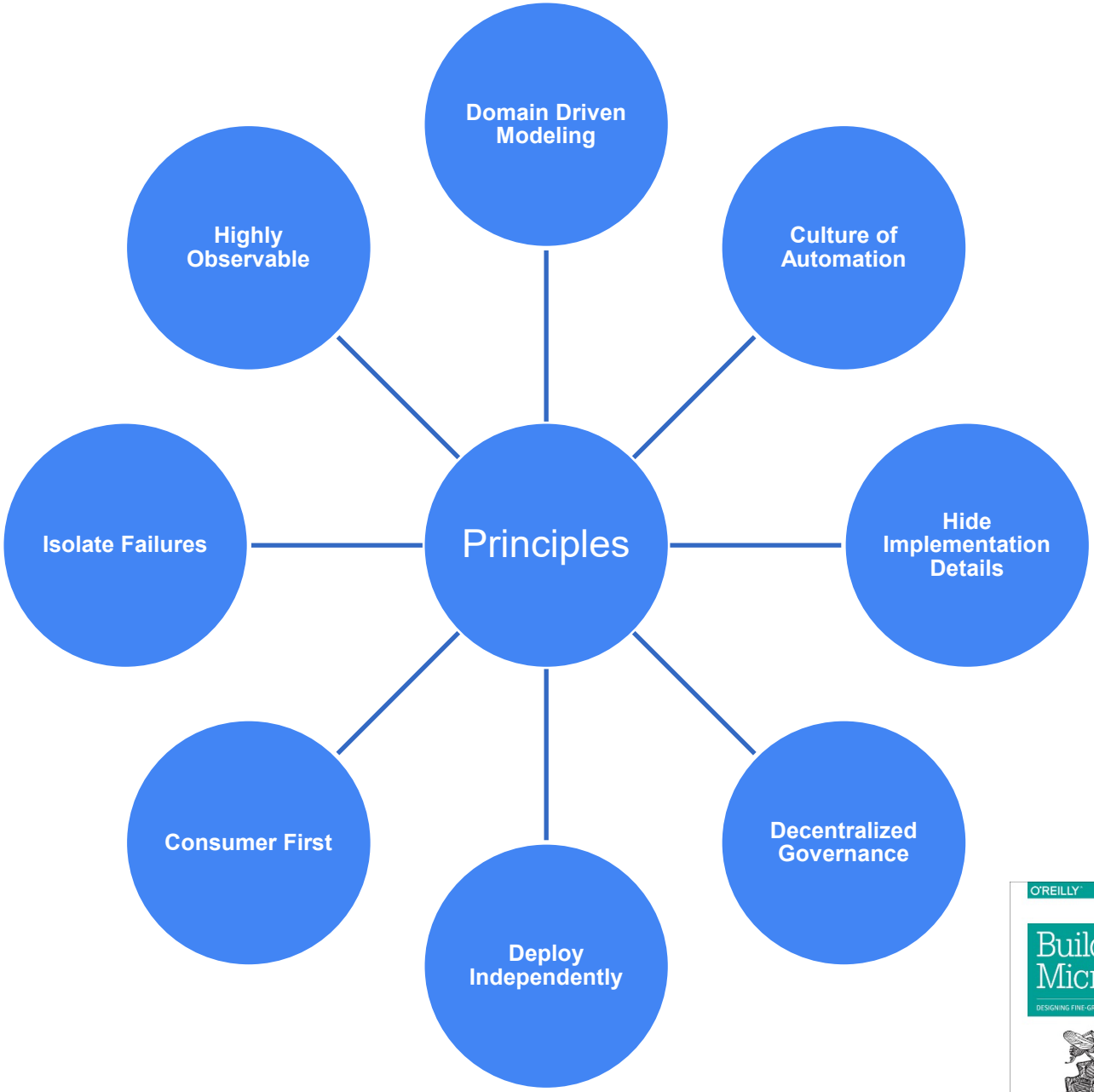
Advantages of Microservices

- Better alignment with the organization
- Ship features faster and safer
- Scalability
- Target security concerns
- Allow the interplay of different systems and languages, no commitment to a single technology stack
- Easily deployable and replicable
- Embrace uncertainty, automation, and faults

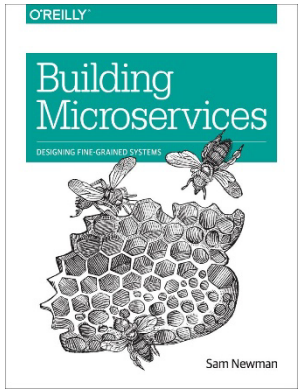
S3D

Microservice challenges

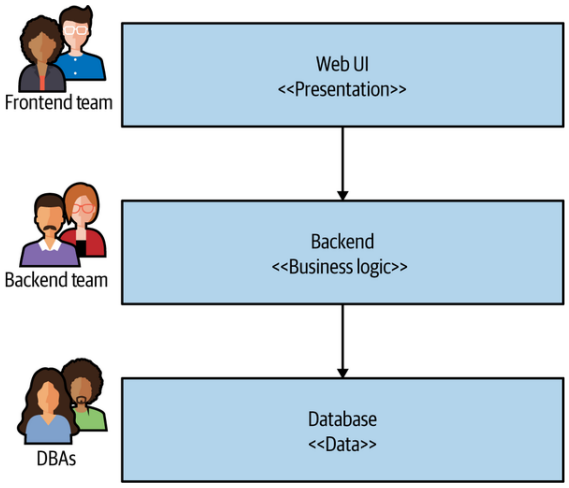
- Too many choices
- Delay between investment and payback
- Complexities of distributed systems
  - network latency, faults, inconsistencies
  - testing challenges
- Monitoring is more complex
- More system states
- Operational complexity
- Frequently adopted by breaking down a monolithic application

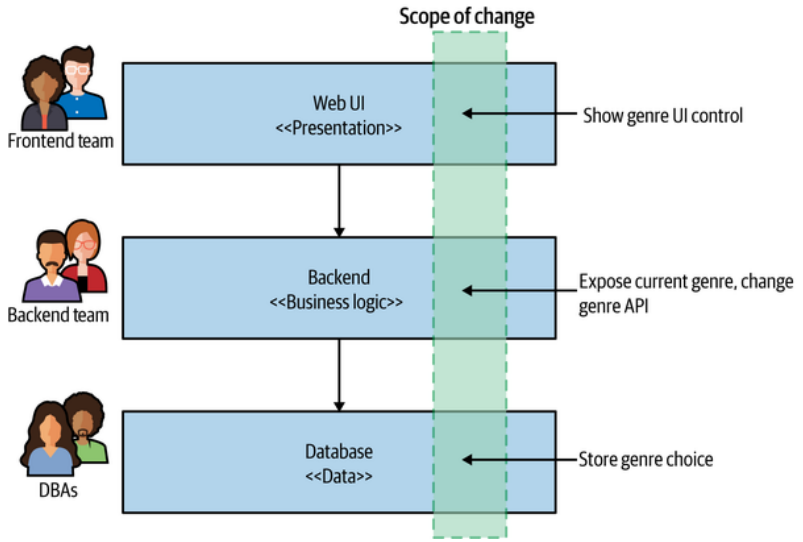


Sam Newman's Principles of Microservices

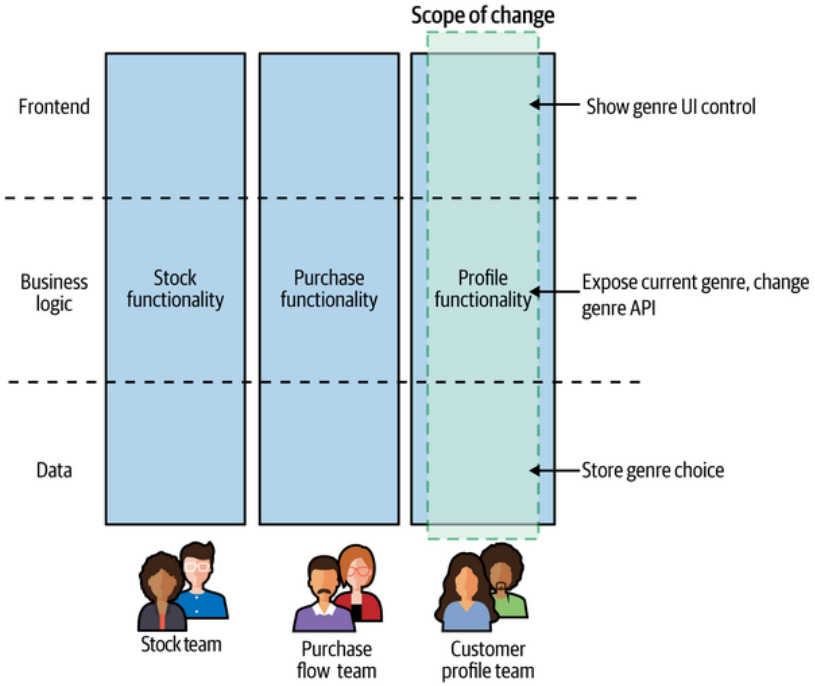


Principle 1:
Domain-driven modeling





Principle 1:
Domain-driven modeling



Principle 2:
Culture of Automation

- API-Driven Machine Provisioning

Example: Infrastructure as code (IaC)

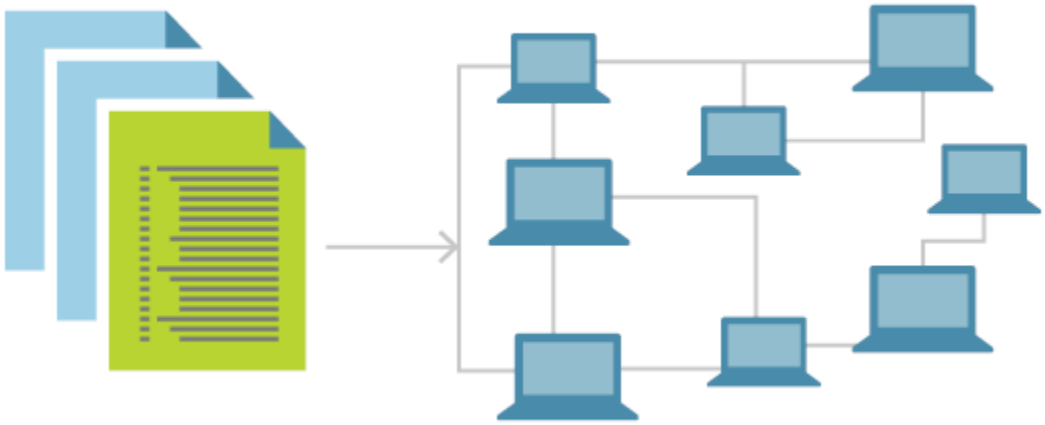
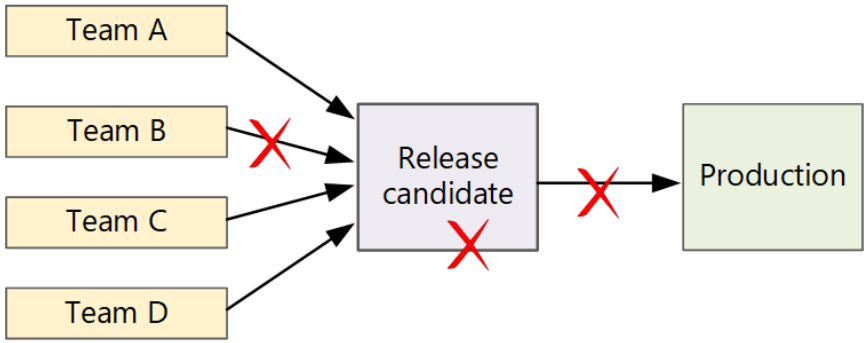


Image source: https://learn.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code

Principle 2:
Culture of Automation

- API-Driven Machine Provisioning
- Continuous Delivery



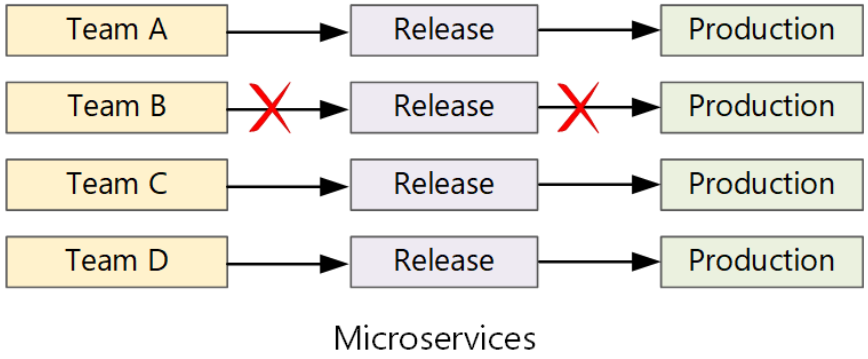
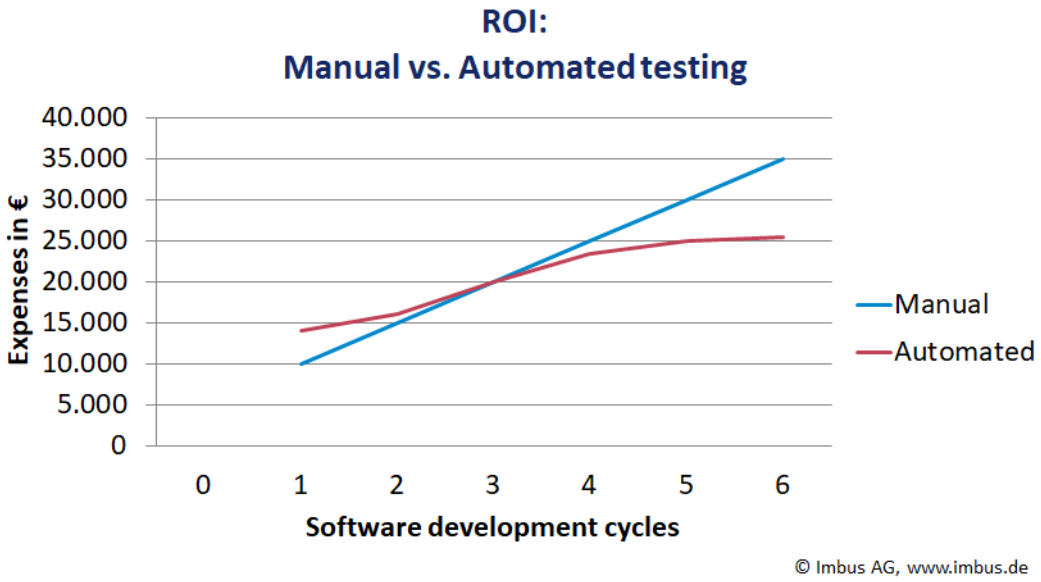


Image Source: https://learn.microsoft.com/en-us/azure/architecture/microservices/ci-cd

Principle 2:
Culture of Automation

- API-Driven Machine Provisioning
- Continuous Delivery
- Automated Testing

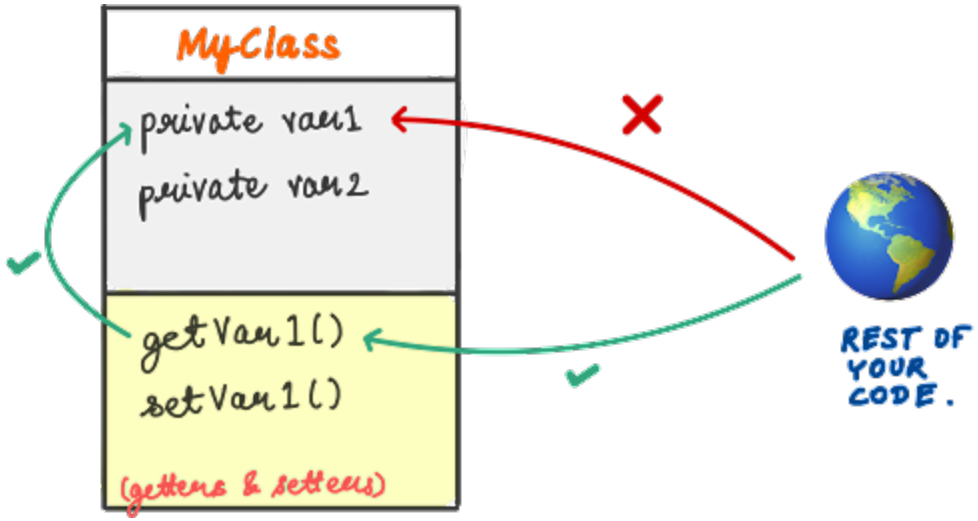
Automated Testing: ROI



This is for classical monolithic software projects; what about for Microservices?

Principle 3:
Hide implementation details

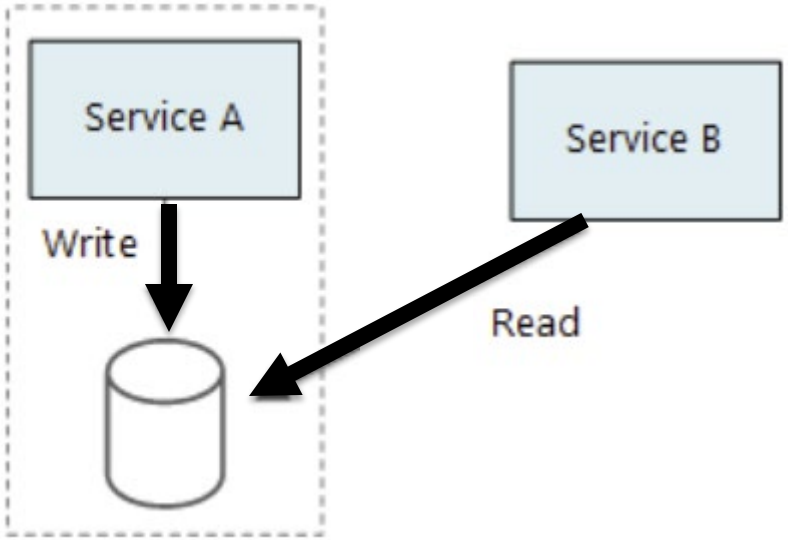
Recall: Encapsulation in OOP



Principle 3:
Hide implementation details

- Design carefully your APIs
- It’s easier to expose some details later than hide them

Sharing database: Anti-pattern

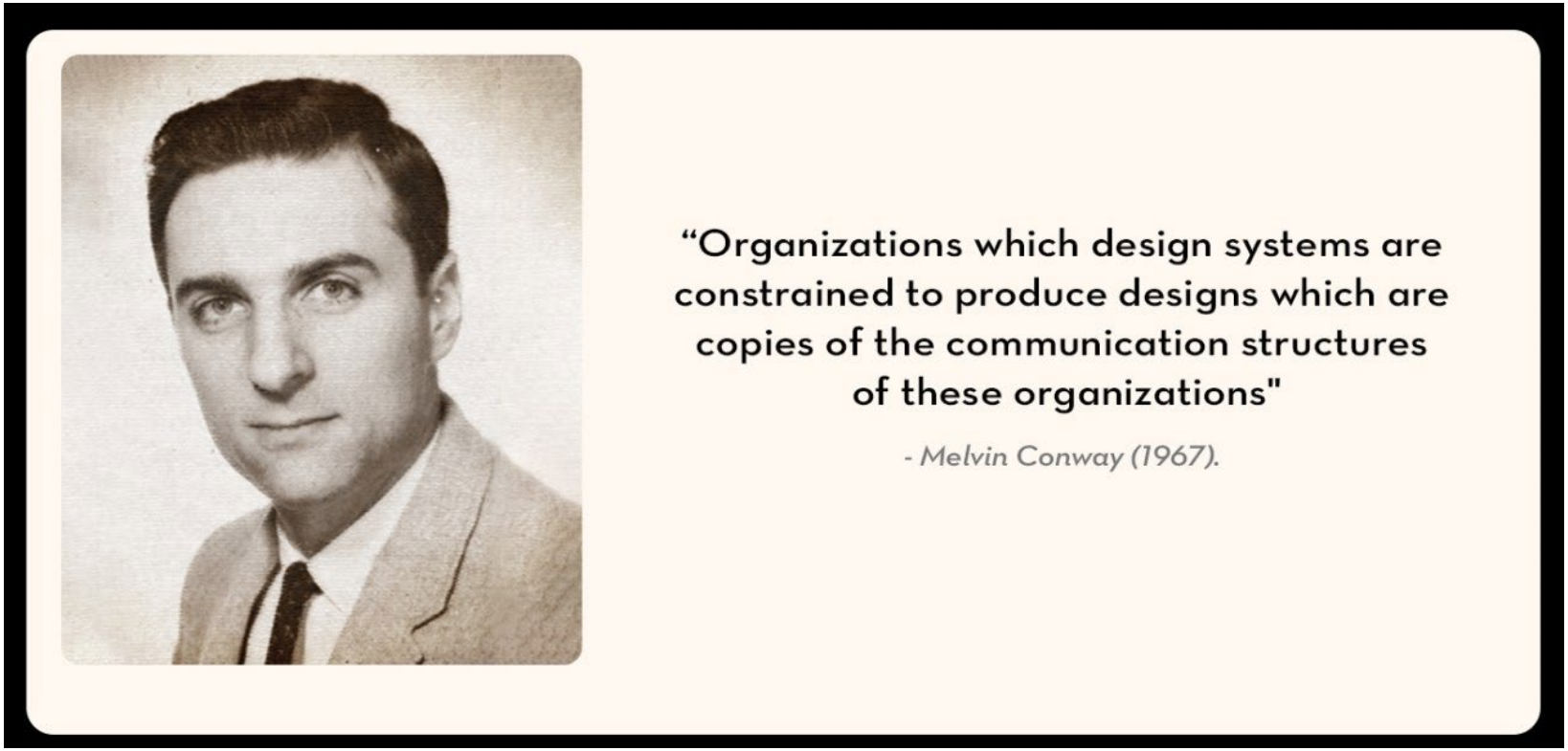


Principle 3:
Hide implementation details

- Design carefully your APIs
- It’s easier to expose some details later than hide them
- Do not share your database!

Principle 4:
Decentralized Governance

- Mind Conway’s Law

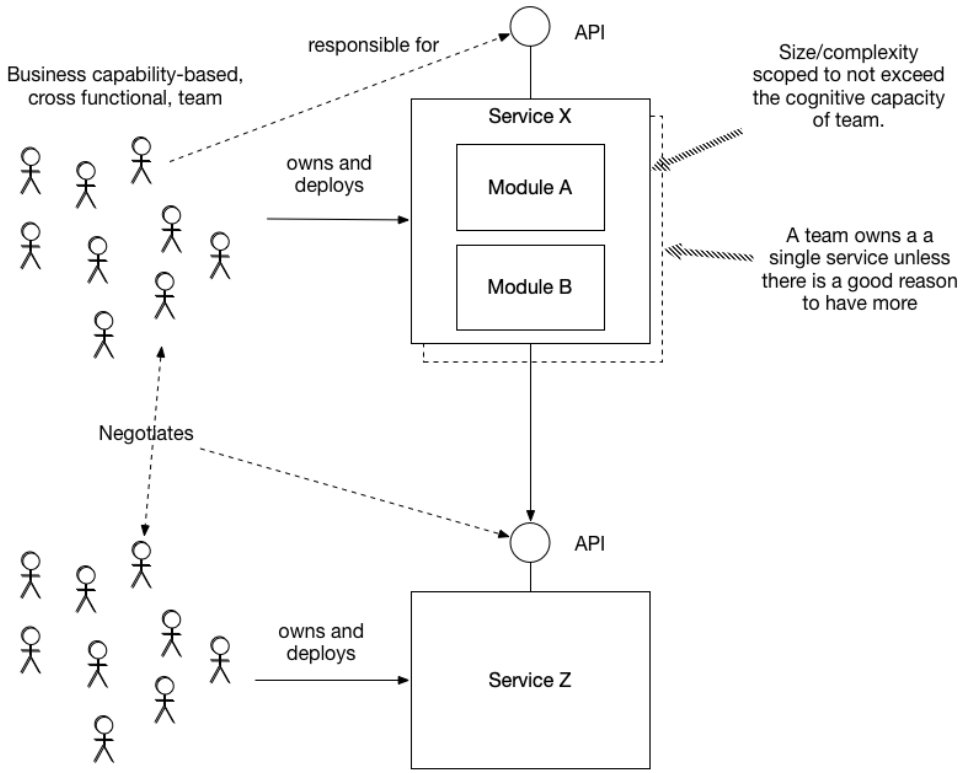


"The traditional model is that you take your software to the wall that separates development and operations, and throw it over and then forget about it. Not at Amazon. You build it, you run it. This brings developers into contact with the day-to-day operation of their software. It also brings them into day-to-day contact with the customer. This customer feedback loop is essential for improving the quality of the service."

-- Werner Vogels in "A conversation with Werner Vogels" in ACM Queue, May 2006

Principle 4:
Decentralized Governance

- Mind Conway’s Law
- You Build It, You Run It
- Embrace team autonomy

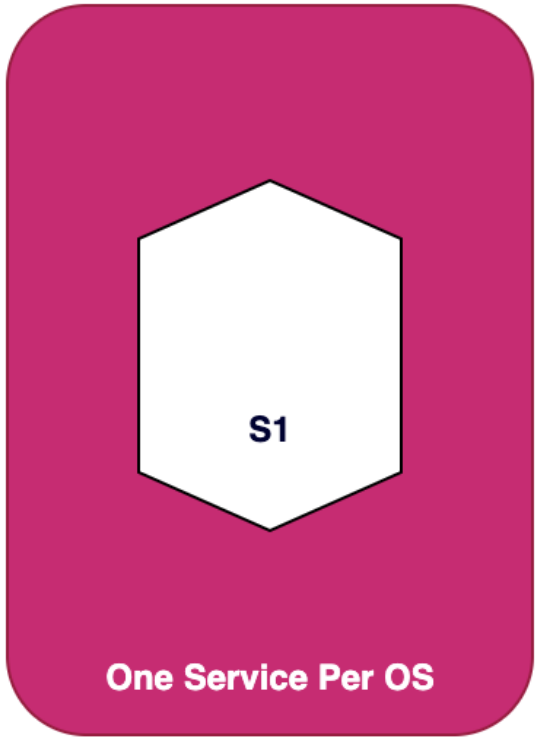

API
responsible for
Business capability-based,
cross functional, team
Size/complexity
scoped to not exceed
the cognitive capacity
of team.
Service X
owns and
deploys
Module A
Module B
A team owns a a
single service unless
there is a good reason
to have more
Negotiates
API
owns and
deploys
Service Z

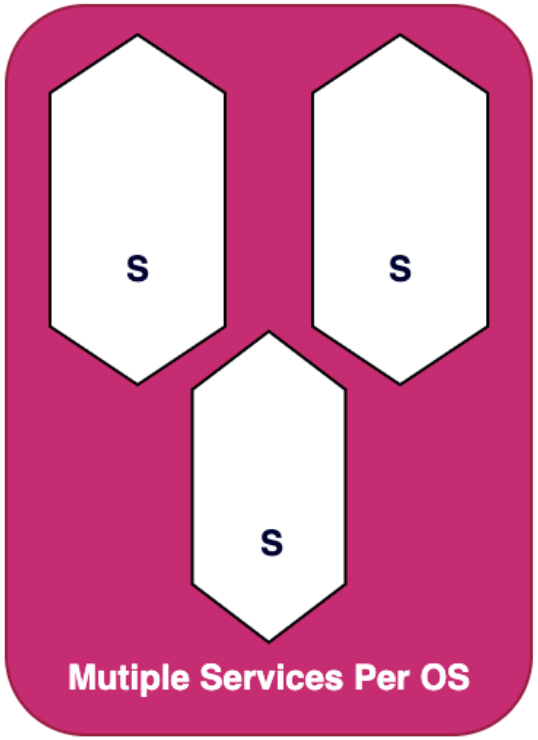
Principle 4:
Decentralized Governance

- Mind Conway’s Law
- You Build It, You Run It
- Embrace team autonomy
- Internal Open Source Model

Principle 5:
Deploy Independently

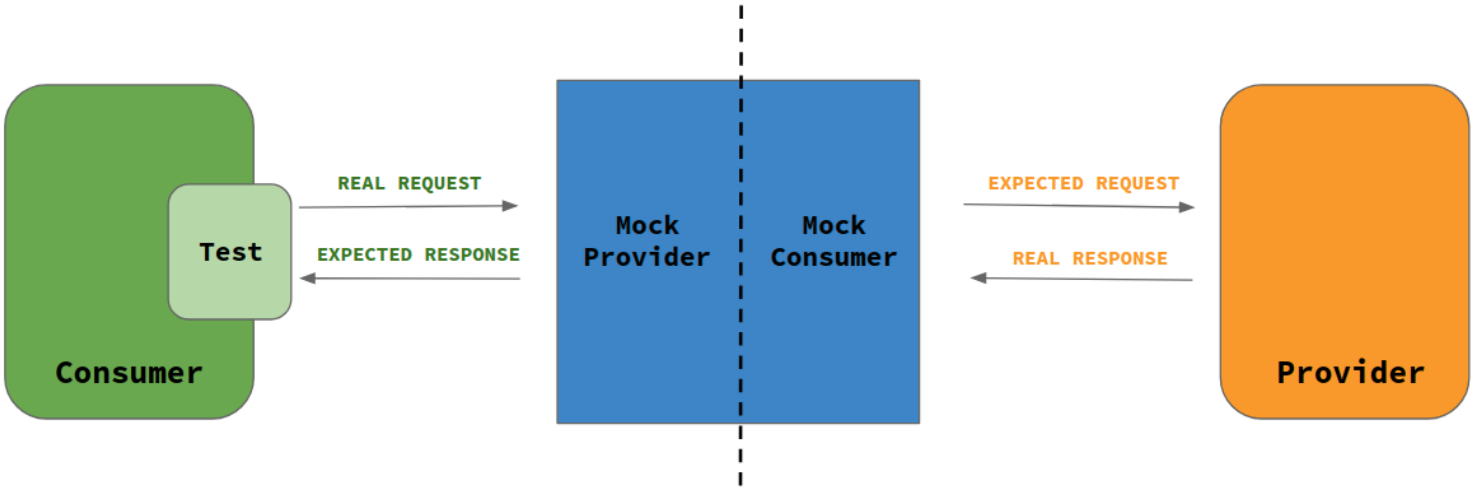
- One Service Per OS





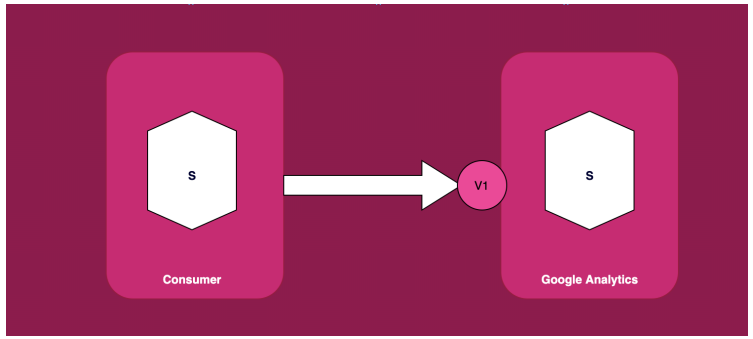
Principle 5:
Deploy Independently

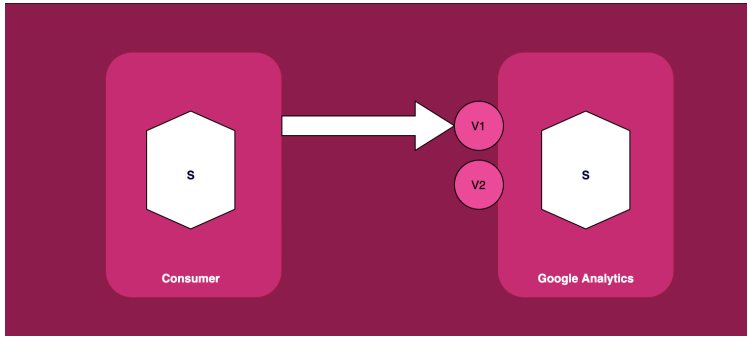
- One Service Per OS
- Consumer-Driven Contracts

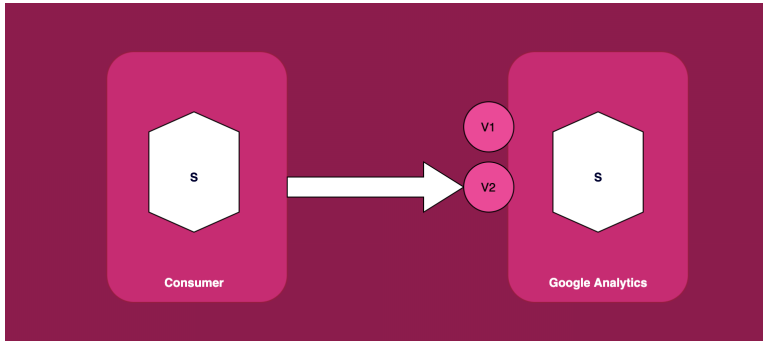


Principle 5:
Deploy Independently

- One Service Per OS
- Consumer-Driven Contracts
- Multiple co-existing versions

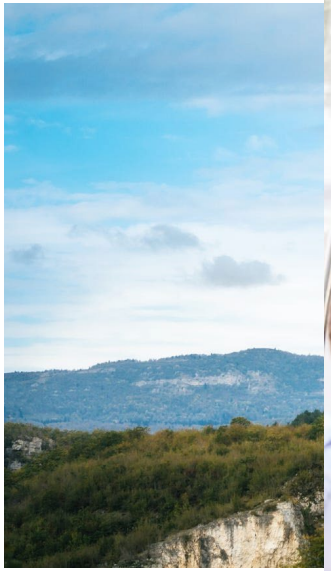






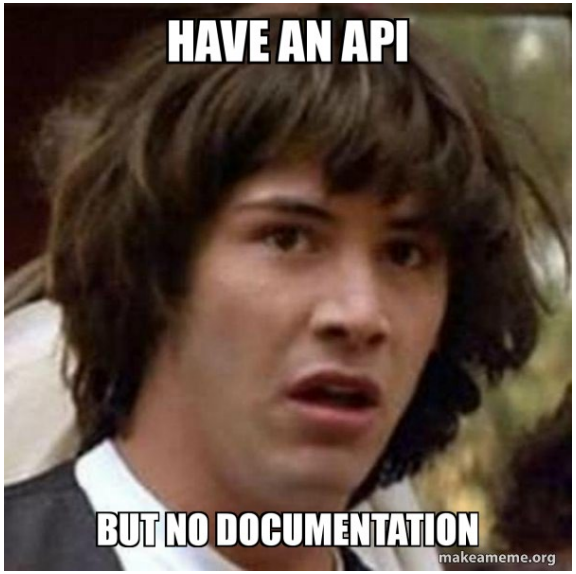
Principle 6:
Consumer First

- Encourage conversations



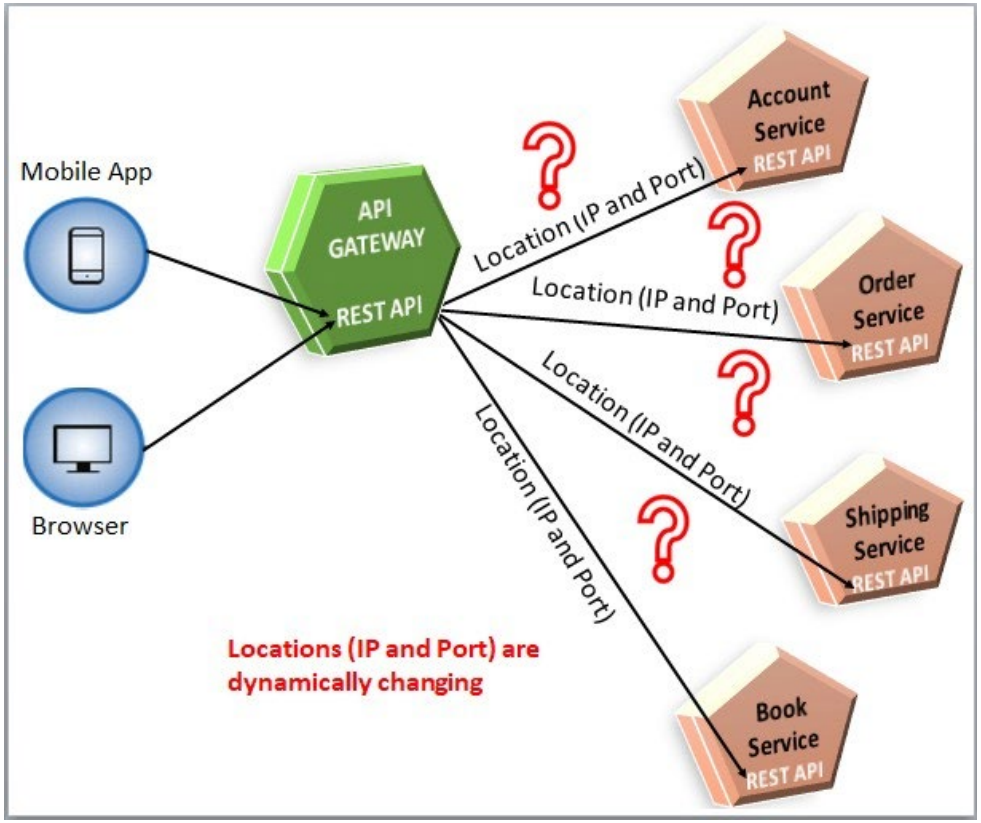
Principle 6: Consumer First

- Encourage conversations
- API Documentation



Principle 6: Consumer First

- Encourage conversations
- API Documentation
- Service Discovery



Principle 7:
Isolate Failure

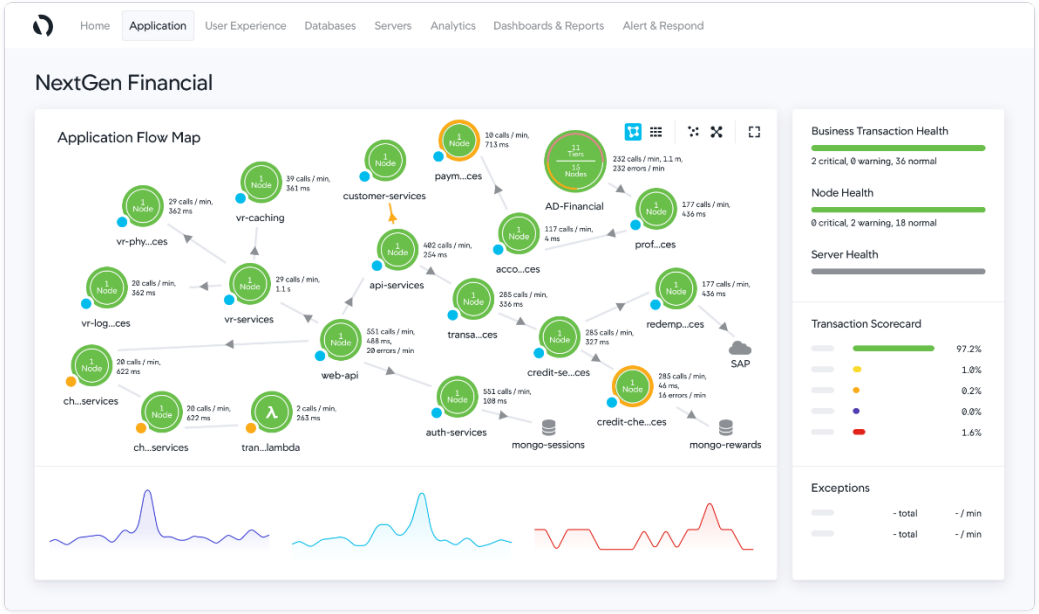
- Avoid cascading failures
- Timeouts between components
- Bulkheading

Principle 7:
Isolate Failure

- Avoid cascading failures
- Timeouts between components
- **Fail fast:** Bulkheading / Circuit breakers

Principle 8:
Highly Observable

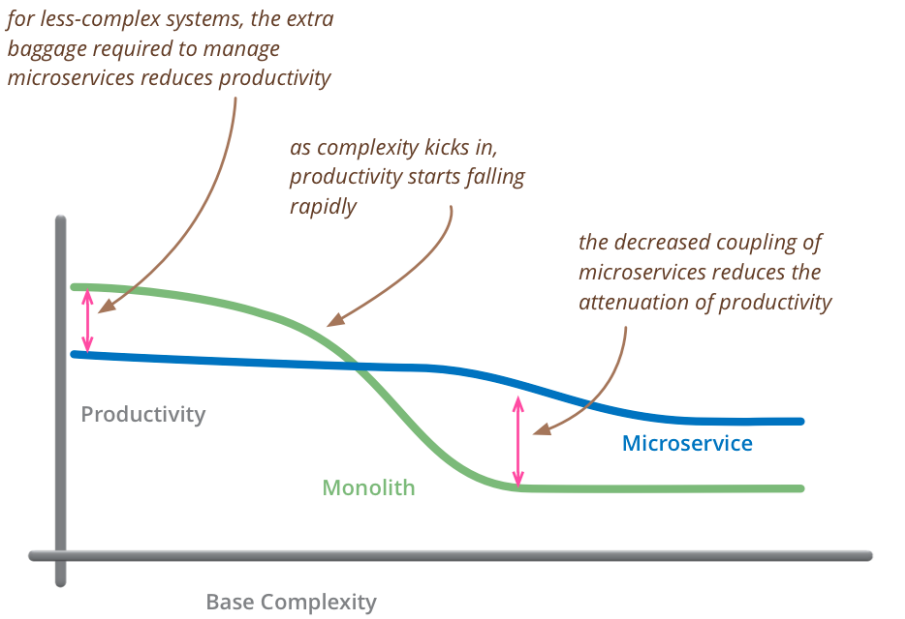
- Standard Monitoring
- Health-Check Pages
- Log and Stats aggregation
- Downstream monitoring



Are microservices always the right choice?

S3D

Microservices overhead



but remember the skill of the team will outweigh any monolith/microservice choice

Taking it to the extreme

SERVERLESS

Serverless (Functions-as-a-Service)

- Instead of writing minimal services, write just functions
- No state, rely completely on cloud storage or other cloud services
- Pay-per-invocation billing with elastic scalability
- Drawback: more ways things can fail, state is expensive
- Examples:
  AWS lambda, CloudFlare workers, Azure Functions
- What might this be good for?

- (New in 2019/20) Stateful Functions:
  Azure Durable Entities, CloudFlare Durable Objects