Architecture: Microservices

17-313 Fall 2025

Foundations of Software Engineering

https://cmu-17313q.github.io

Eduardo Feo Flushing

Learning Goals

- Contrast the monolithic application design with a modular design based on microservices.
- Reason about tradeoffs of microservices architectures.
- Principles of microservices: how to benefit and avoid their pitfalls

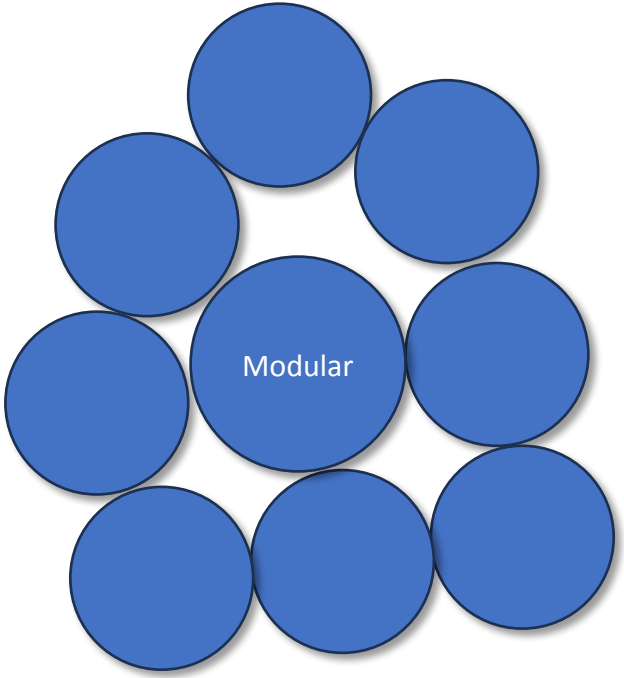
Outline

- From Monoliths to Service Oriented Architecture
  - Case Study: Chrome Web Browser
- Microservices
  - Monolith vs Microservices
  - Advantages
  - Challenges
- Microservices: Principles
- Serverless

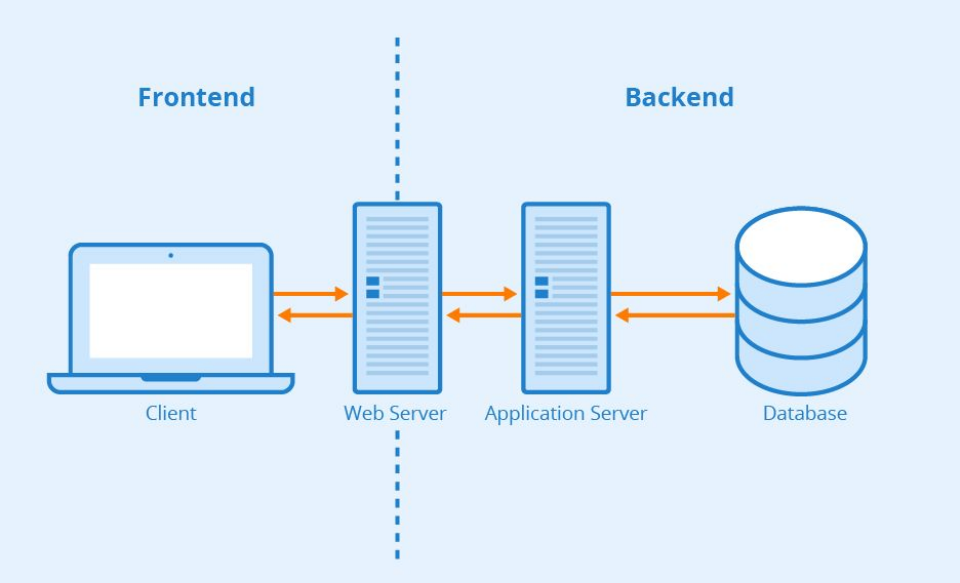
Before we get to microservices…

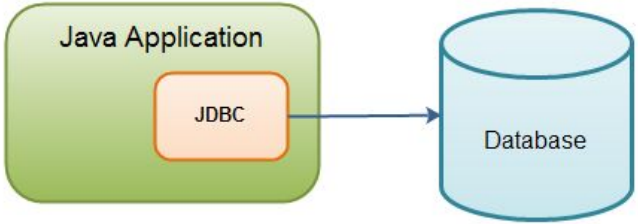
MONOLITHS


Monolithic


Modular

Monolithic styles

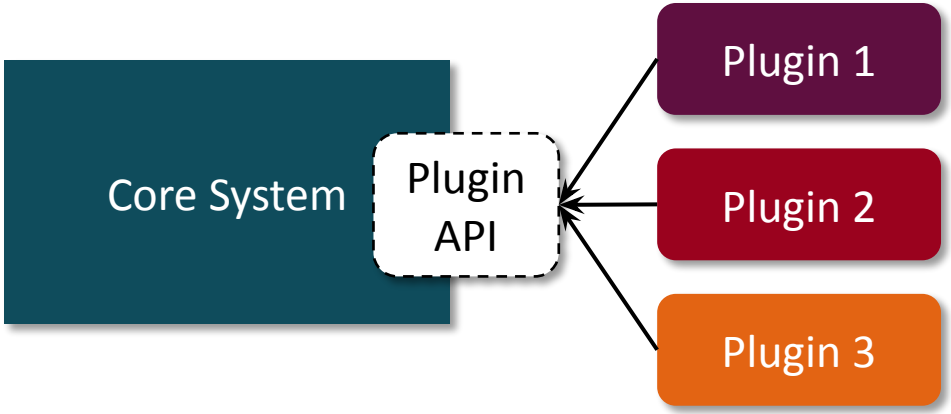




Source: https://www.seobility.net (CC BY-SA 4.0)

Modularity comes in many ways

- **Plug-in architectures**
  - Distinct code repositories, joined to a monolithic run-time
  - Examples: Linux kernel modules, NodeBB themes, VS Code extensions
  - Separates development, but runs as “one”



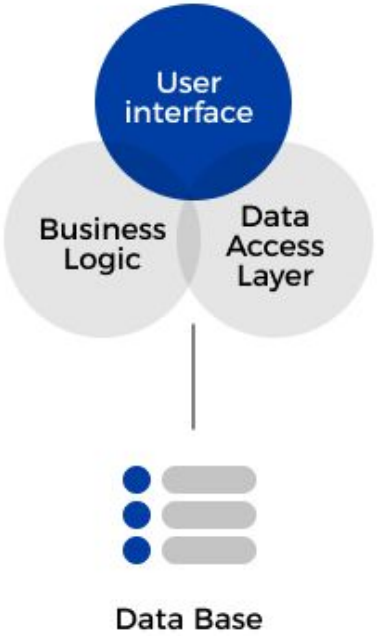
Modularity comes in many ways

- Plug-in architectures
- Service-oriented architectures
  - Distinct processes communicating via messages (e.g., Web browsers)
  - Separates run-time resource management and failure / security issues.
- Distributed microservices
  - Independent, autonomous services communicating via web APIs
  - Separates almost all concerns

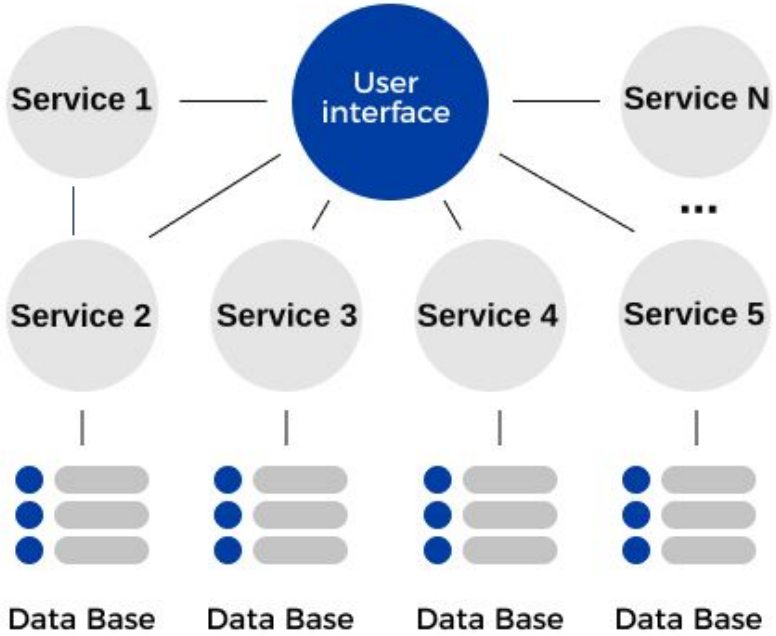
Separation of concerns

SERVICE-ORIENTED ARCHITECTURE

Monolithic

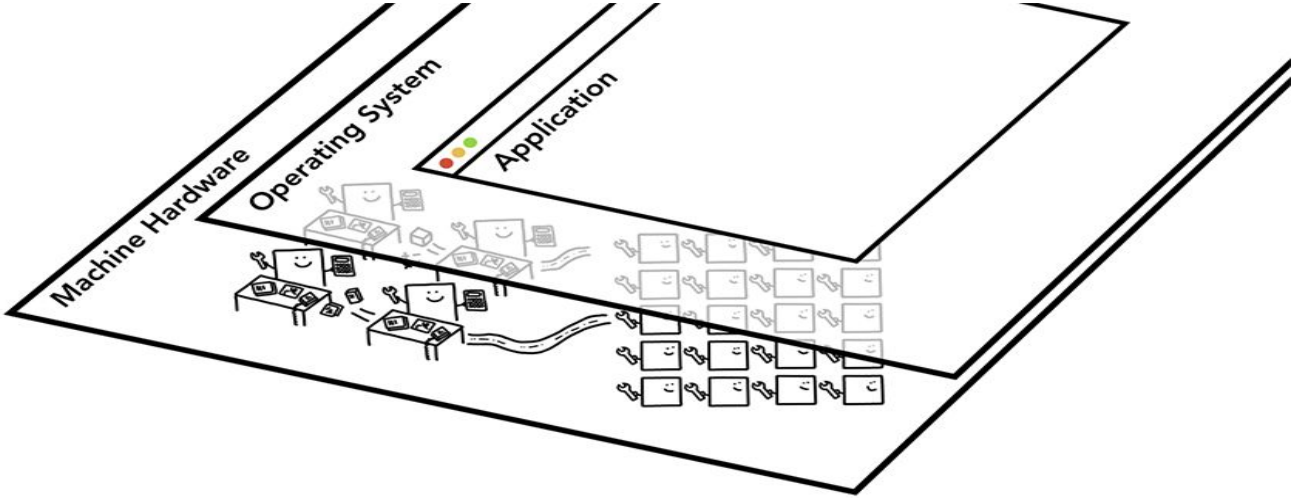


Service-oriented



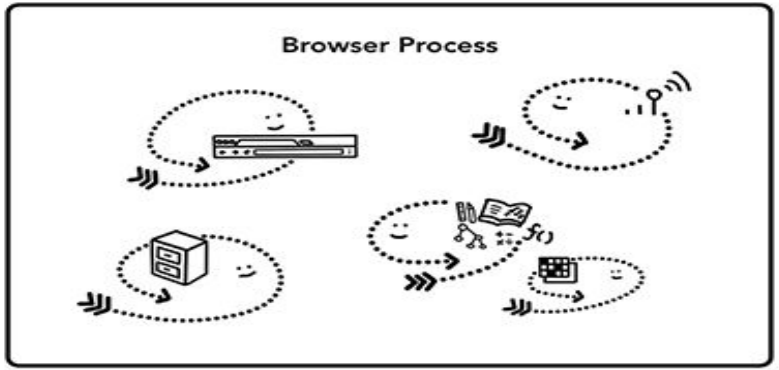
Example: Chrome

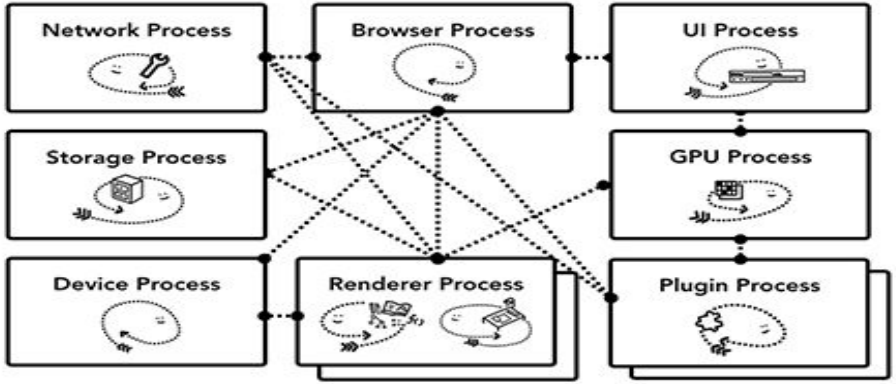
Web Browsers



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

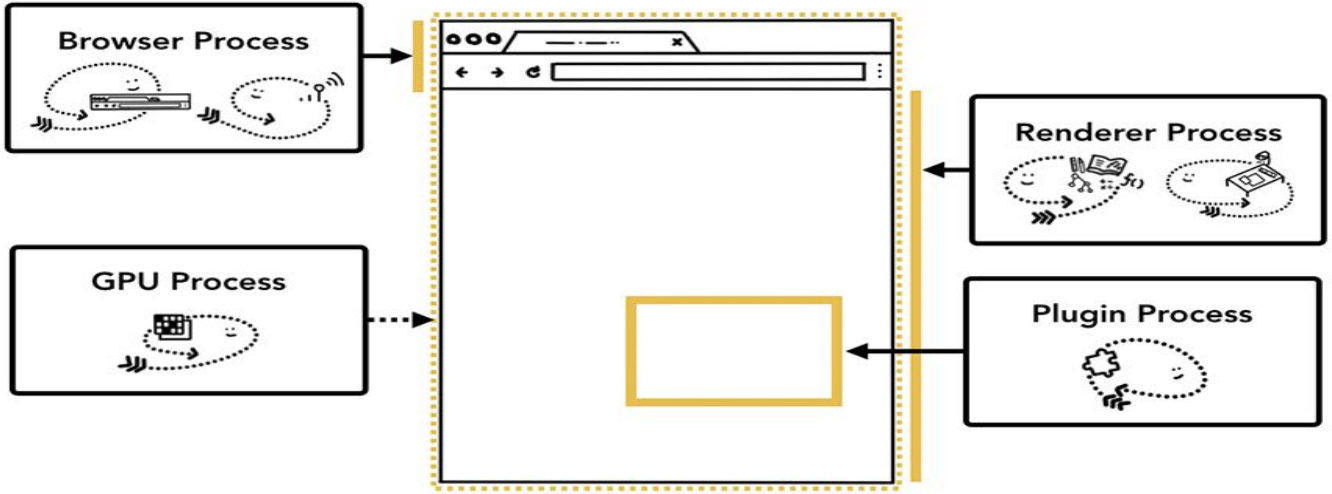
The evolution of browser architectures





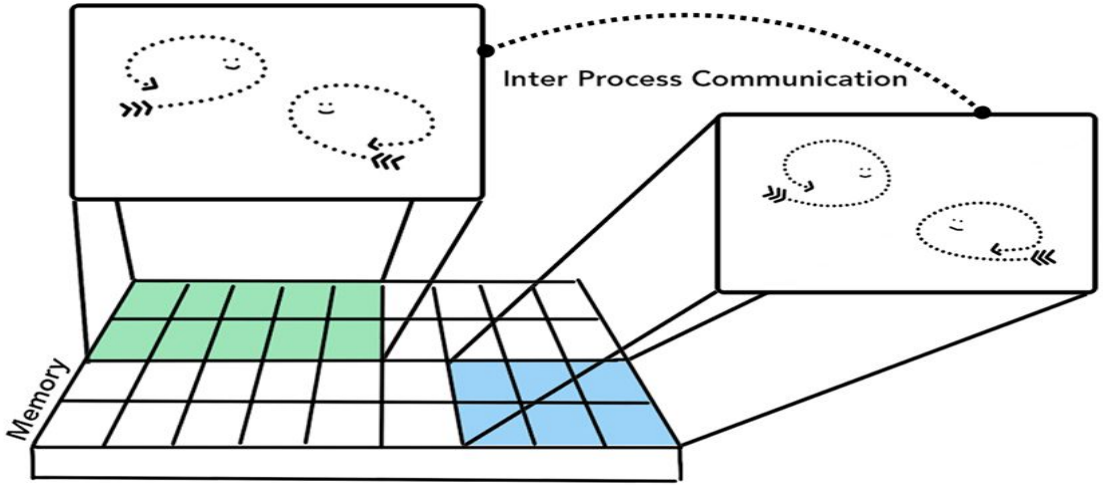
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Service-oriented browser architecture



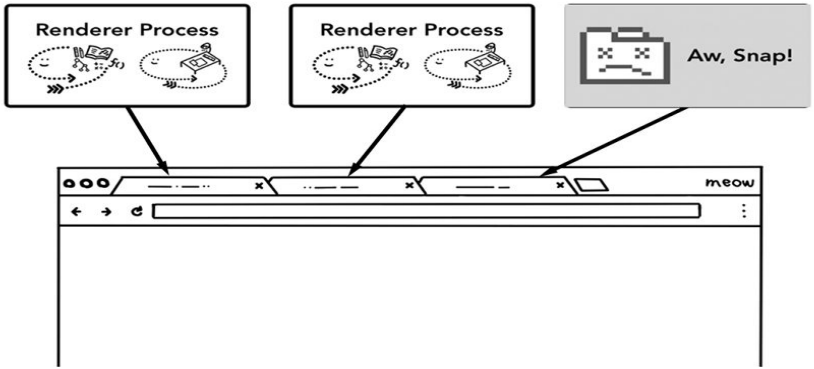
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

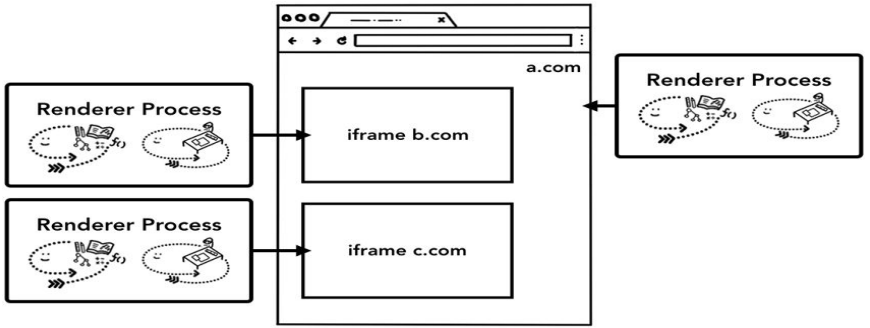
Implementation:
Multi-process browser with IPC



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

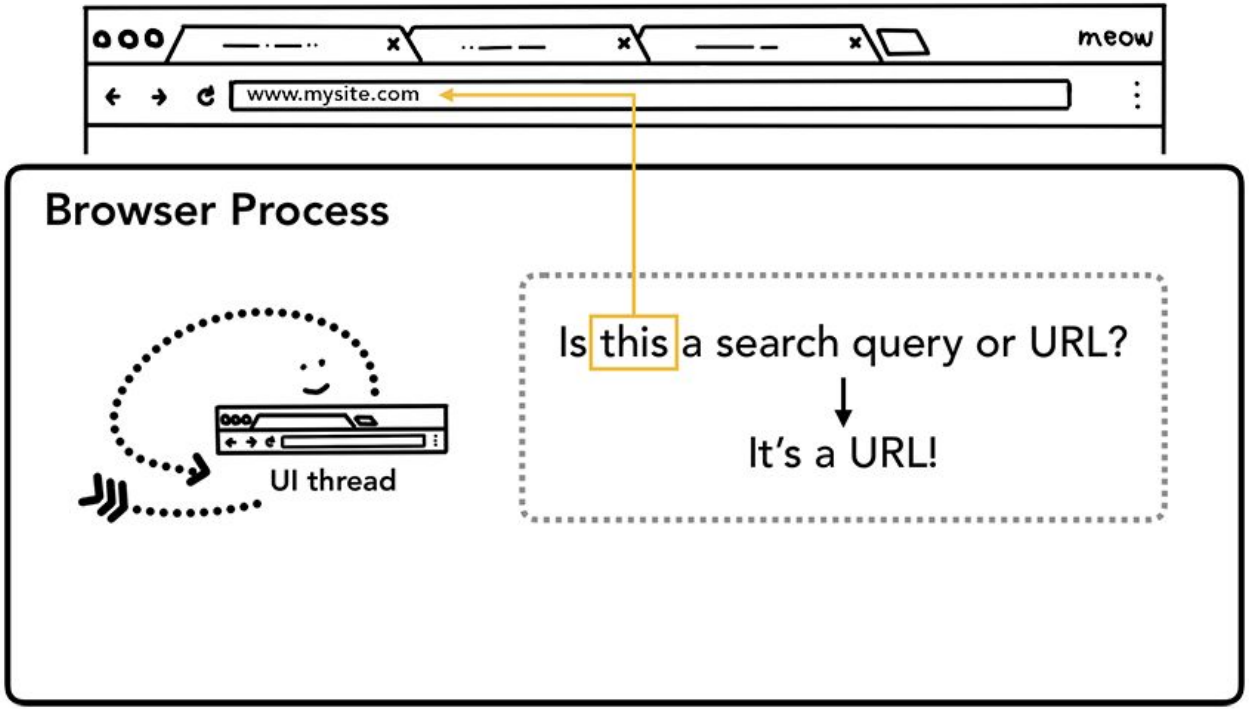
Benefits of a service-oriented browser architecture





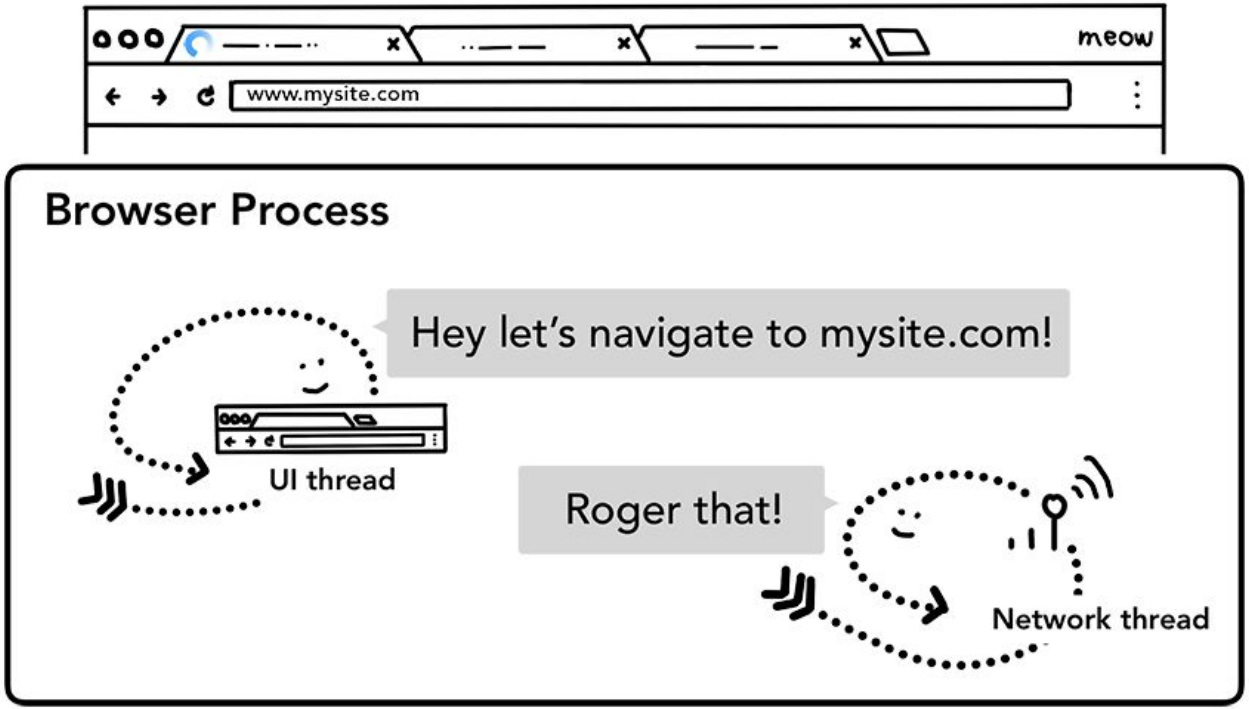
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Navigating to a web site uses service requests



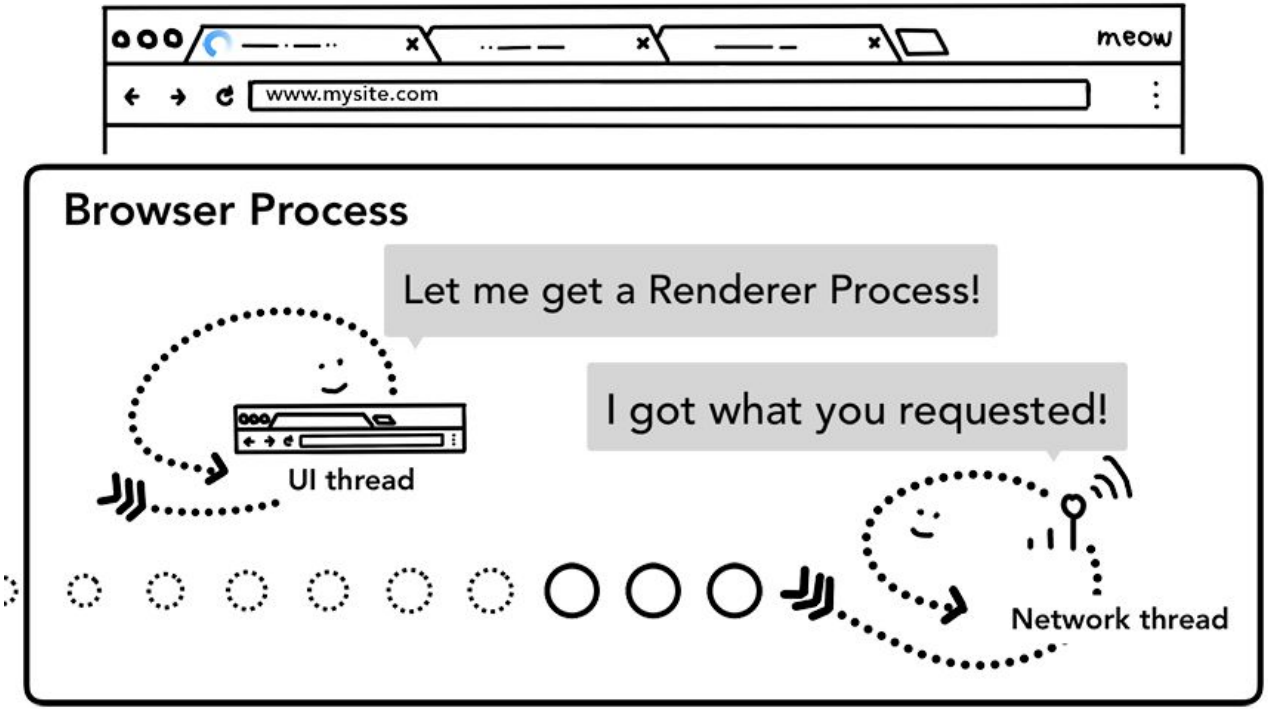
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Navigating to a web site uses service requests



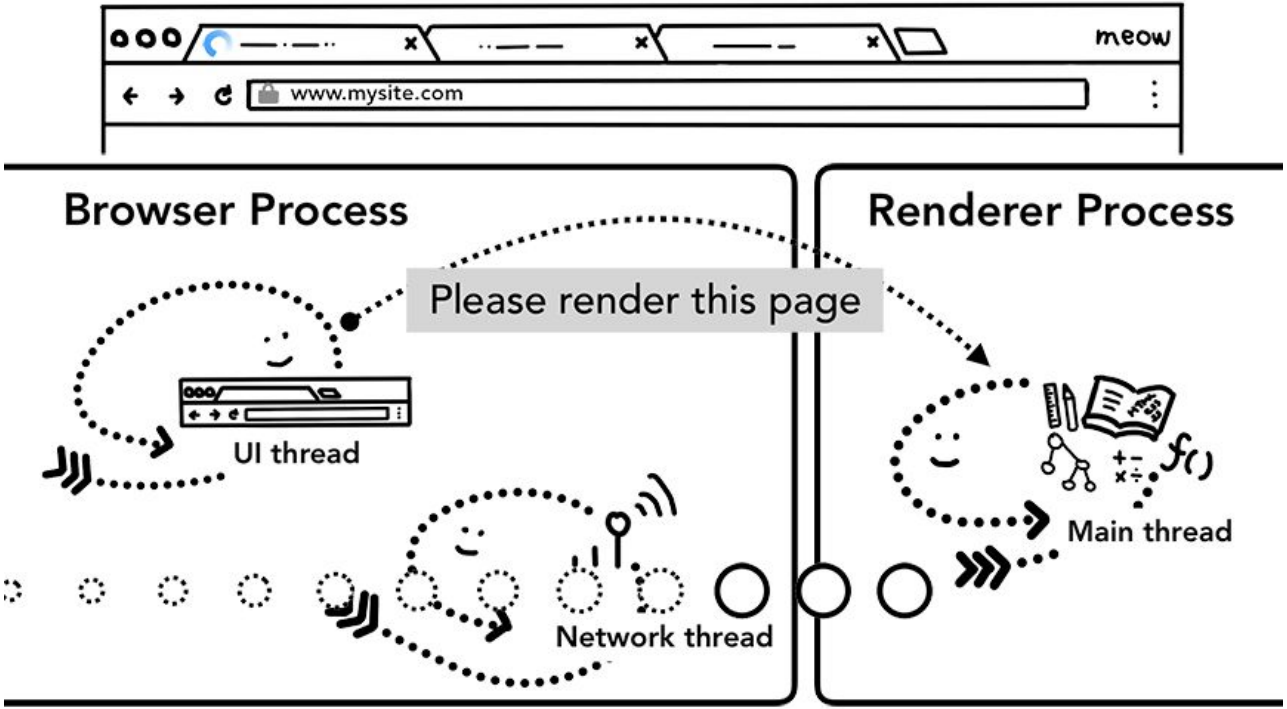
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Navigating to a web site uses service requests



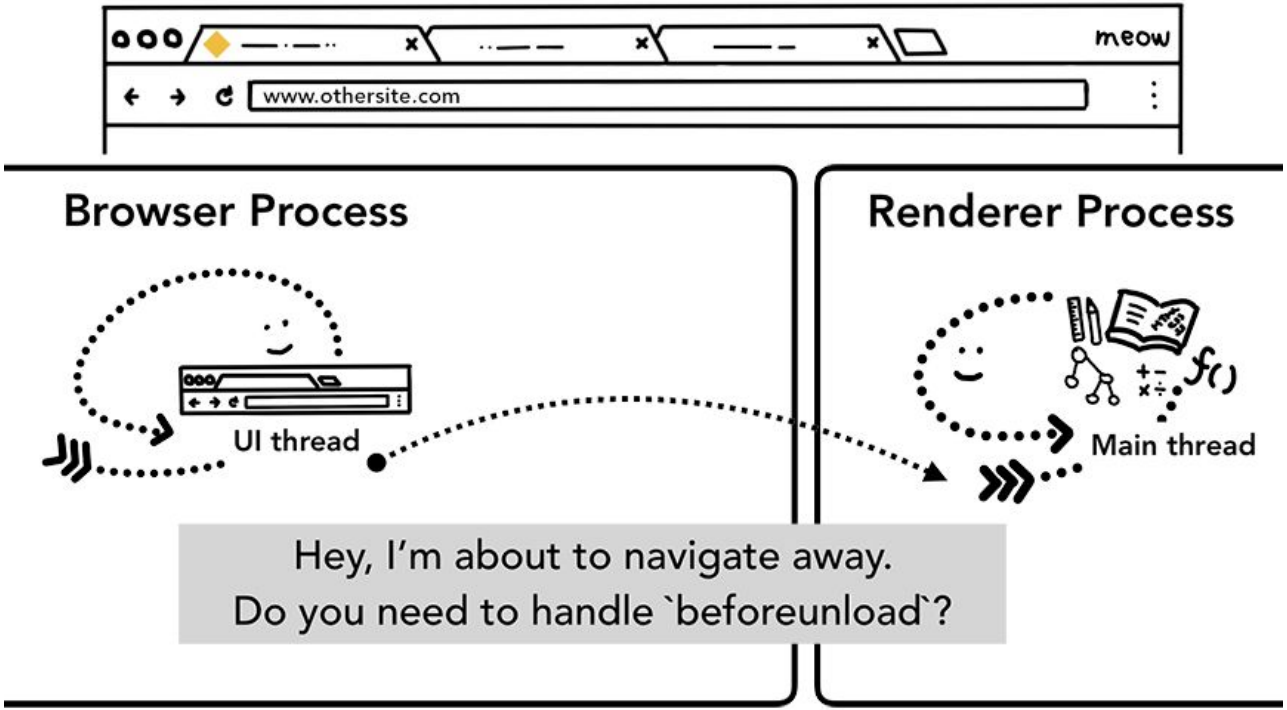
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Navigating to a web site uses service requests



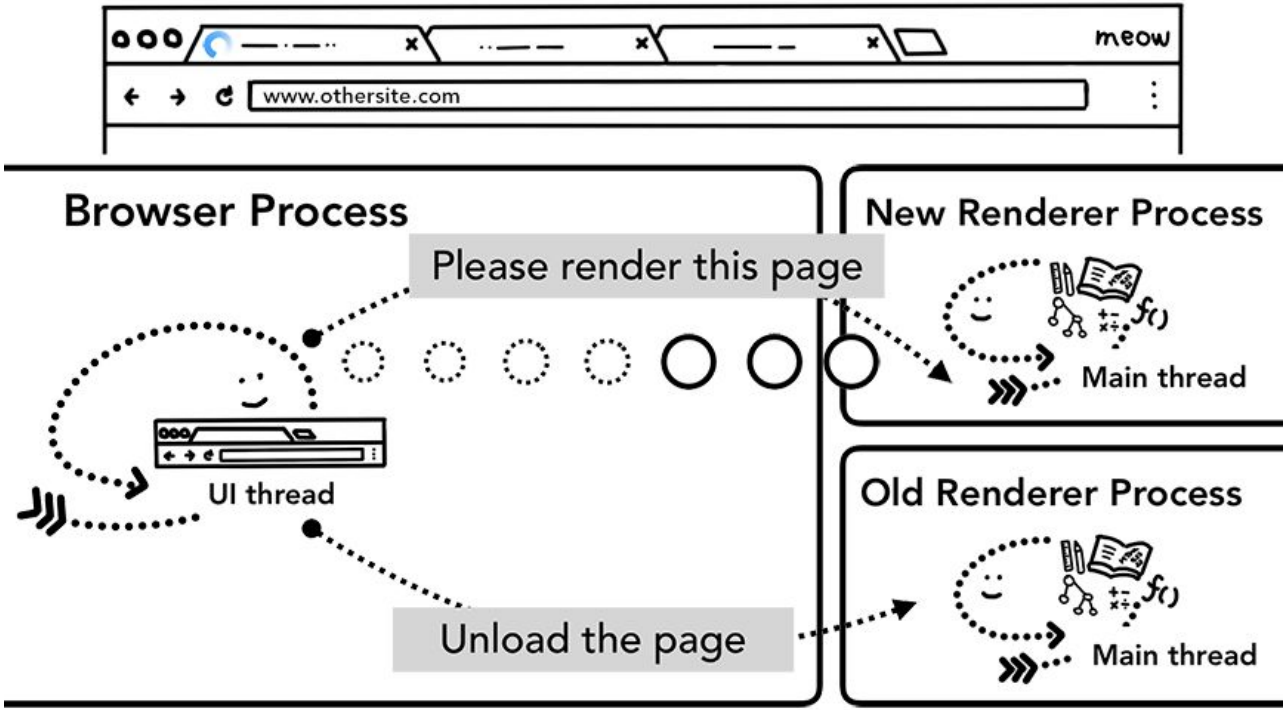
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Navigating to a web site uses service requests

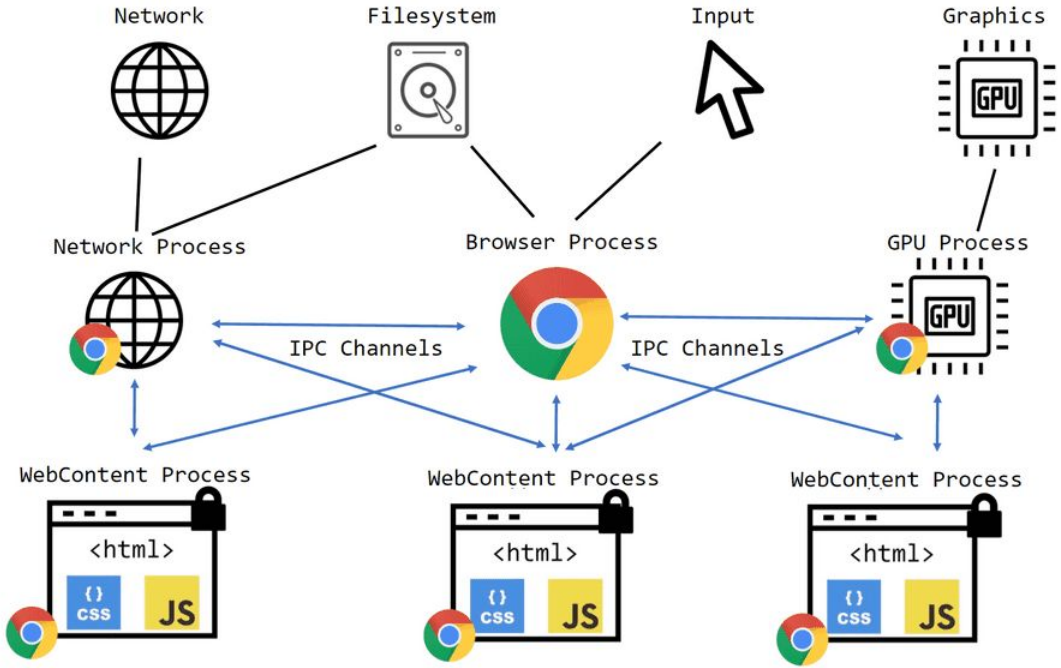


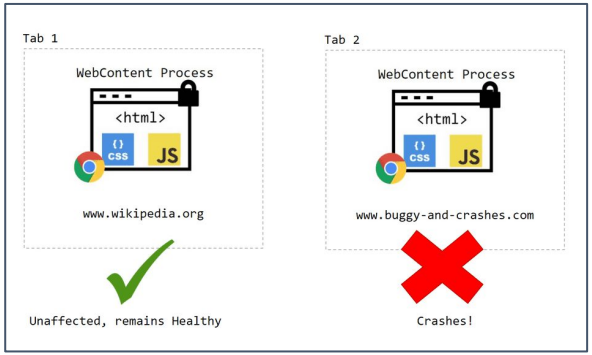
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

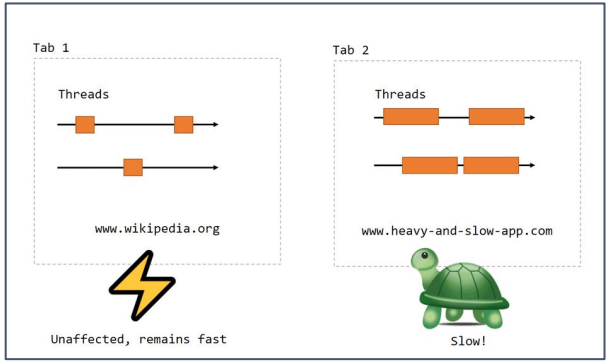
Navigating to a web site uses service requests



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







https://webperf.tips/tip/browser-process-model/

One browser, many processes

```
 547187 ?        SNl    0:00 /opt/google/chrome/chrome_crashpad_handler --no-periodic-tasks --m
--annotation=ver=128.0.6613.137 --initial-client-fd=4 --shared-client-connection
 547193 ?        SN     0:00 /opt/google/chrome/chrome --type=zygote --no-zygote-sandbox --crash
 547194 ?        SN     0:00 /opt/google/chrome/chrome --type=zygote --crashpad-handler-pid=547
 547196 ?        SN     0:04 /opt/google/chrome/chrome --type=zygote --crashpad-handler-pid=547185 --enabl                    eporter=, --cha
 547220 ?        SNl   48:08 /opt/google/chrome/chrome --type=gpu-process --crashpad-handler-pid=547185 --enable-crash-reporter=
--shared-files --field-trial-handle=3,i,16924338087686289425,14108195033817998739,262144 --disable-features=EyeDropper --var
 547222 ?        SNl   21:09 /opt/google/chrome/chrome --type=utility --utility-sub-type=network.mojom.NetworkService --lang=en-US
--disable-features=EyeDropper --variations-seed-version=20240923-050122.947000
 547224 ?        SNl    0:14 /opt/google/chrome/chrome --type=utility --utility-sub-type=storage.mojom.StorageService --lang=en-US -
--disable-features=EyeDropper --variations-seed-version=20240923-050122.947000
 547410 ?        SNl    2:30 /opt/google/chrome/chrome --type=renderer --crashpad-handler-pid=547185 --enable-crash-reporter=, --r
--shared-files=v8_context_snapshot_data:100 --field-trial-handle=3,i,16924338087686289425,14108195033817998739,262144 --di
 547533 ?        SNl    1:23 /opt/google/chrome/chrome --type=renderer --crashpad-handler-pid=547185 --enable-crash-reporter=, --r
--shared-files=v8_context_snapshot_data:100 --field-trial-handle=3,i,16924338087686289425,14108195033817998739,262144 --di
 547594 ?        SNl   10:27 /opt/google/chrome/chrome --type=utility --utility-sub-type=audio.mojom.AudioService --lang=en-US --ser
--disable-features=EyeDropper --variations-seed-version=20240923-050122.947000
 547755 ?        SNl    0:08 /opt/google/chrome/chrome --type=renderer --crashpad-handler-pid=547185 --enable-crash-reporter=, --r
--shared-files=v8_context_snapshot_data:100 --field-trial-handle=3,i,16924338087686289425,14108195033817998739,262144 --di
 554653 ?        SNl    2:43 /opt/google/chrome/chrome --type=renderer --crashpad-handler-pid=547185 --enable-crash-reporter=, --e
--shared-files=v8_context_snapshot_data:100 --field-trial-handle=3,i,16924338087686289425,14108195033817998739,262144 --di
 554722 ?        SNl    0:11 /opt/google/chrome/chrome --type=renderer --crashpad-handler-pid=547185 --enable-crash-reporter=, --n
--shared-files=v8_context_snapshot_data:100 --field-trial-handle=3,i,16924338087686289425,14108195033817998739,262144 --di
.....
1500520 ?       SNl    0:00 /opt/google/chrome/chrome --type=utility --utility-sub-type=data_decoder.mojom.DataDecoderService --la
--field-trial-handle=3,i,16924338087686289425,14108195033817998739,262144 --disable-features=EyeDropper --variations-seed-v
1500532 ?       SNl    0:00 /opt/google/chrome/chrome --type=renderer --crashpad-handler-pid=547185 --enable-crash-reporter=, --
```

Service-oriented architecture: More benefits

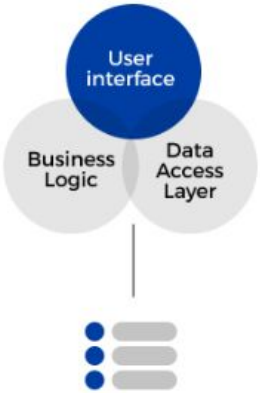
- Ability to change components independently
- Independent processes (Isolation, Security)
- Focusing on doing one thing well

MICROSERVICES

Why do we need microservices if we already have modular architectures and SOA?

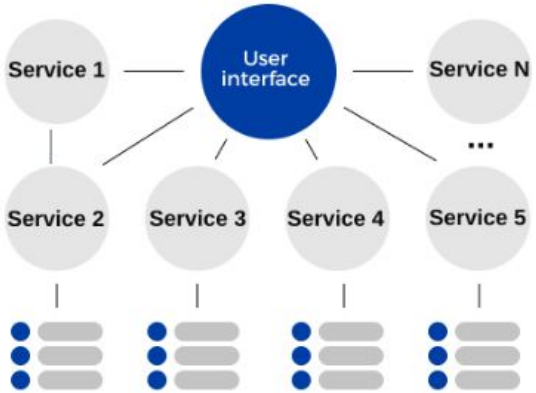
- **Monoliths**:
  - Often lack modularity, and even “modular” monoliths are tightly coupled in runtime.
- **Service oriented architectures:**
  - More flexible, but often heavyweight and centralized.

Monolithic



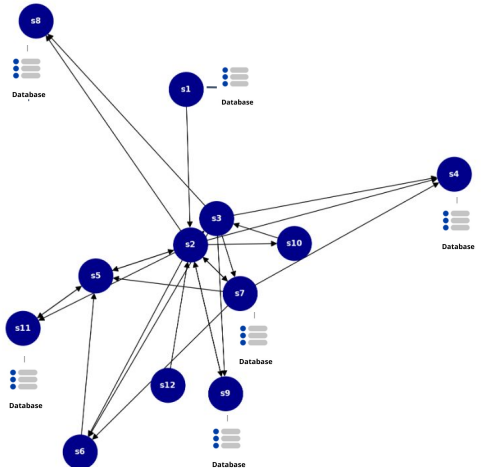
Database

Service oriented



Database Database Database Database

Microservices




Microservices
Microservices Everywhere

"*Small autonomous services that work well together*"

Sam Newman

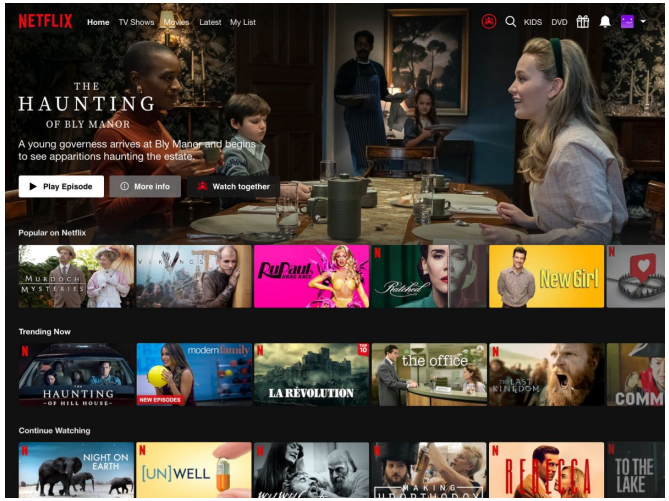
Microservices

COMCAST

UBER

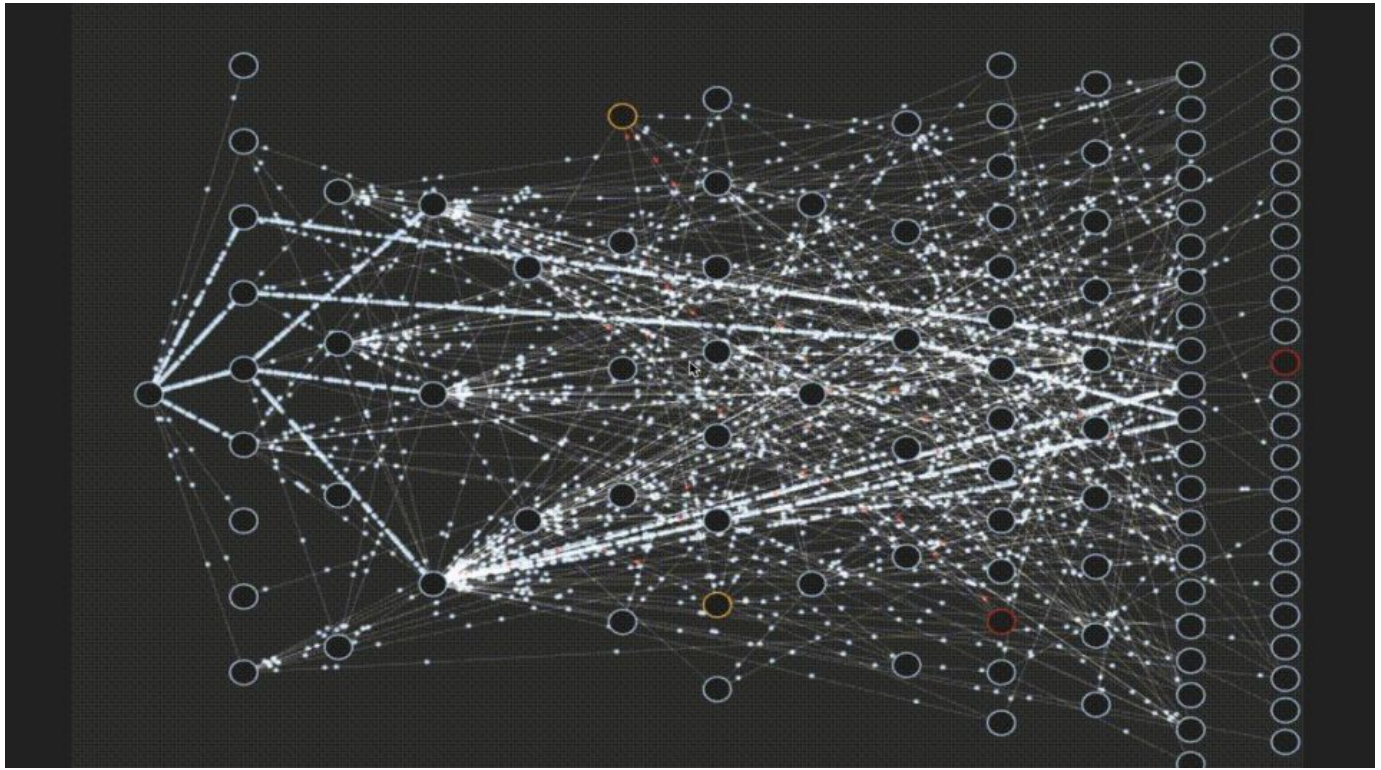
GROUPON®

Netflix Microservices – App Boot

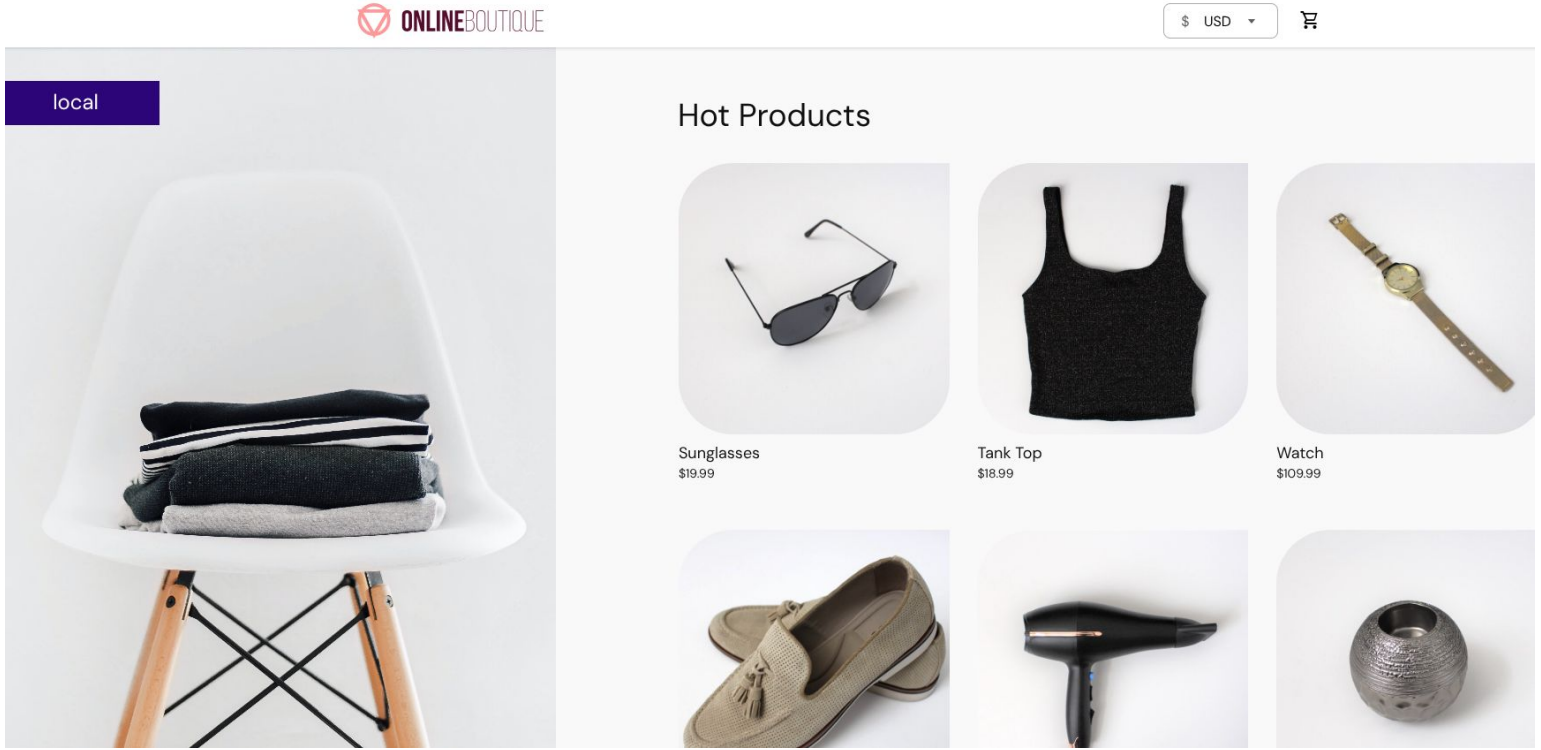


- Recommendations
- Trending Now
- Continue Watching
- My List
- Metrics

(as of 2016)

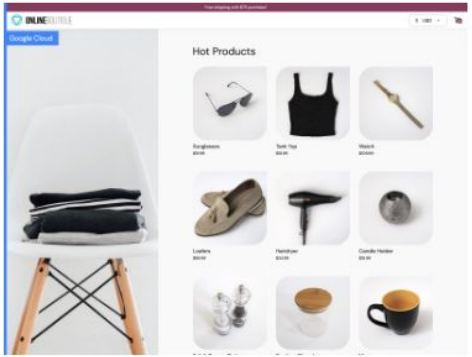


Microservices: Demo

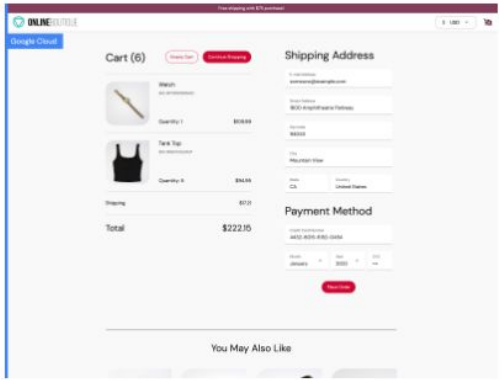


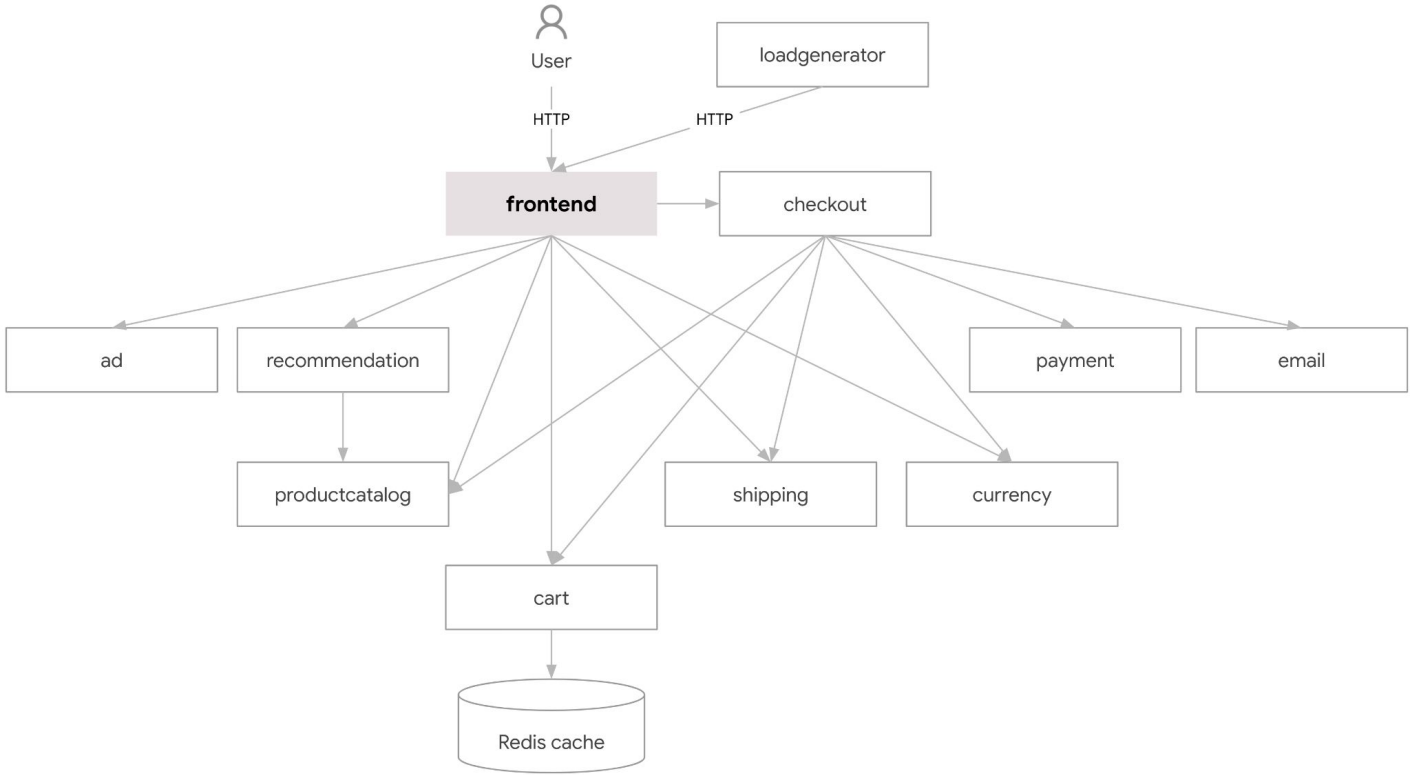
Service	Language	Description
frontend	Go	Exposes an HTTP server to serve the website. Does not require signup/login and generates session IDs for all users automatically.
cartservice	C#	Stores the items in the user's shopping cart in Redis and retrieves it.
productcatalogservice	Go	Provides the list of products from a JSON file and ability to search products and get individual products.
currencyservice	Node.js	Converts one money amount to another currency. Uses real values fetched from European Central Bank. It's the highest QPS service.
paymentservice	Node.js	Charges the given credit card info (mock) with the given amount and returns a transaction ID.
shippingservice	Go	Gives shipping cost estimates based on the shopping cart. Ships items to the given address (mock)
emailservice	Python	Sends users an order confirmation email (mock).
checkoutservice	Go	Retrieves user cart, prepares order and orchestrates the payment, shipping and the email notification.
recommendationservice	Python	Recommends other products based on what's given in the cart.
adservice	Java	Provides text ads based on given context words.
loadgenerator	Python/Locust	Continuously sends requests imitating realistic user shopping flows to the frontend.

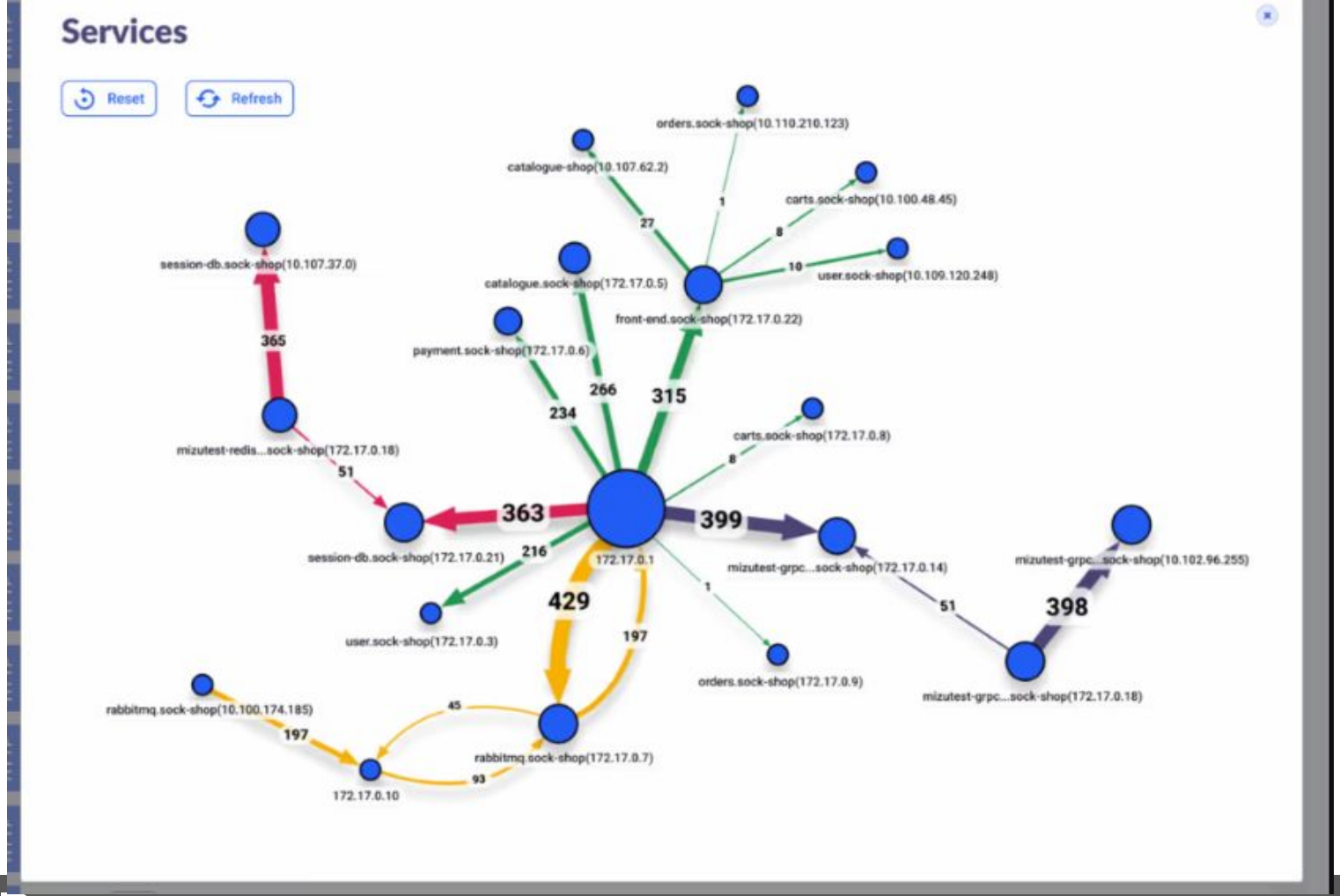
Home Page



Checkout Screen






Services
Reset
Refresh
orders.sock-shop(10.110.210.123)
catalogue-shop(10.107.62.2)
carts.sock-shop(10.100.48.45)
session-db.sock-shop(10.107.37.0)
catalogue.sock-shop(172.17.0.5)
user.sock-shop(10.109.120.248)
front-end.sock-shop(172.17.0.22)
payment.sock-shop(172.17.0.6)
carts.sock-shop(172.17.0.8)
mizutest-redis...sock-shop(172.17.0.18)
session-db.sock-shop(172.17.0.21)
172.17.0.1
mizutest-grpc...sock-shop(172.17.0.14)
mizutest-grpc...sock-shop(10.102.96.255)
user.sock-shop(172.17.0.3)
orders.sock-shop(172.17.0.9)
mizutest-grpc...sock-shop(172.17.0.18)
rabbitmq.sock-shop(10.100.174.185)
rabbitmq.sock-shop(172.17.0.7)
172.17.0.10
1
27
8
10
365
234
266
315
51
363
399
216
429
197
45
93
398

Monoliths vs Microservices

Activity: In teams of 3-4

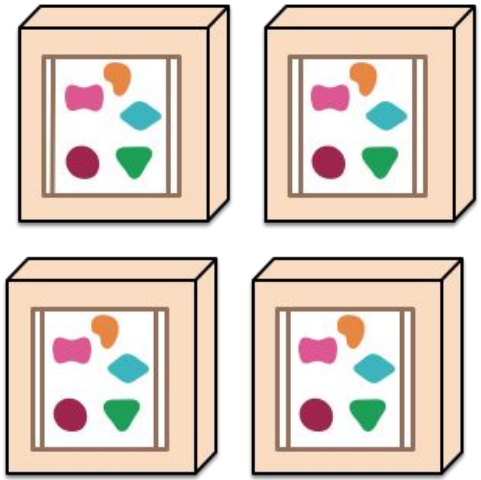
What are the consequences of this architecture? On:

- Scalability
- Reliability
- Performance
- Development
- Maintainability
- Testability
- Ownership

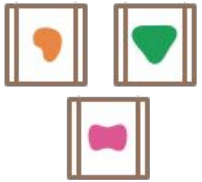
Scalability

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

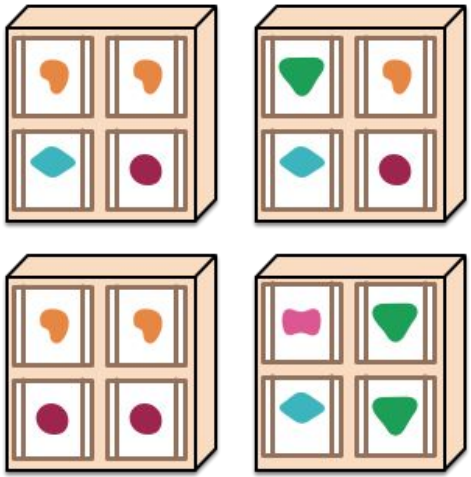
... and scales by replicating the monolith on multiple servers



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



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

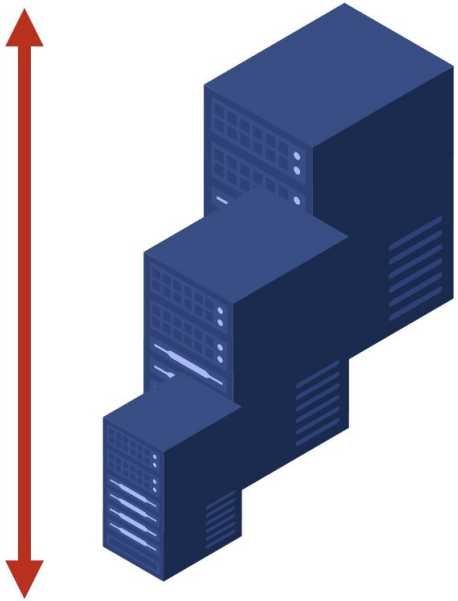


Source: http://martinfowler.com/articles/microservices.html

Types of scaling: vertical vs. horizontal

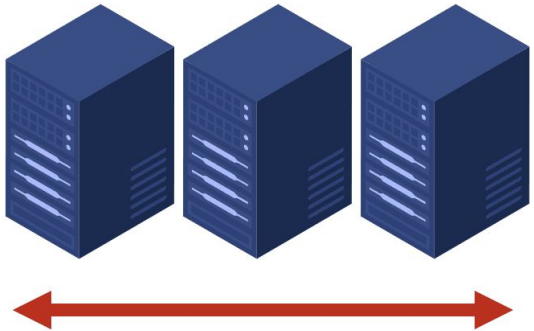
Vertical Scaling

Increase or decrease the capacity of existing services/instances.

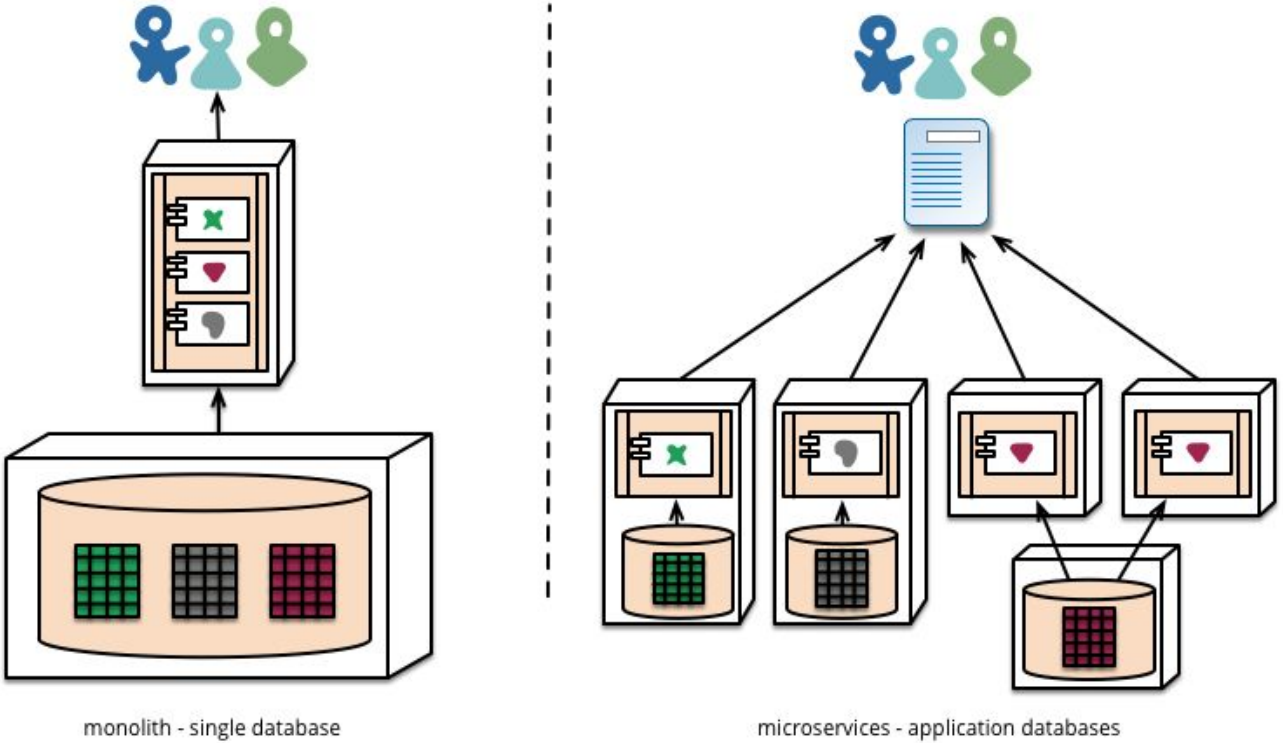


Horizontal Scaling

Add more resources like virtual machines to your system to spread out the workload across them.



Data management and consistency

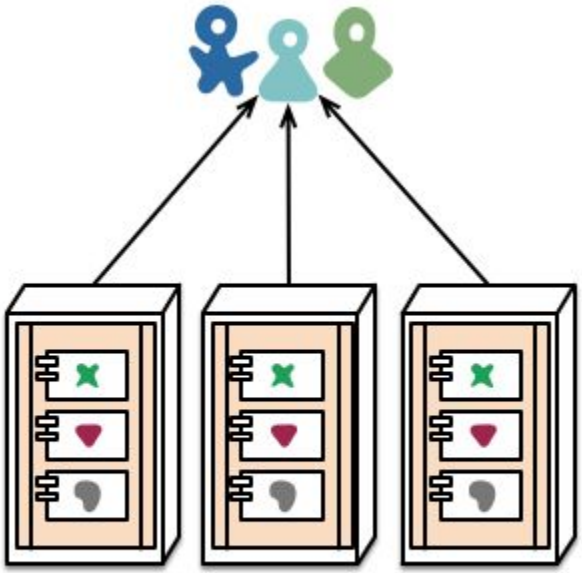


monolith - single database

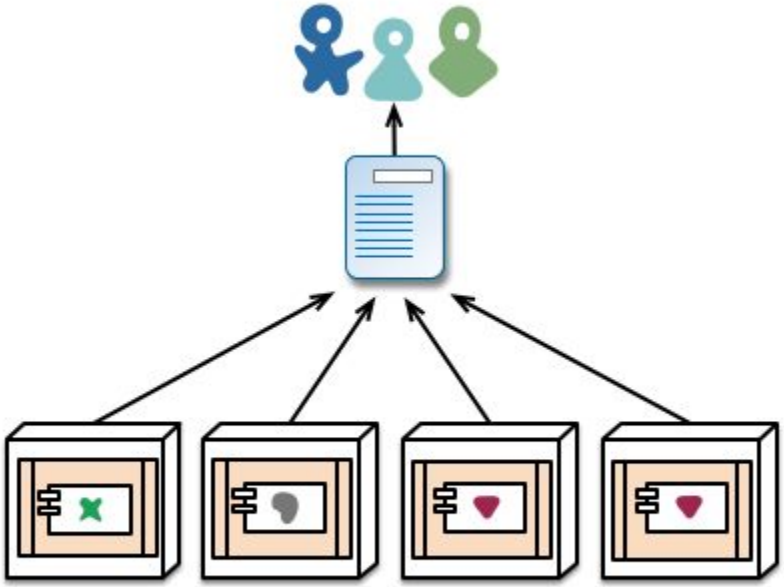
microservices - application databases

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

Deployment and Evolution



monolith - multiple modules in the same process

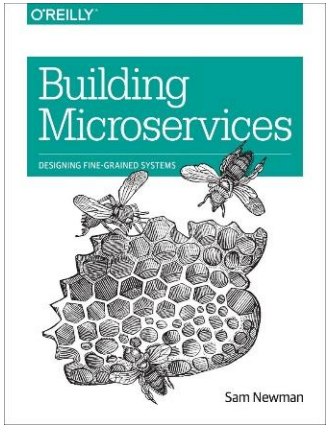


microservices - modules running in different processes

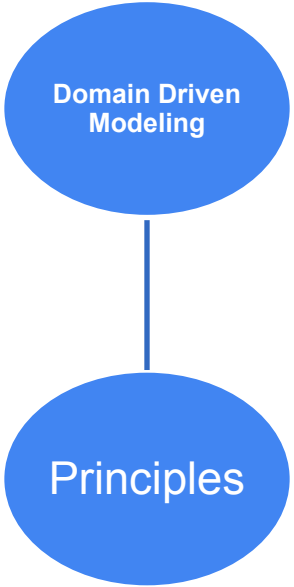
Source: http://martinfowler.com/articles/microservices.html

MICROSERVICES: PRINCIPLES

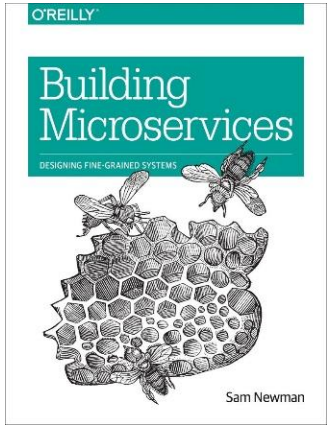
Principles

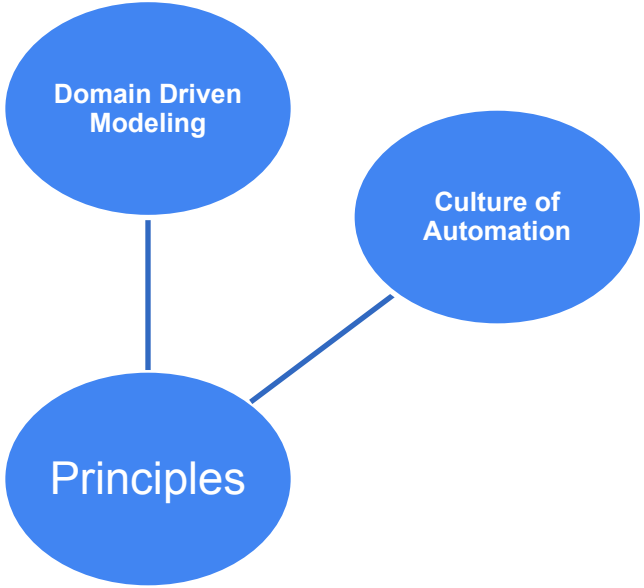


Sam Newman's Principles of Microservices

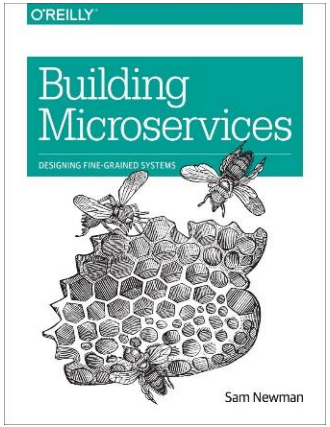


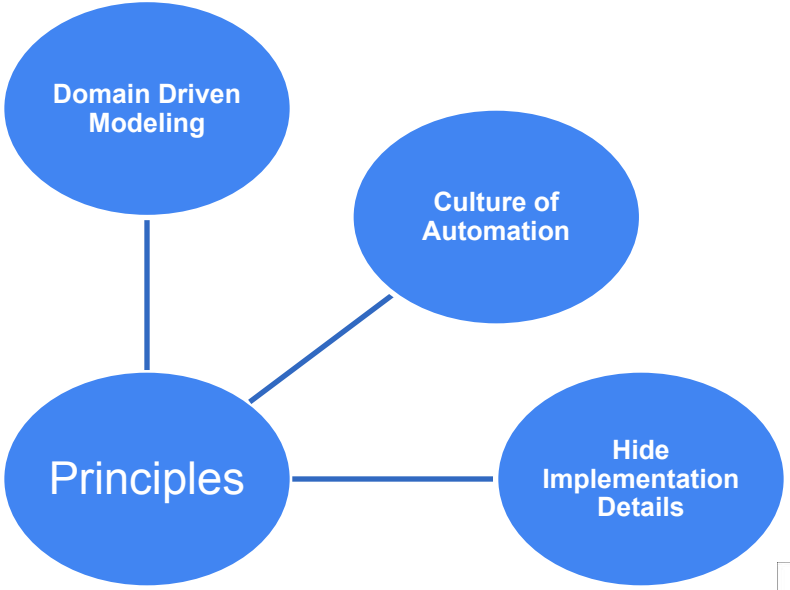
Sam Newman's Principles of Microservices

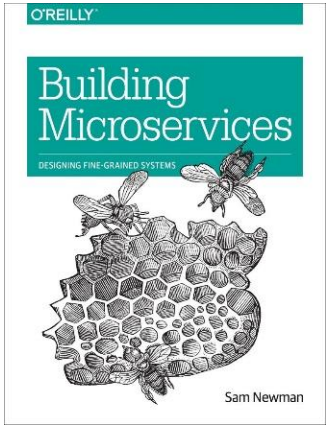




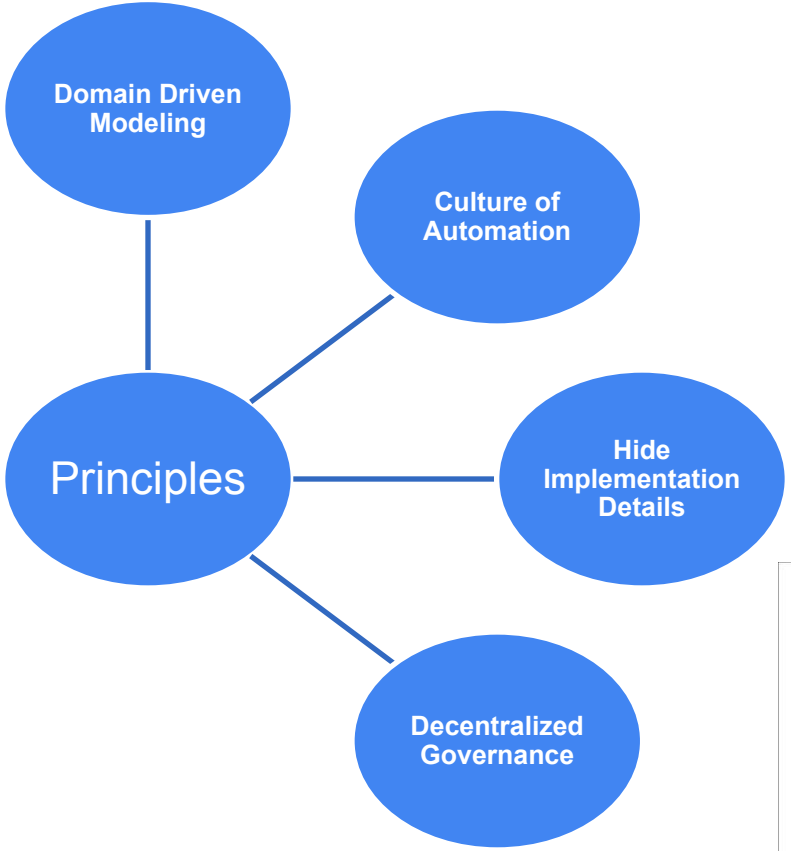
Sam Newman's Principles of Microservices



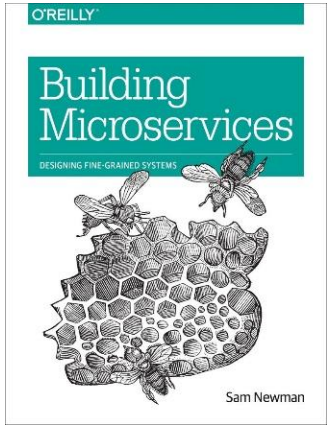


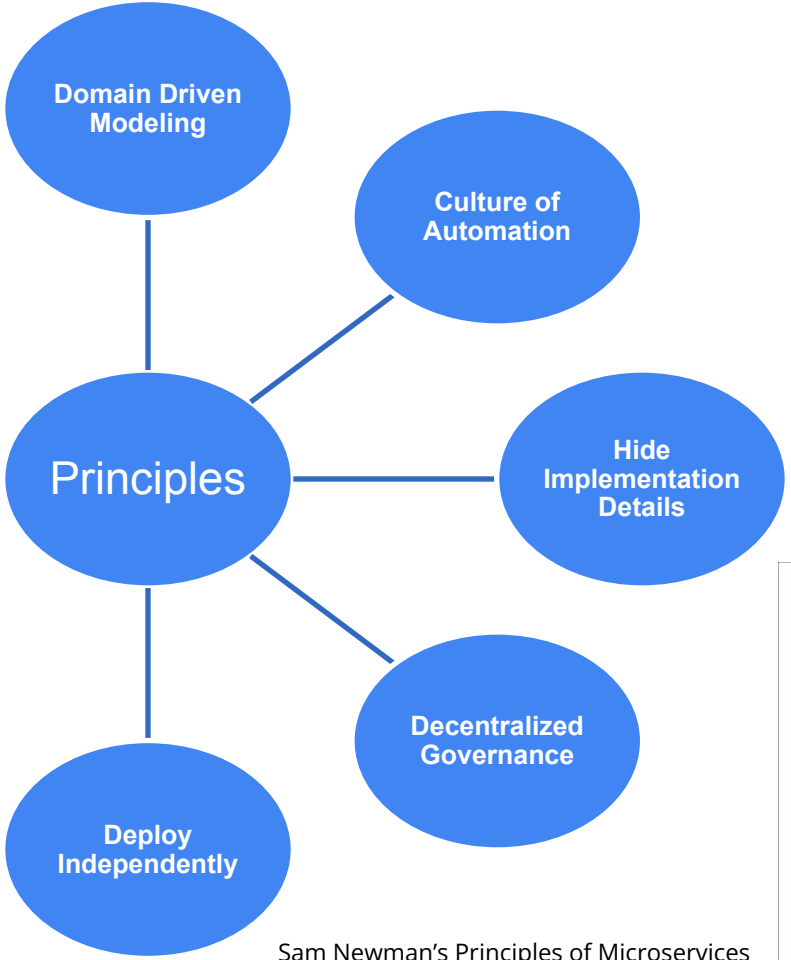


Sam Newman's Principles of Microservices

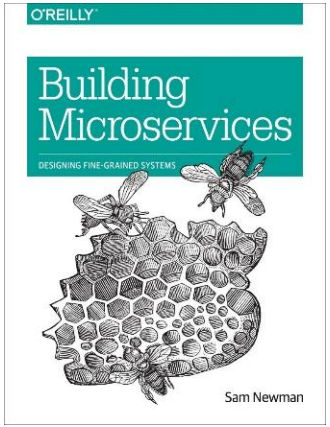


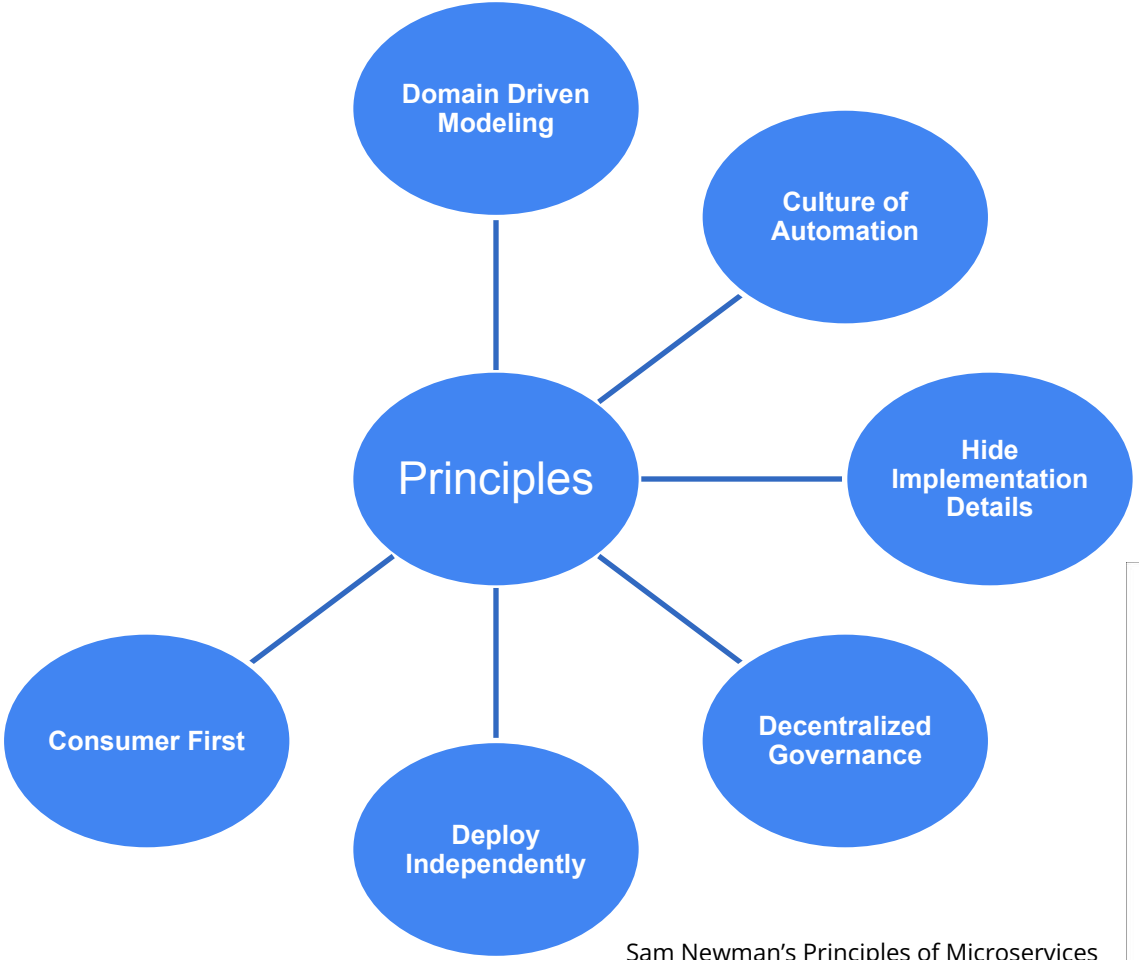
Sam Newman's Principles of Microservices



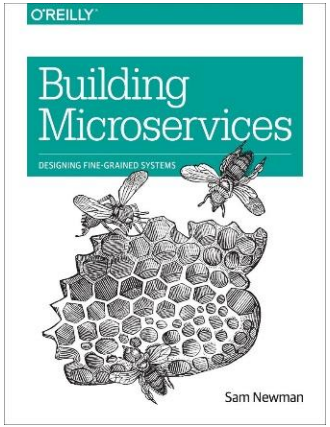


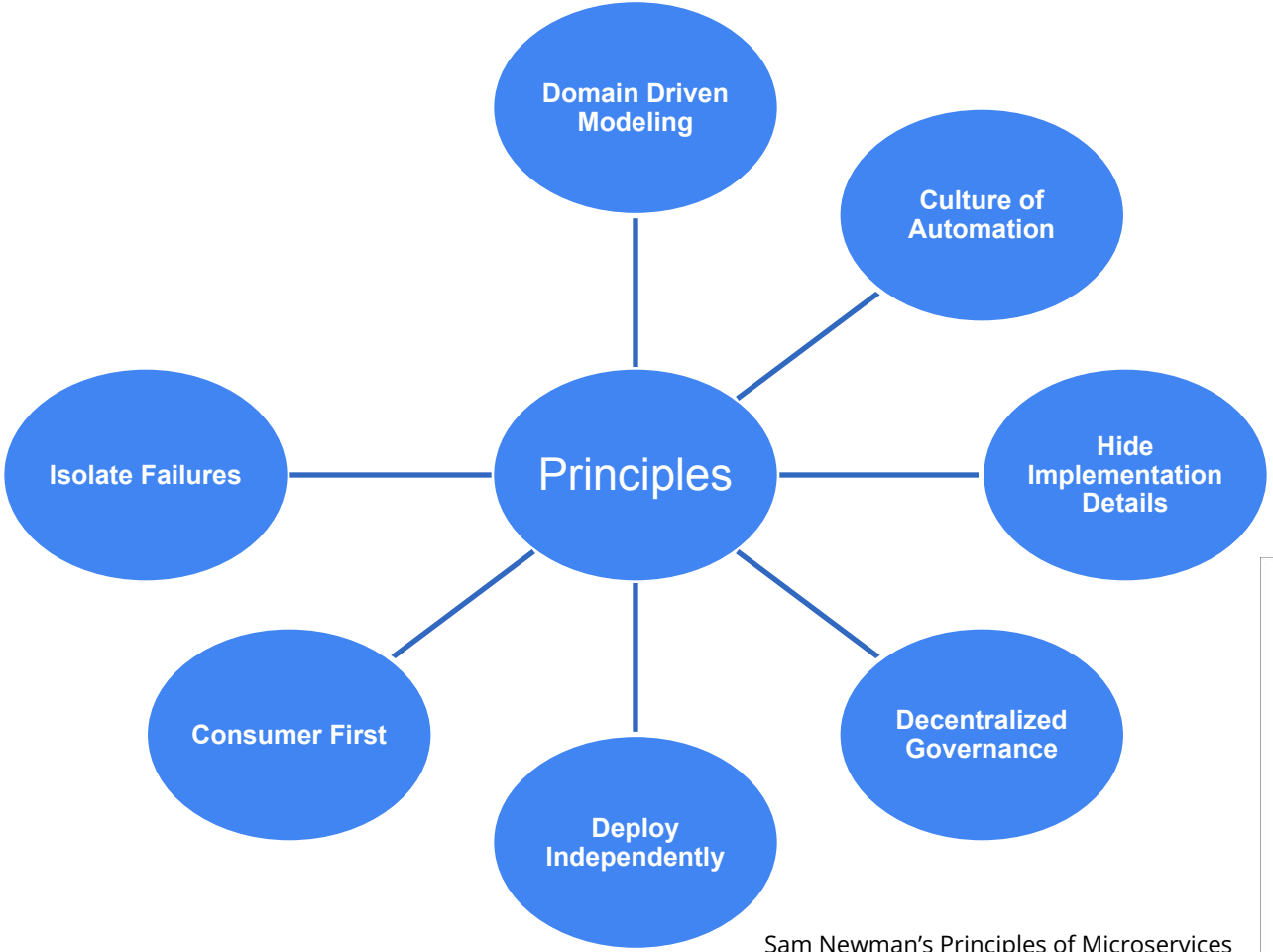
Sam Newman's Principles of Microservices



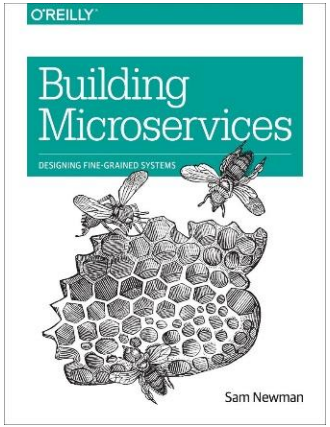


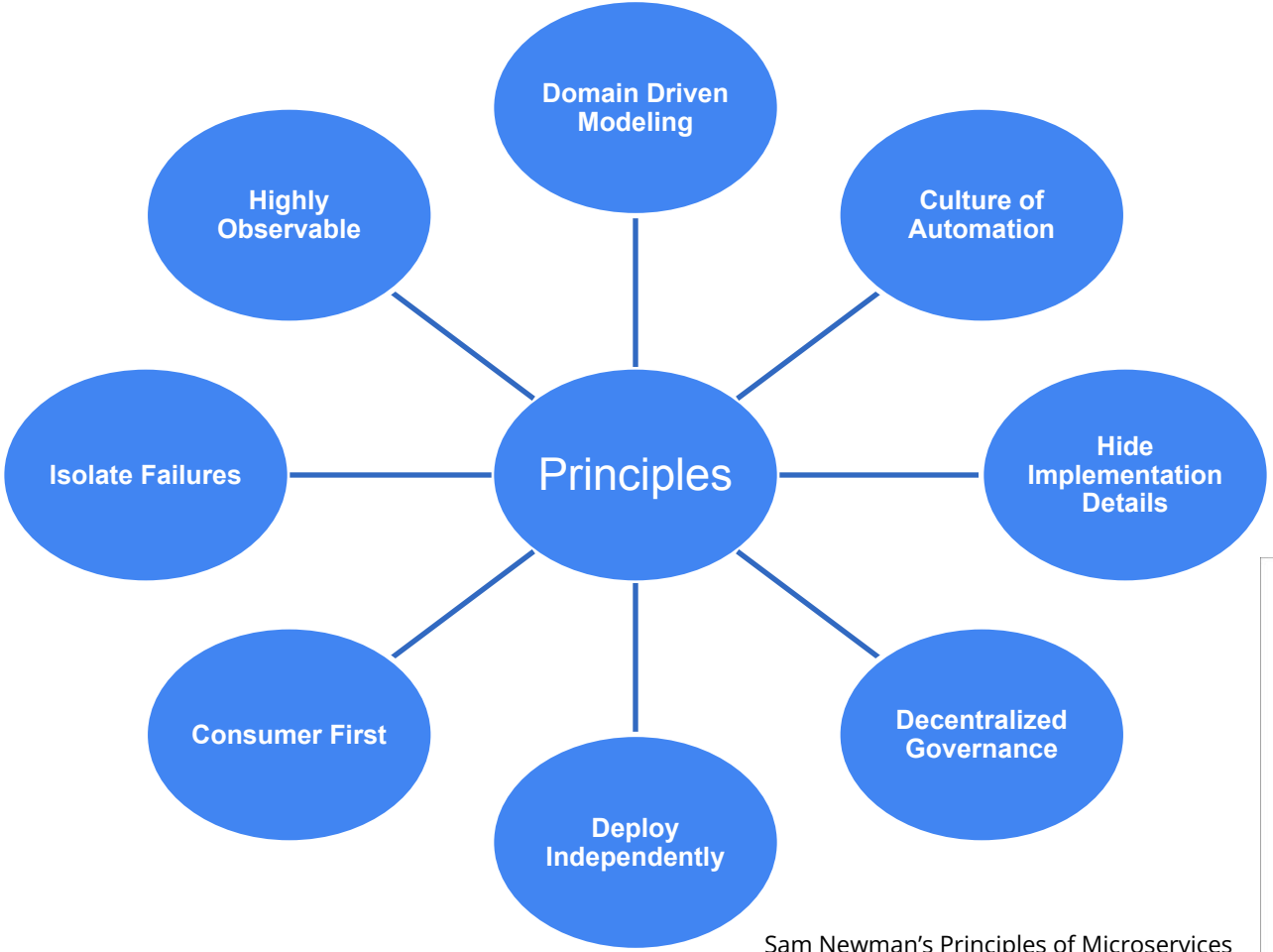
Sam Newman's Principles of Microservices



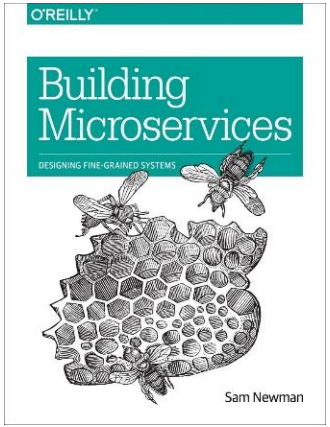


Sam Newman's Principles of Microservices



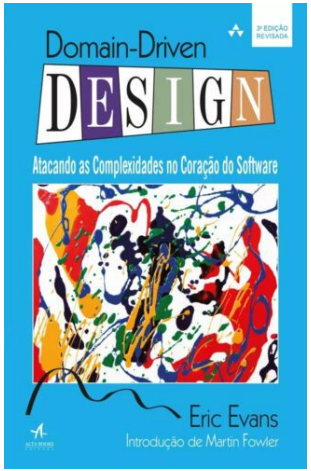


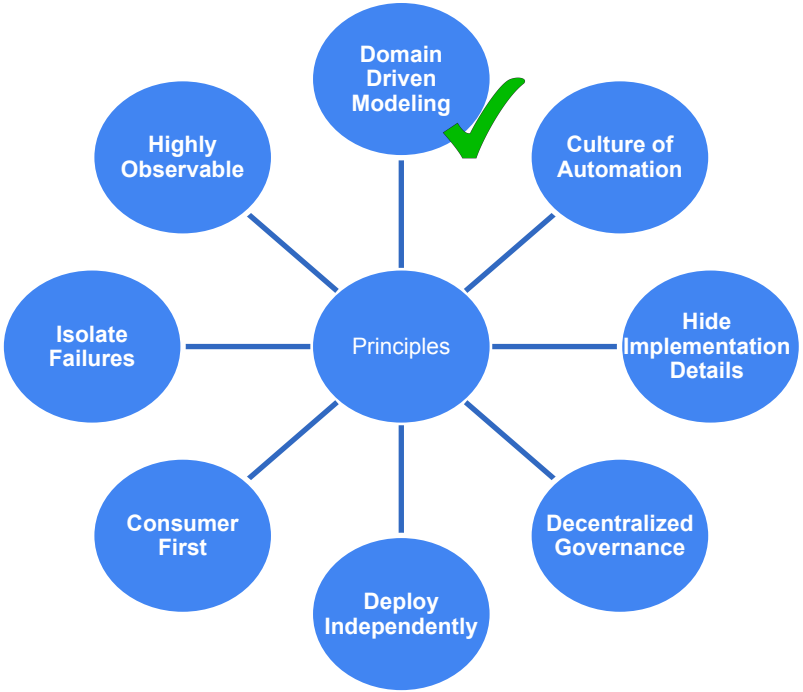
Sam Newman's Principles of Microservices



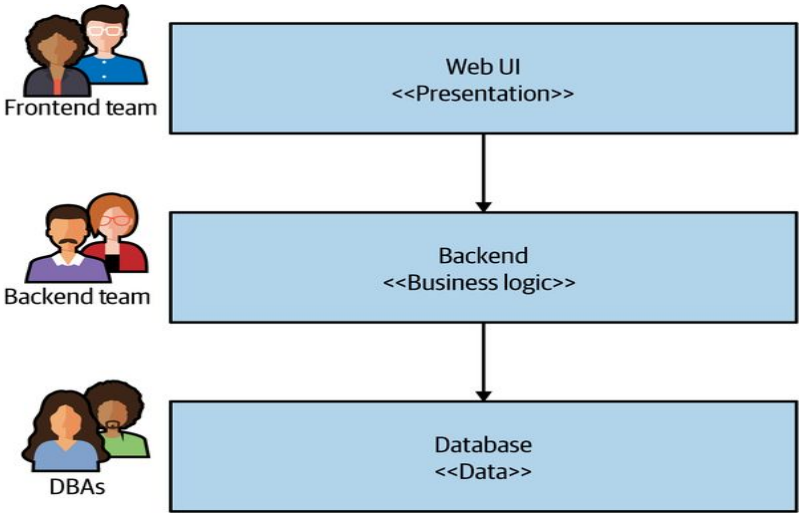
Principle 1: Domain-driven modeling

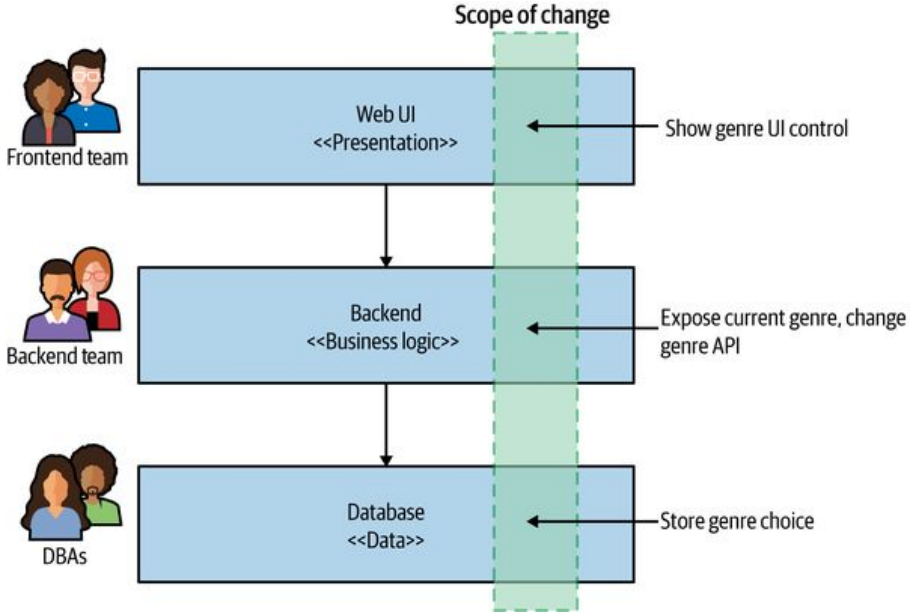
- Model services around business capabilities



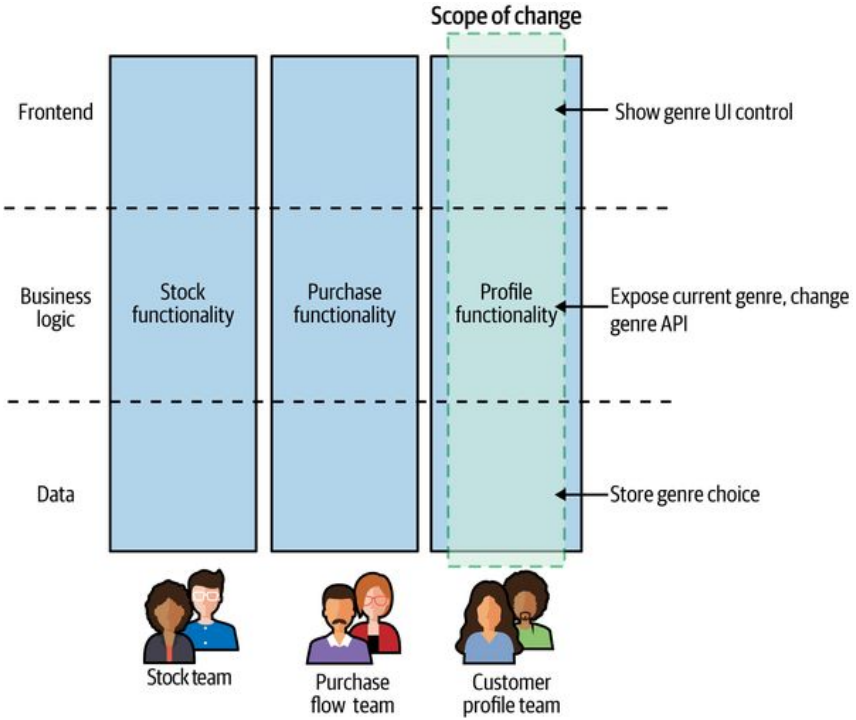


Principle 1: Domain-driven modeling



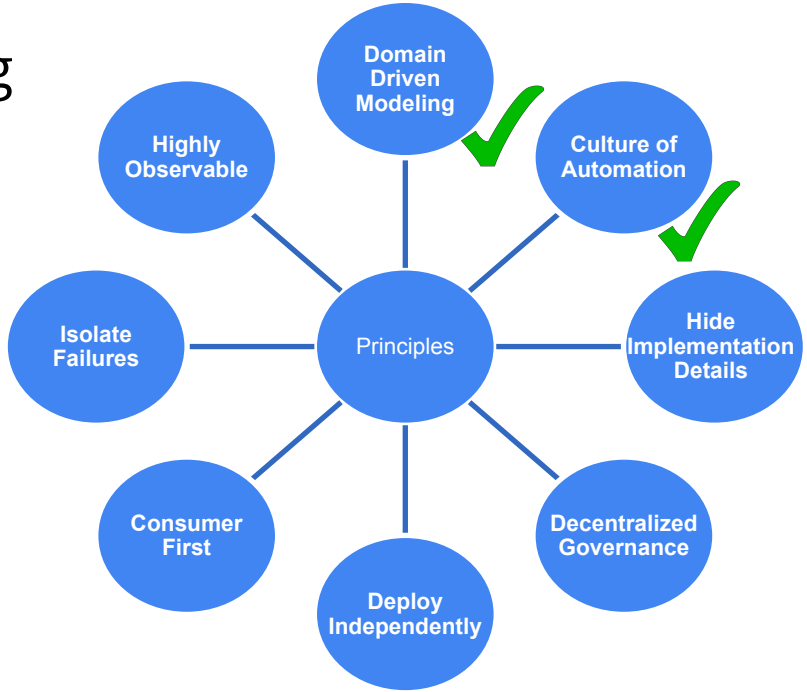


Principle 1: Domain-driven modeling



Principle 2: Culture of Automation

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



API-Driven Machine Provisioning

Example: Infrastructure as code (IaC)

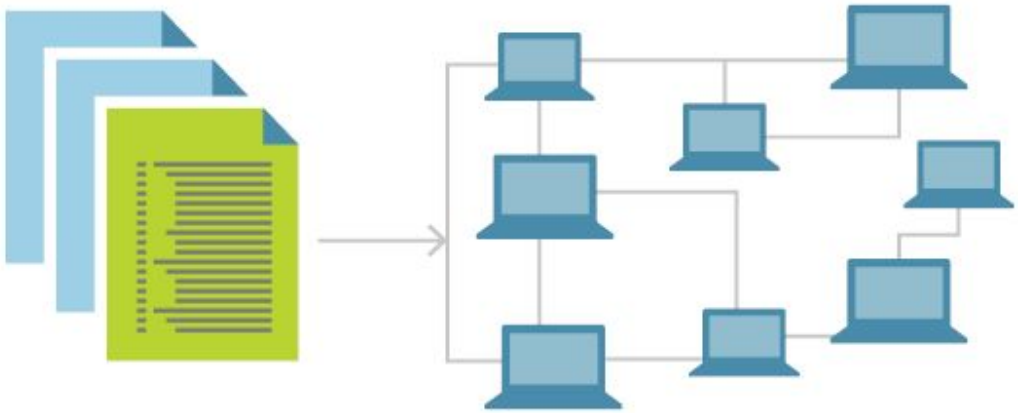
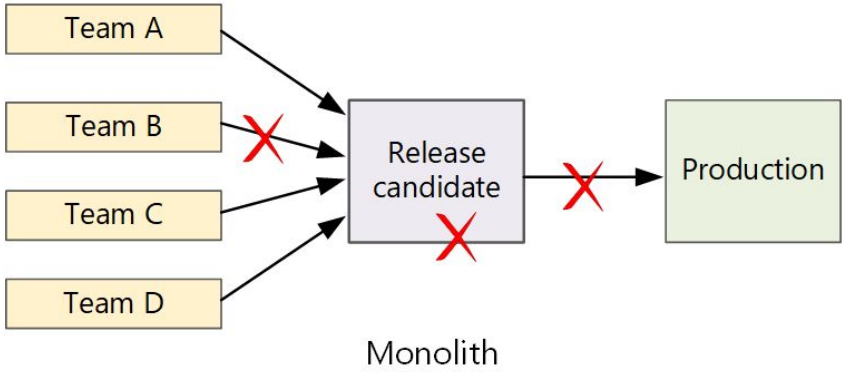


Image source: https://learn.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code

Continuous Delivery

More on this topic later



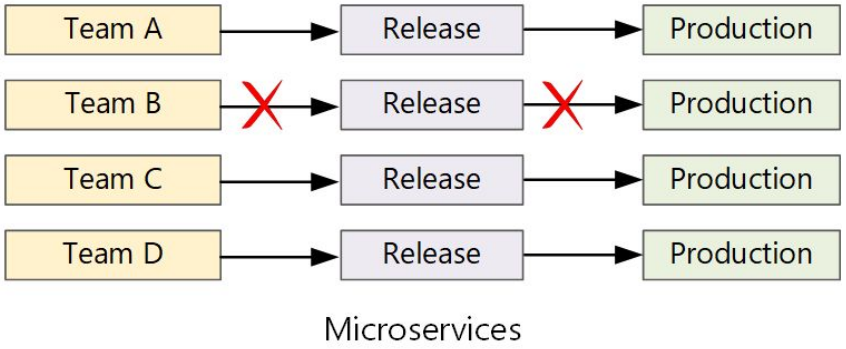
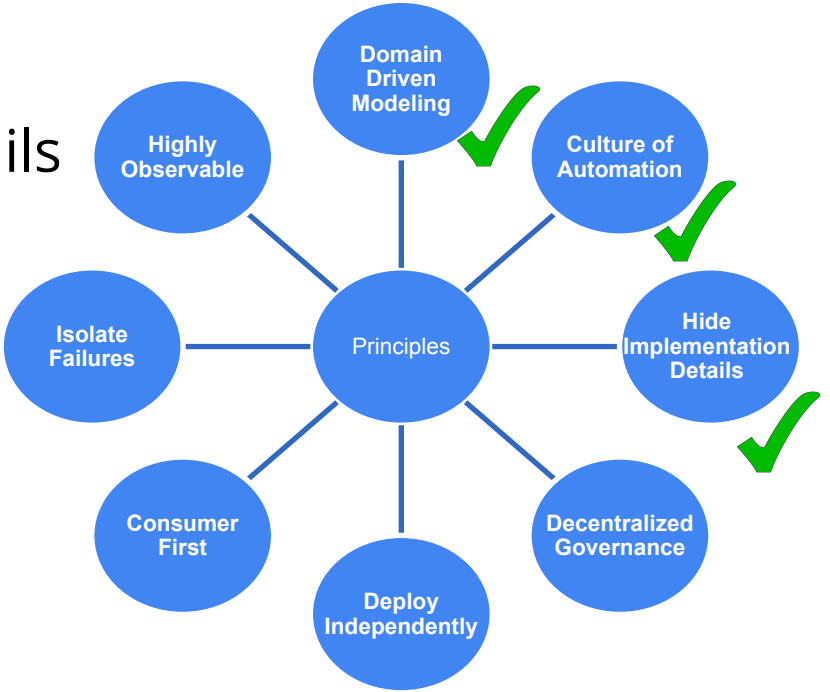


Image Source: https://learn.microsoft.com/en-us/azure/architecture/microservices/ci-cd

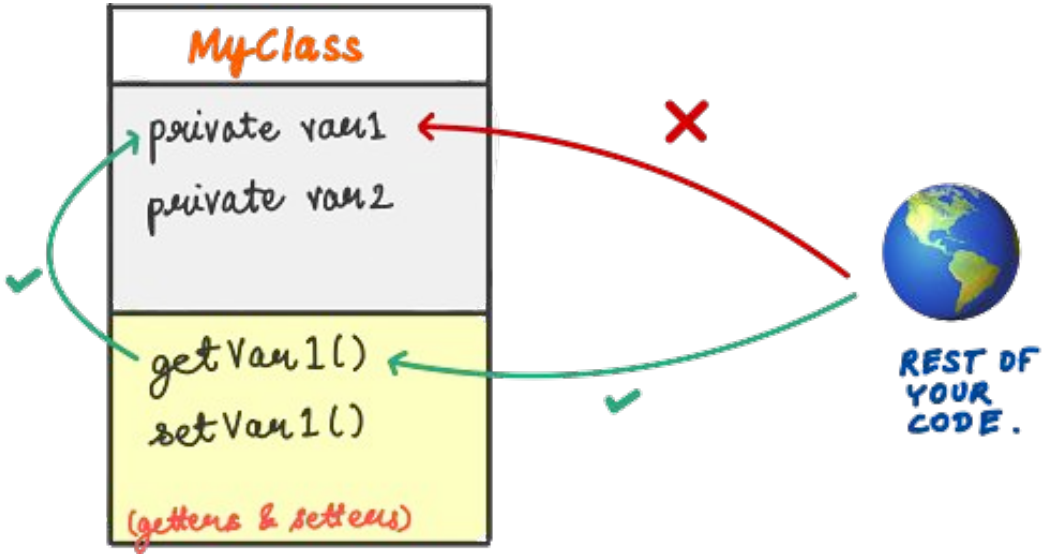
Principle 3: Hide implementation details

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

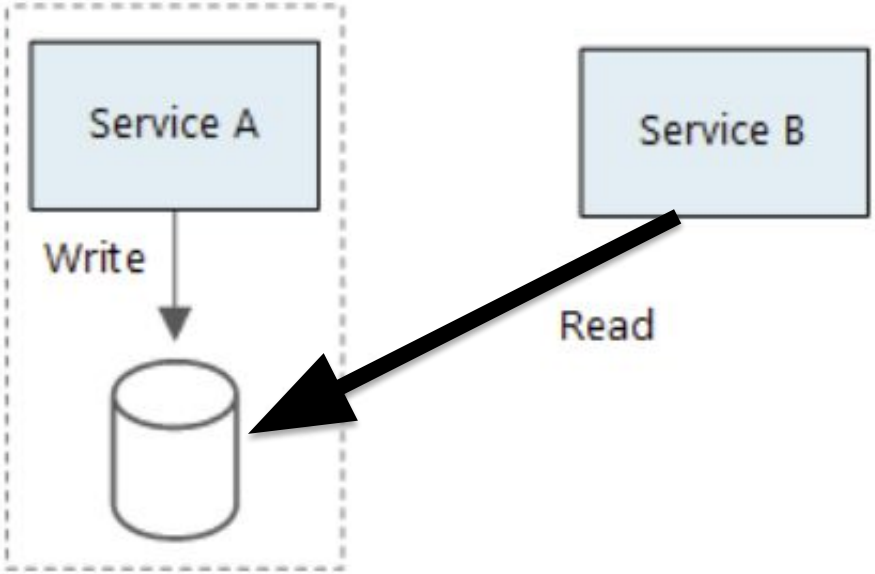


Principle 3: Hide implementation details

Recall: Encapsulation in OOP

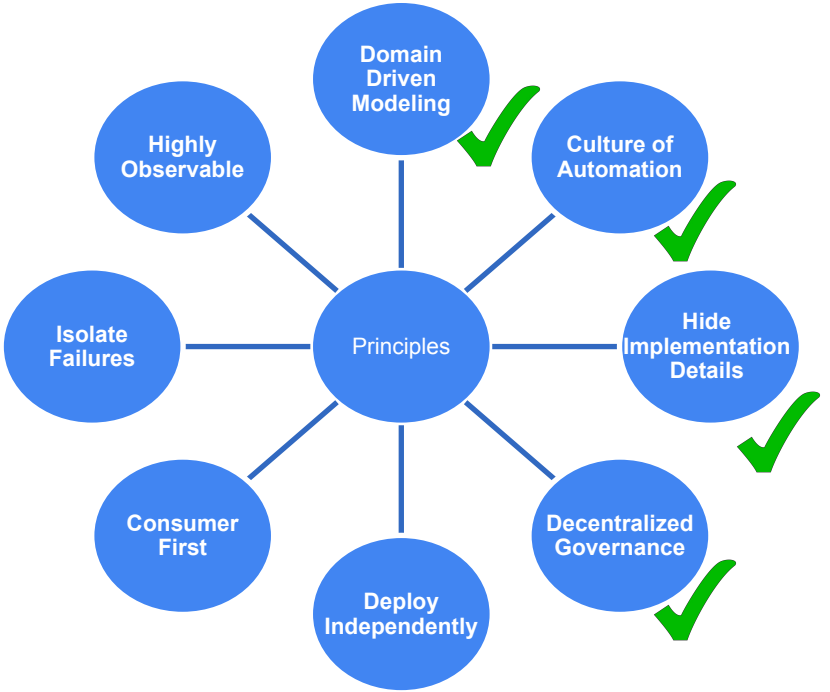


Sharing database: Anti-pattern

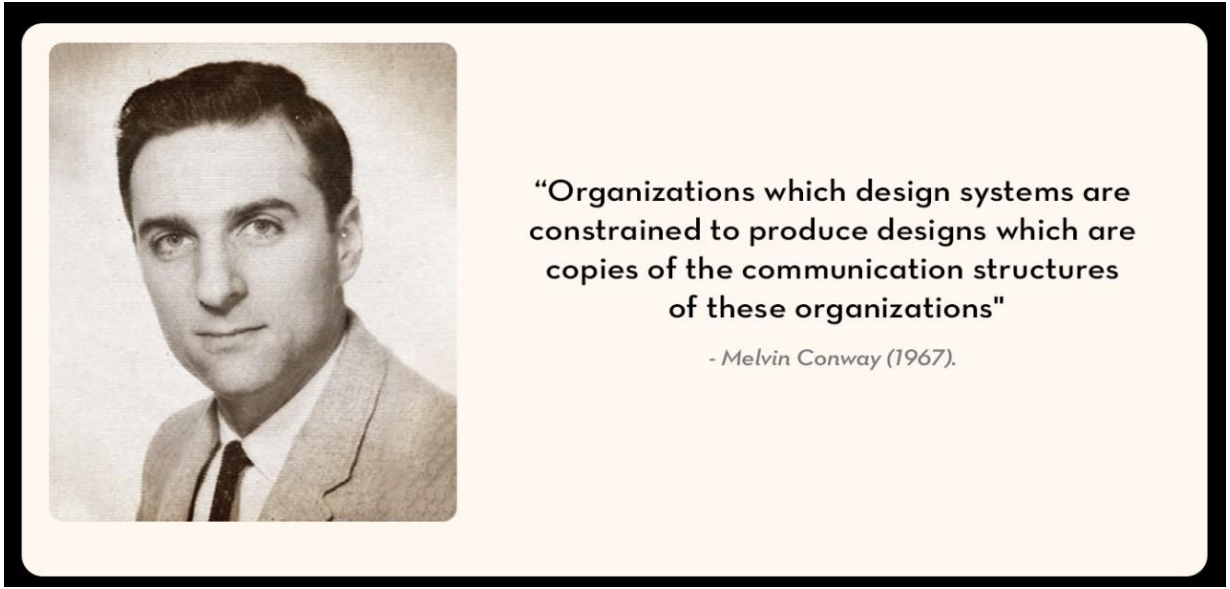


Principle 4: Decentralized Governance

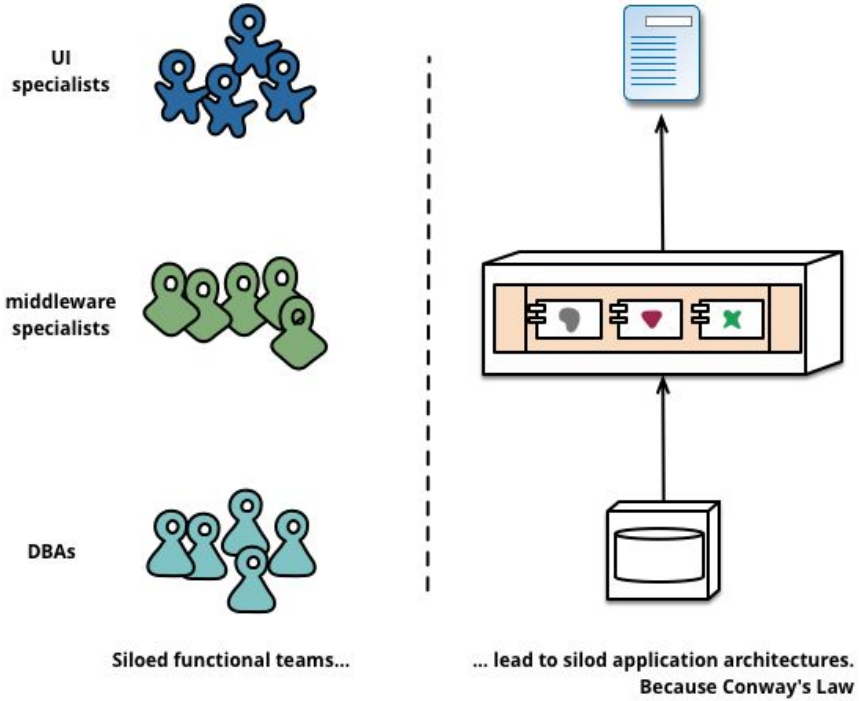
- Mind Conway's Law
- You Build It, You Run It
- Embrace team autonomy
- Internal Open Source Model

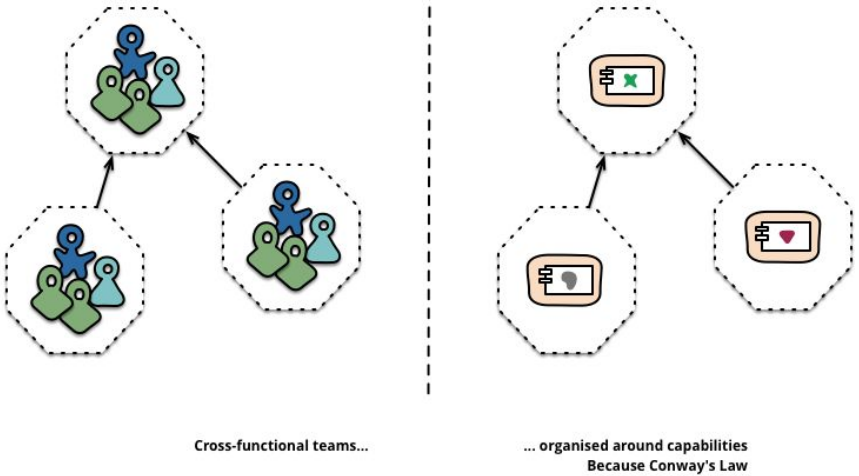


Mind Conway's Law



Mind Conway's Law





"Products" not "Projects"

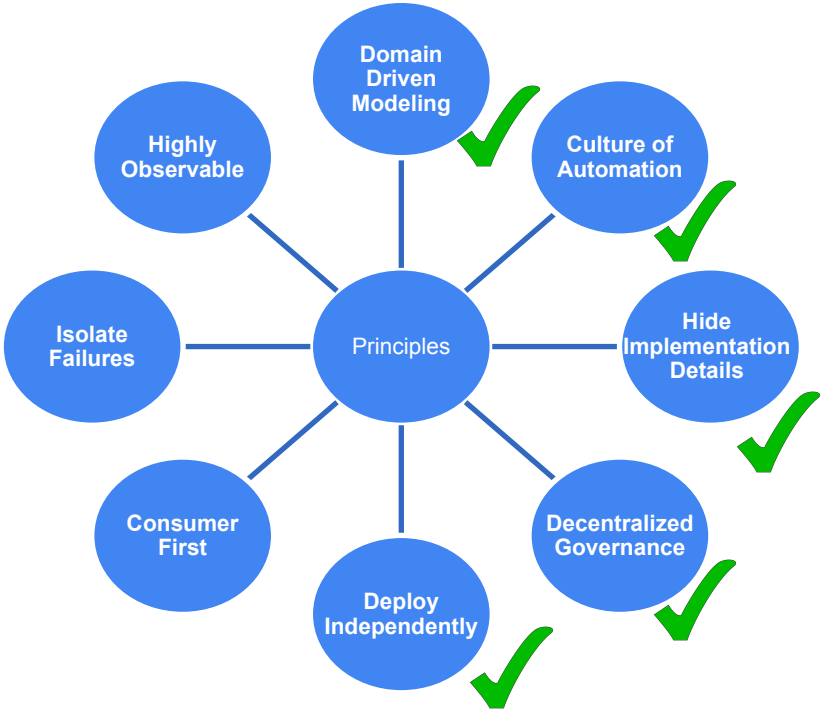
YOU BUILD IT YOU RUN ~~AWAY~~ IT

"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

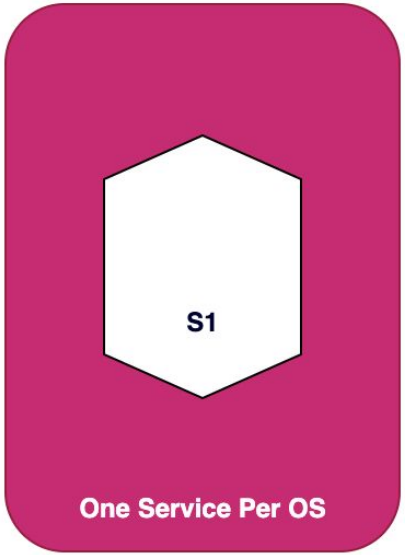
Principle 5: Deploy Independently

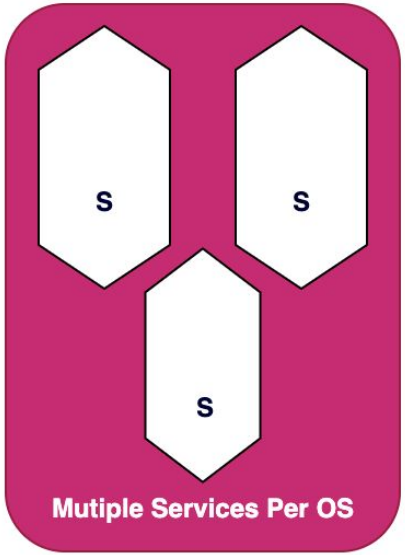
- One Service Per OS
- Consumer-Driven Contracts
- Multiple co-existing versions



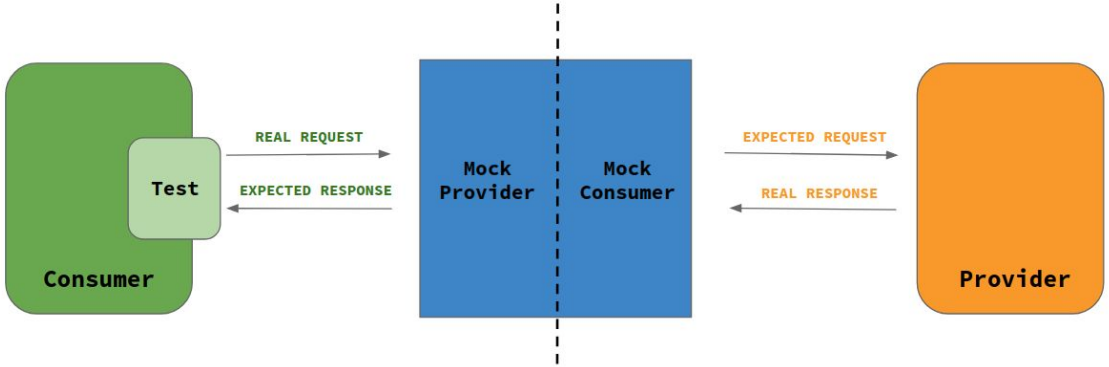
One Service Per OS

Q.
What is the problem with this deployment?

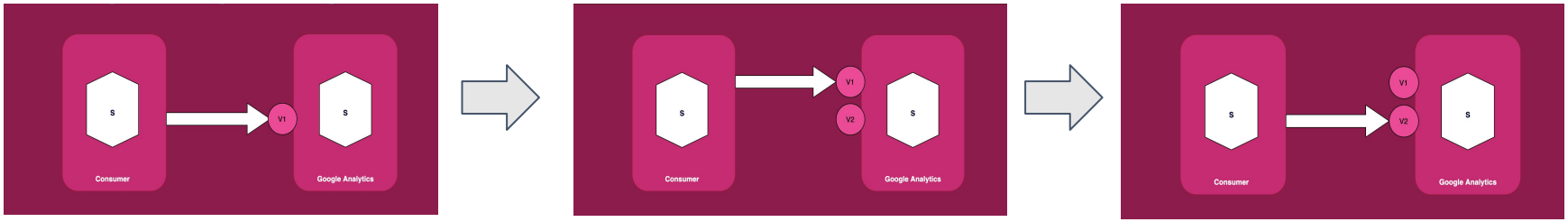




Consumer-Driven Contracts

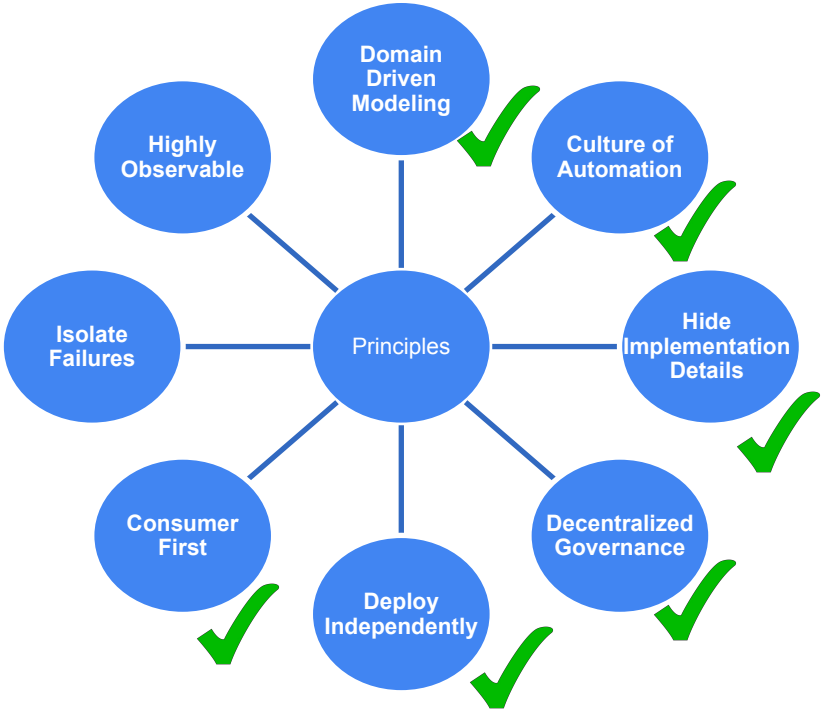


Multiple coexisting versions



Principle 6: Consumer First

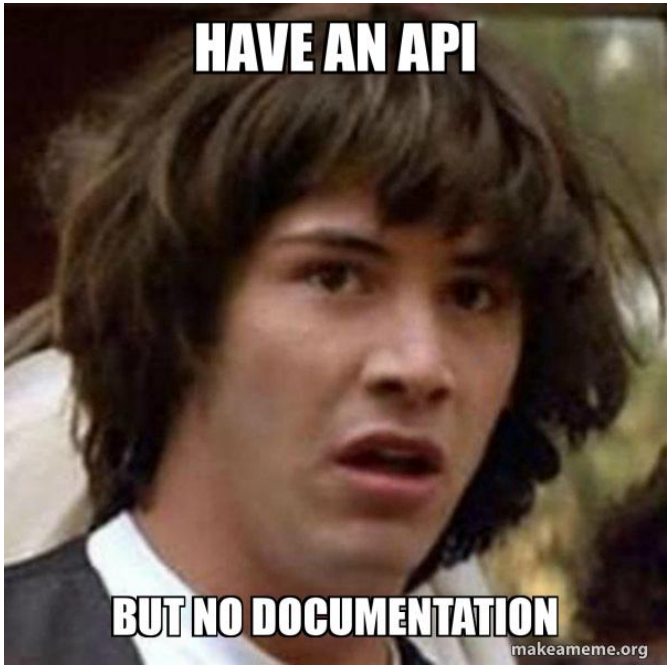
- Encourage conversations
- API Documentation
- Service Discovery



Encourage conversations

vs

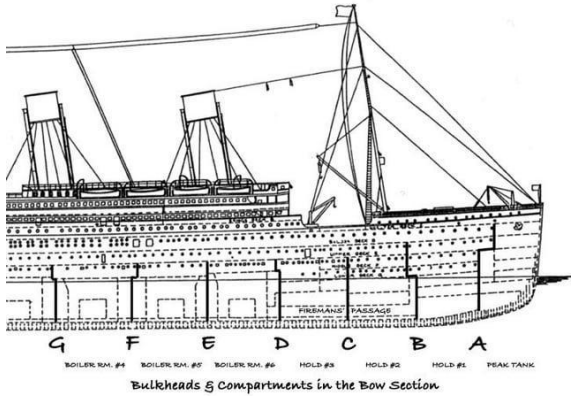
API Documentation



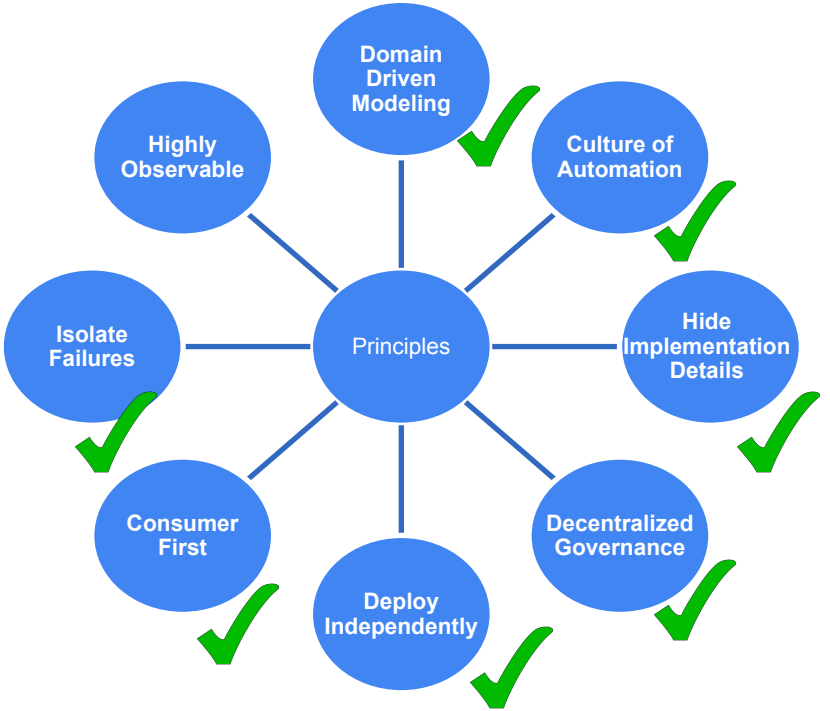
Swagger™

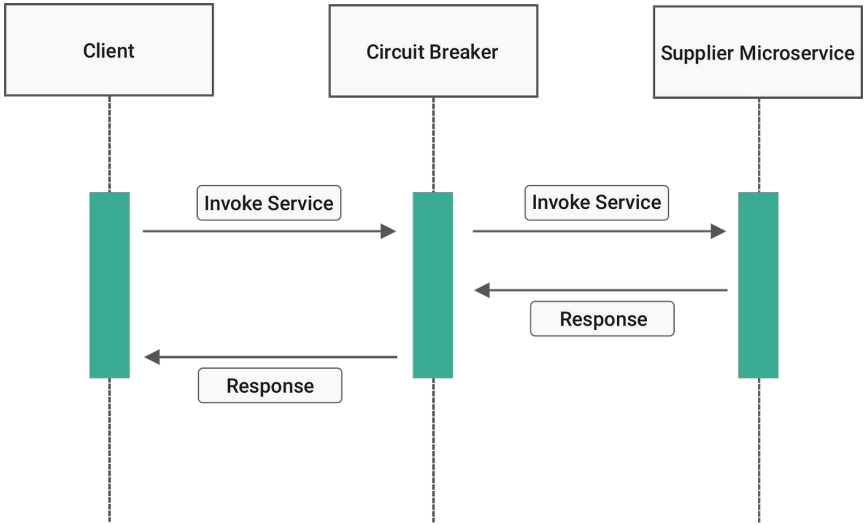
Principle 7: Isolate Failure

- Avoid cascading failures
- Timeouts between components
- **Fail fast** aka *Design for Failure*
  - Bulkheading / Circuit breakers

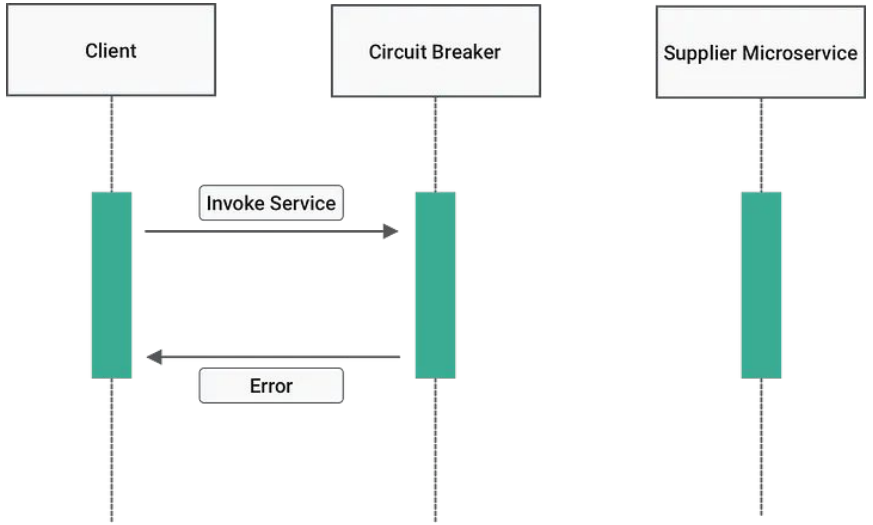


Bulkheads & Compartments in the Bow Section





Closed circuit

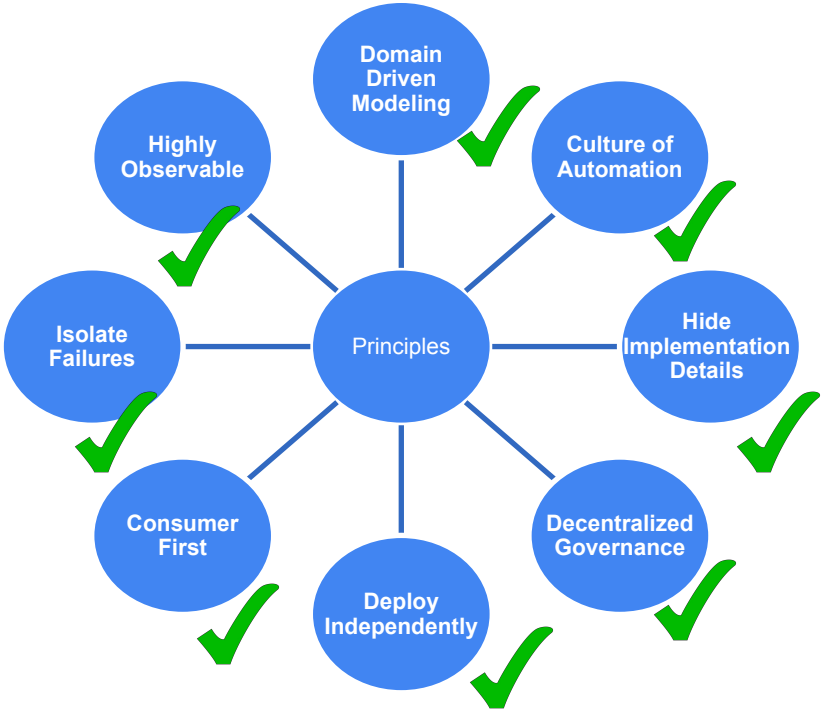


Open circuit

Image source: blogs.halodoc.io

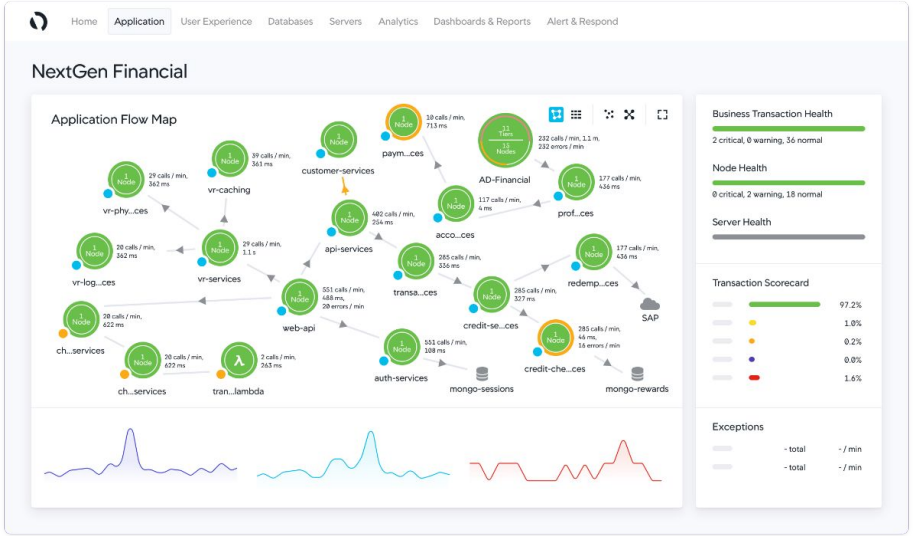
Principle 8: Highly Observable

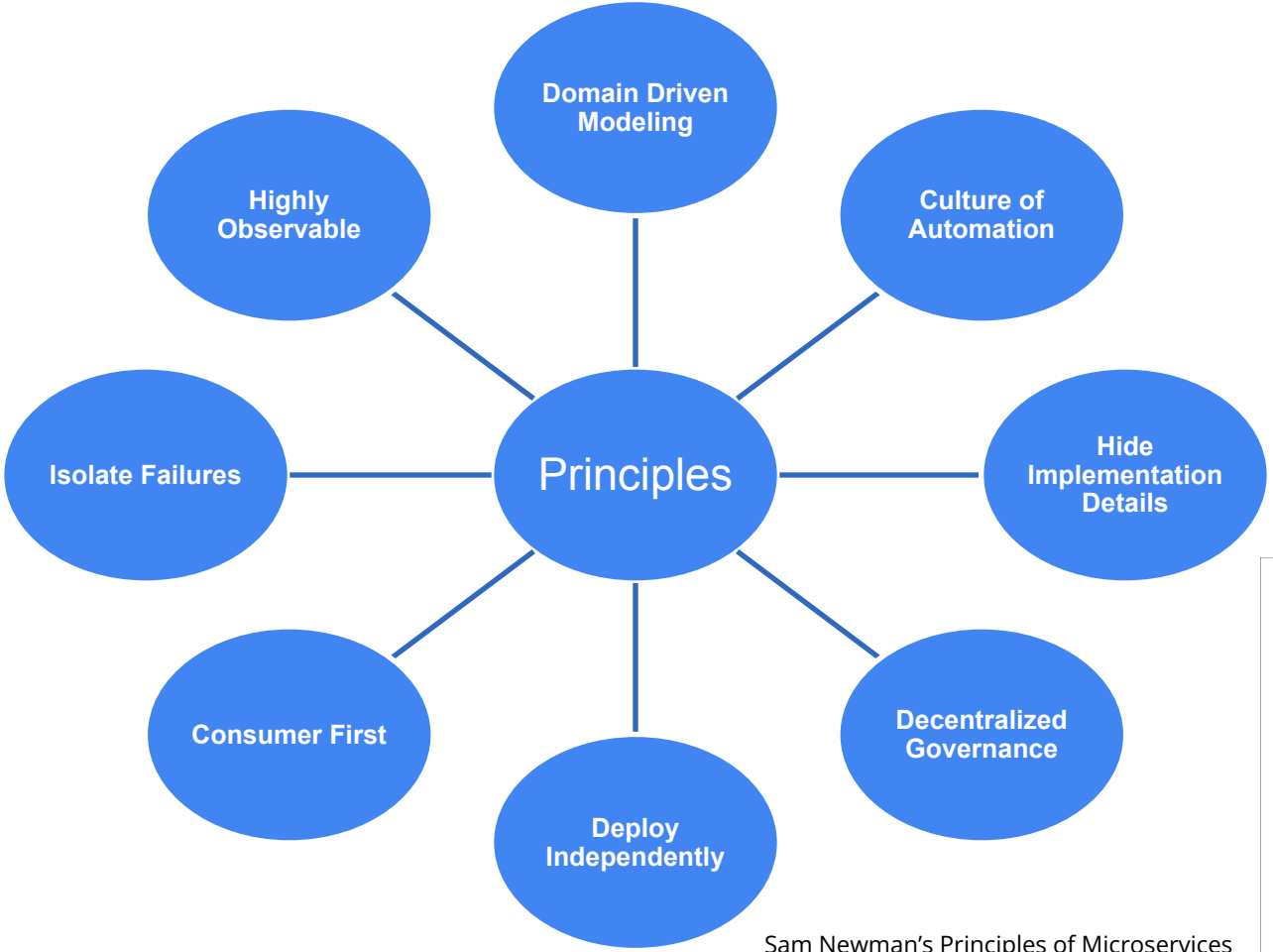
- Standard Monitoring
- Health-Check Pages
- Log and Stats aggregation
- Downstream monitoring



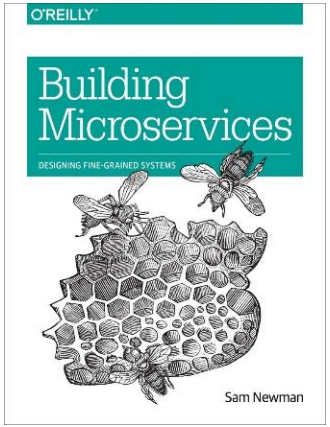
Principle 8: Highly Observable

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





Sam Newman's Principles of Microservices



Are microservices always the right choice?

Summary: 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
- More points of failure
- Operational complexity
- Frequently adopted by breaking down a monolithic application

Summary: Advantages of Microservices

- 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

Microservices overhead