# How to scale your Applications with **Micro-Frontends**

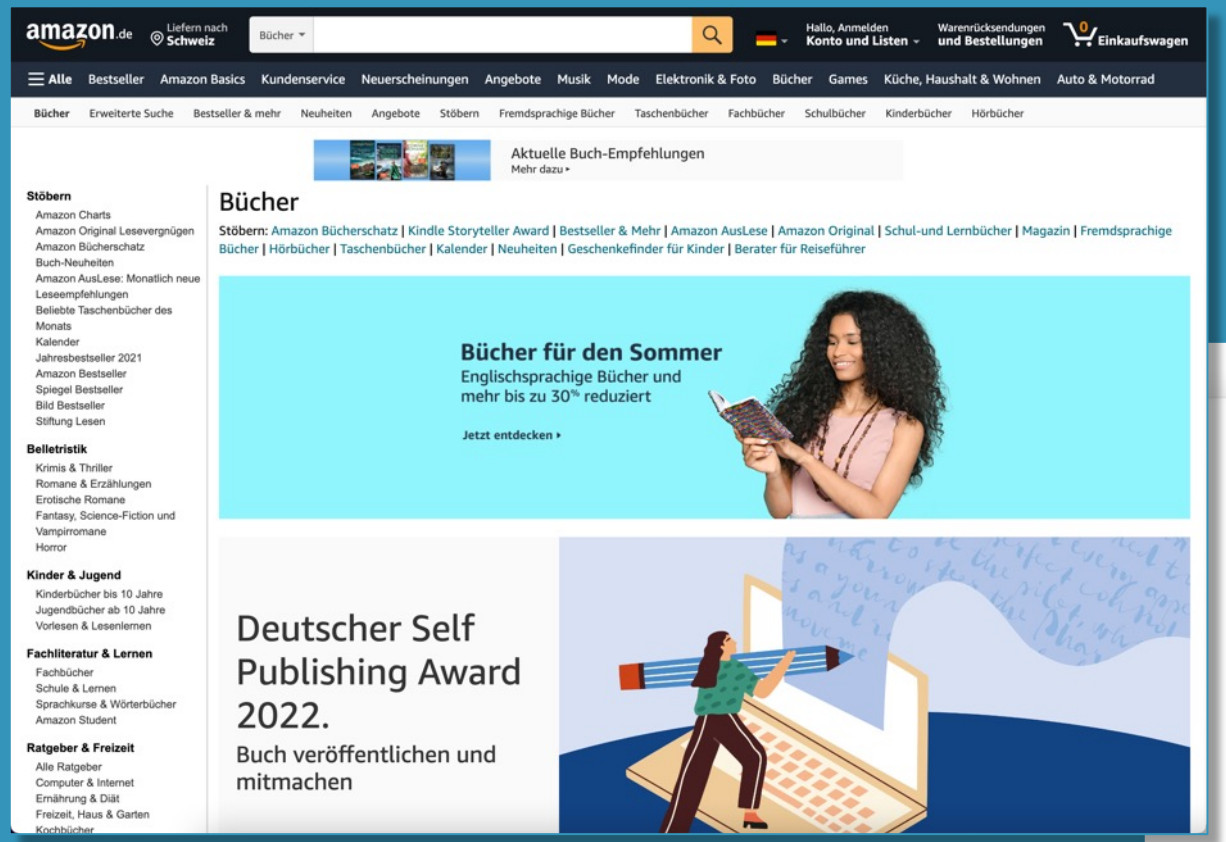
*Java Forum Stuttgart 2022*

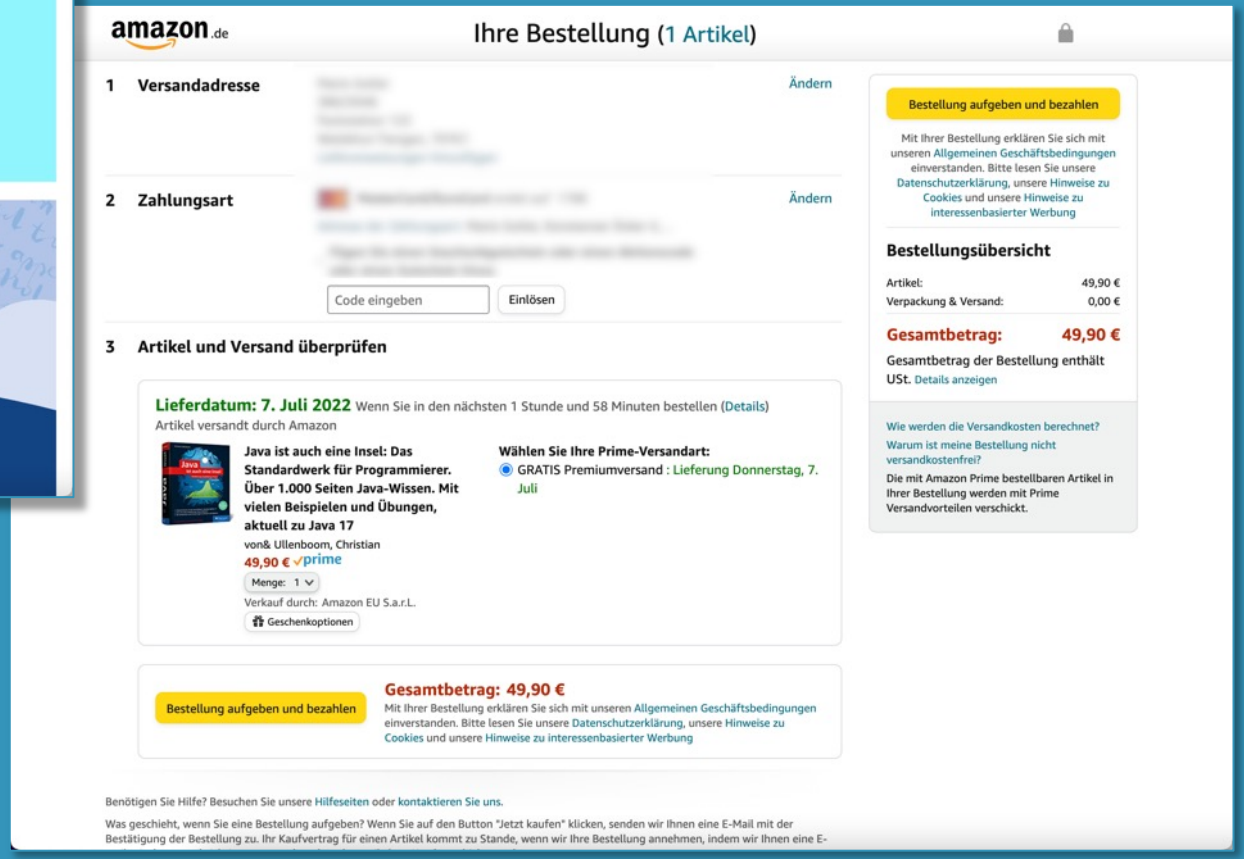
**Mario Goller**
*Swisscom AG*

swisscom

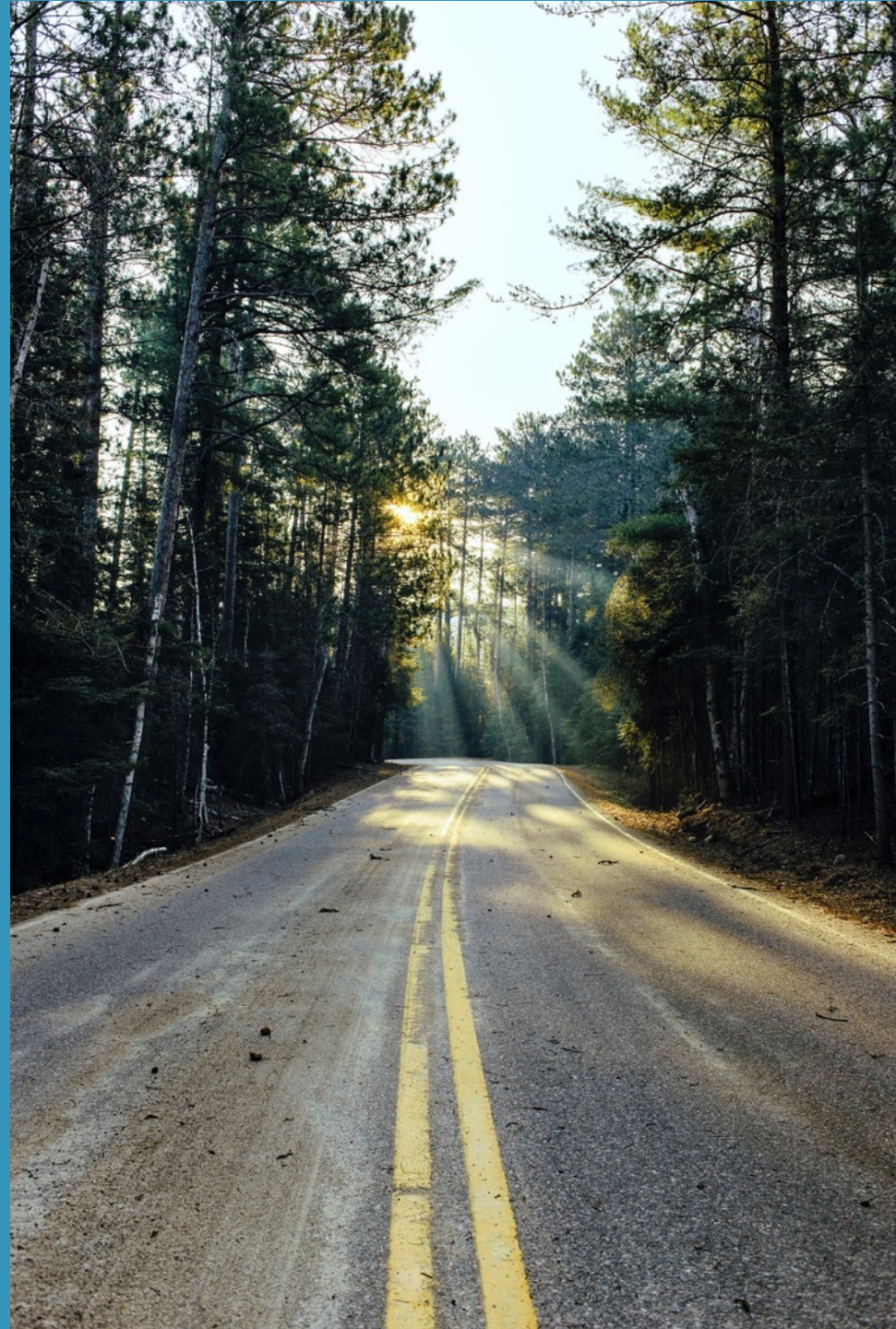
Hintergrund Foto erstellt von andreas - de.freepik.com

# Example:
# Amazon websites

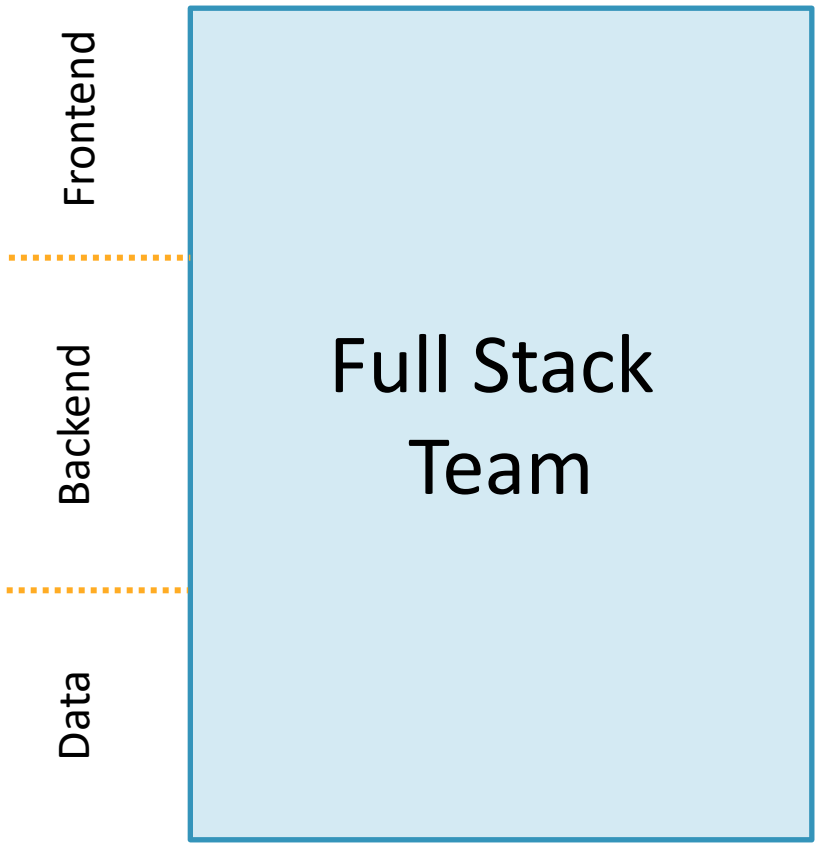


# The Way to Micro-Frontends



# *The Monolith*




# *Frontend & Backend*

Frontend

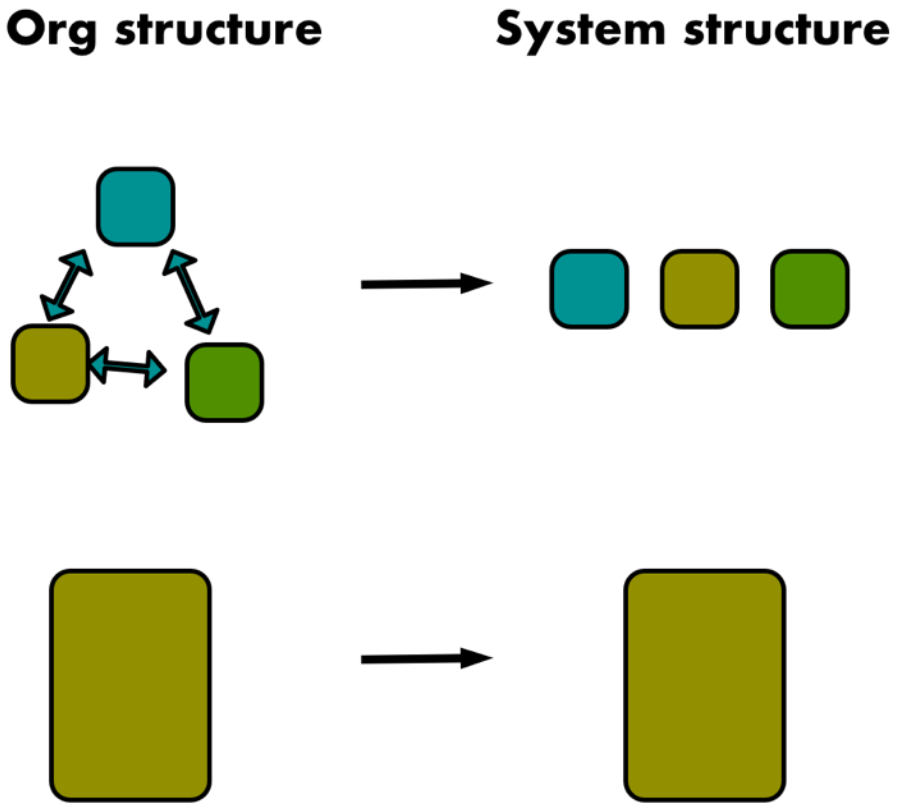
Frontend Team

Backend

Backend /
Devops Team

Data

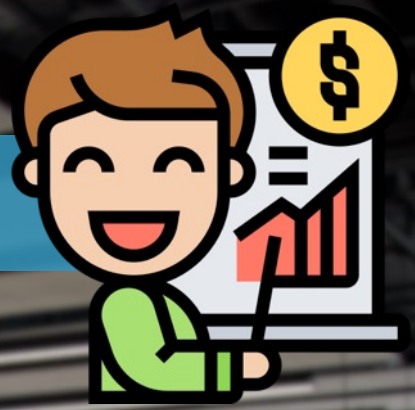
# Conway's Law




*Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.*

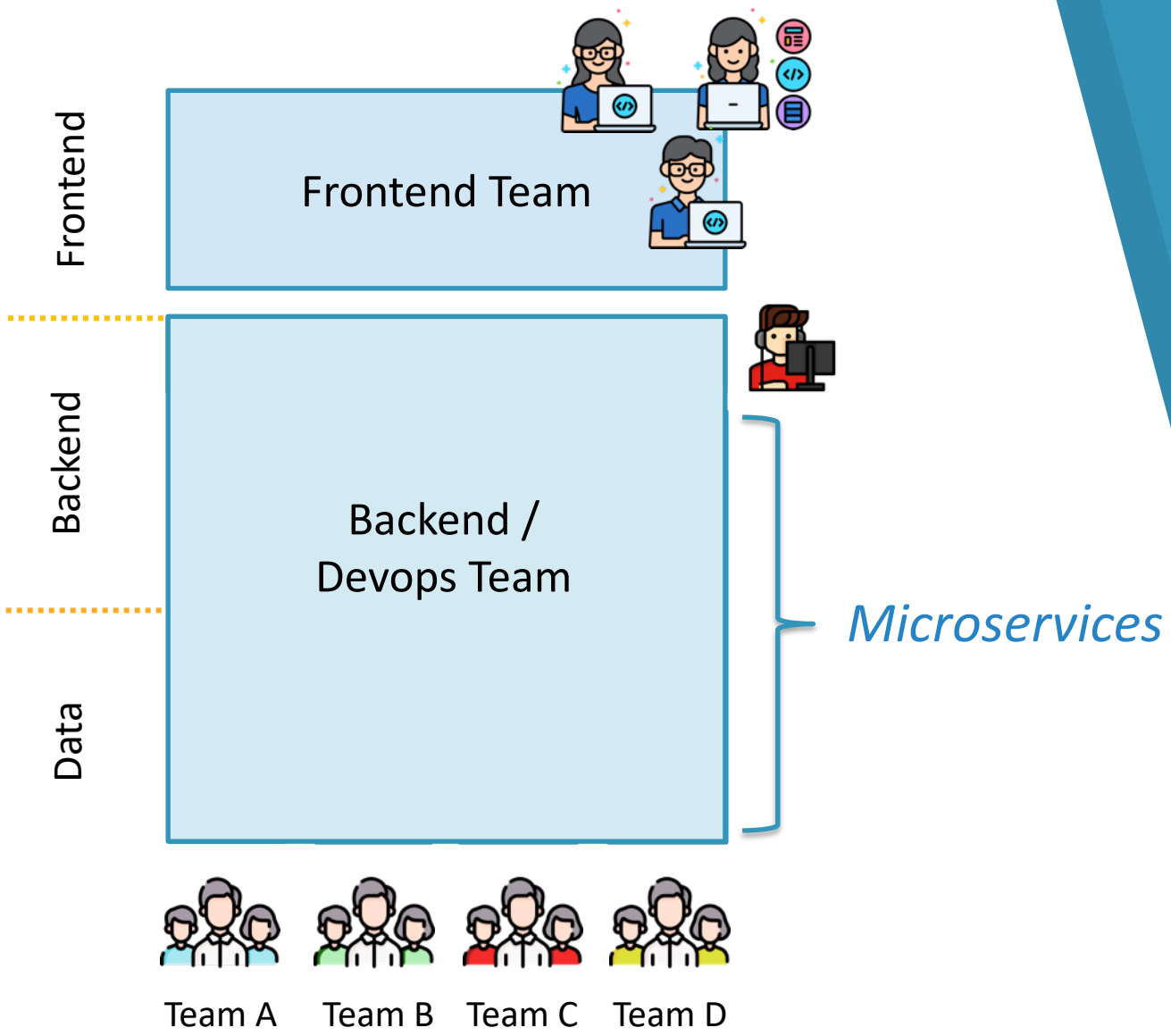
*— M. Conway*

# We need to deliver more and faster!



# Architecture with Microservices

No E2E responsibility!



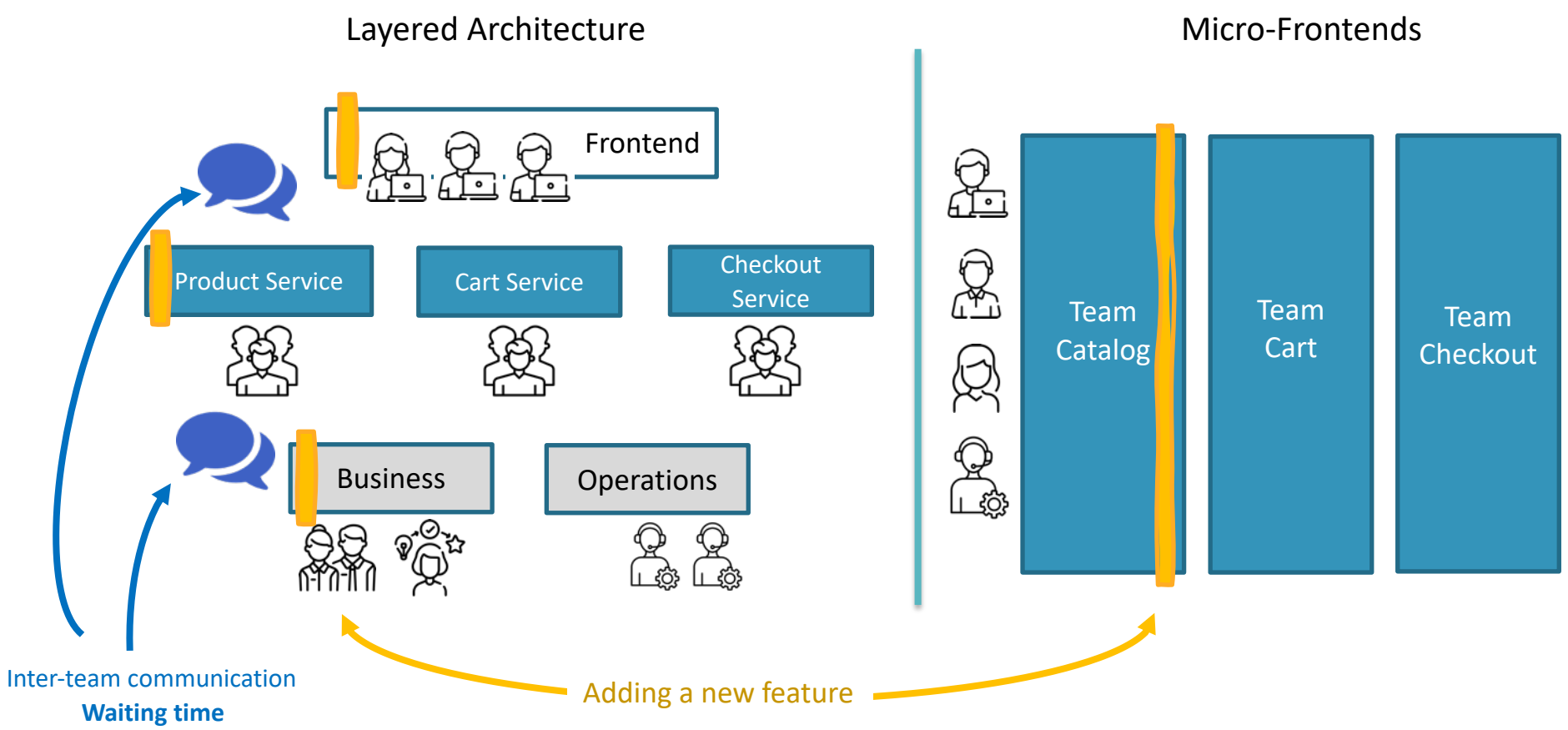
# Our daily Fight @Frontend

- **Scaling Issues**
  - Frontend Application = Frontend team
- **Code and Testing Complexity**
  - Increased Risk for deployments
  - Slows continuous delivery
- **Communication Issues**
  - Communication overhead for managing different parts of the UI

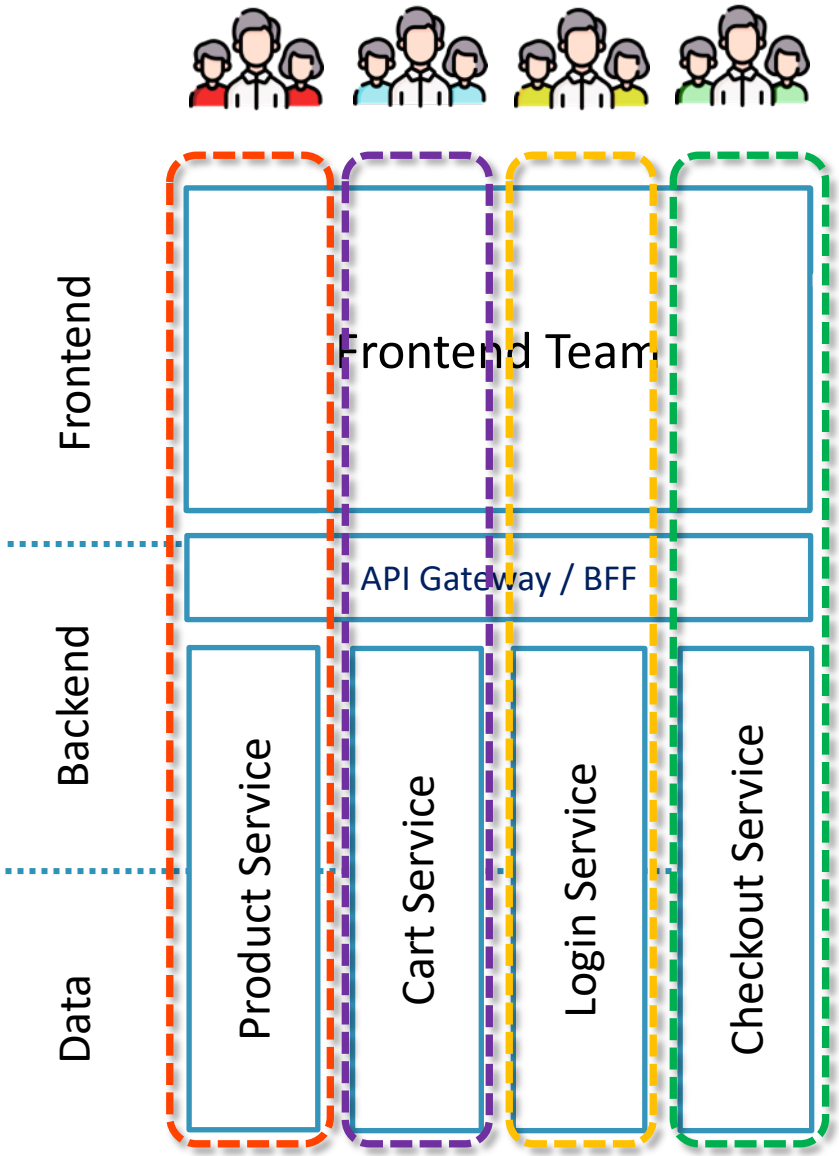
# Technical Teams

VS.

# Product Teams



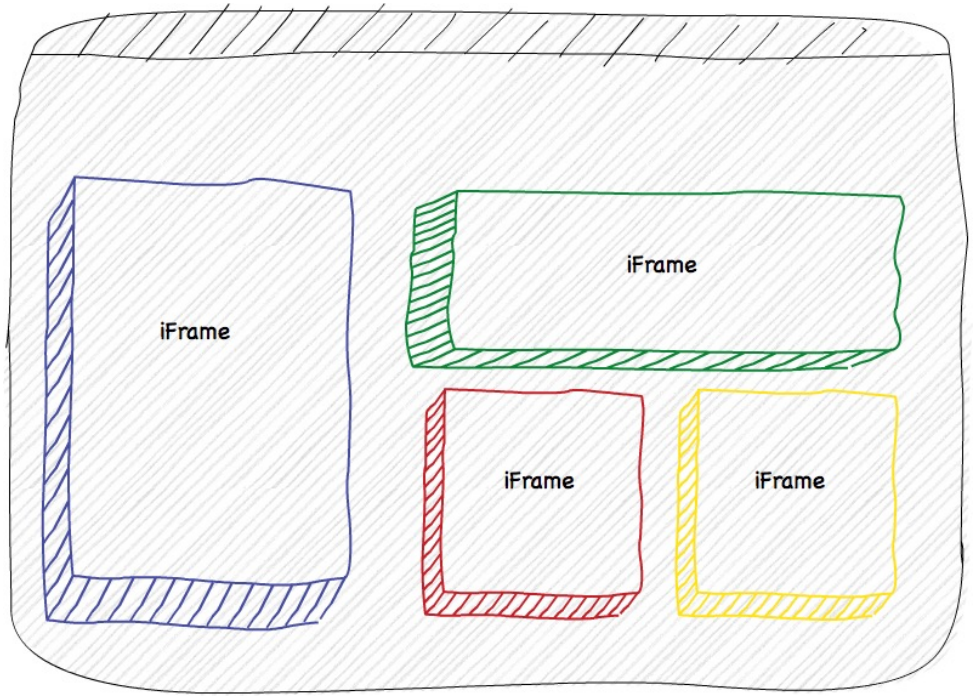
# Microservices + Micro Frontends




**End to End Stream-aligned Teams**


TELL ME MORE ABOUT THIS
"MICRO-THING"
imgflip.com

# Micro-Frontends are quite old in the Browser




"Self-Contained Systems"

"Vertically decomposed applications"

"Microservice websites"

# What is a Micro-Frontend?

" Micro-Frontends are the technical representation of a business subdomain, they allow independent implementations with the same or different technology.

Finaly, they should minimize the code shared with other subdomains and they are owned by a single team. "

*Luca Mezzalira (AWS)*

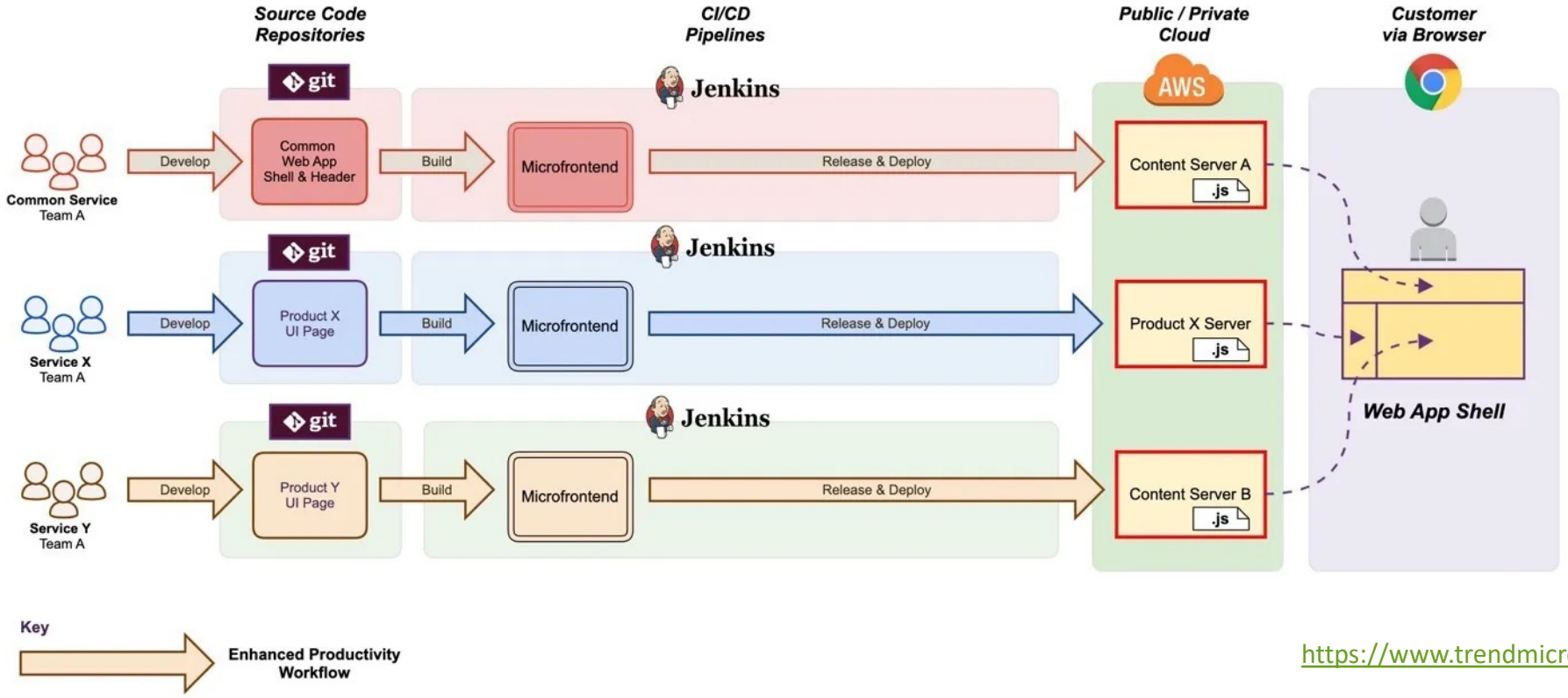
"An architectural style where independently deliverable frontend applications are composed into a greater whole"

*https://martinfowler.com*

# Key Benefits of Micro-Frontends

- ✓ Better scalability
- ✓ Codebases are smaller and more manageable
- ✓ Team ownership
- ✓ Faster development
- ✓ Deployment independence
- ✓ Upgrade, update, or even rewrite parts of the frontend more smoothly
- ✓ Isolate failure - Easier to ensure that rest of the app remains stable
- ✓ Easier testing
- ✓ (Be Technology Agnostic)

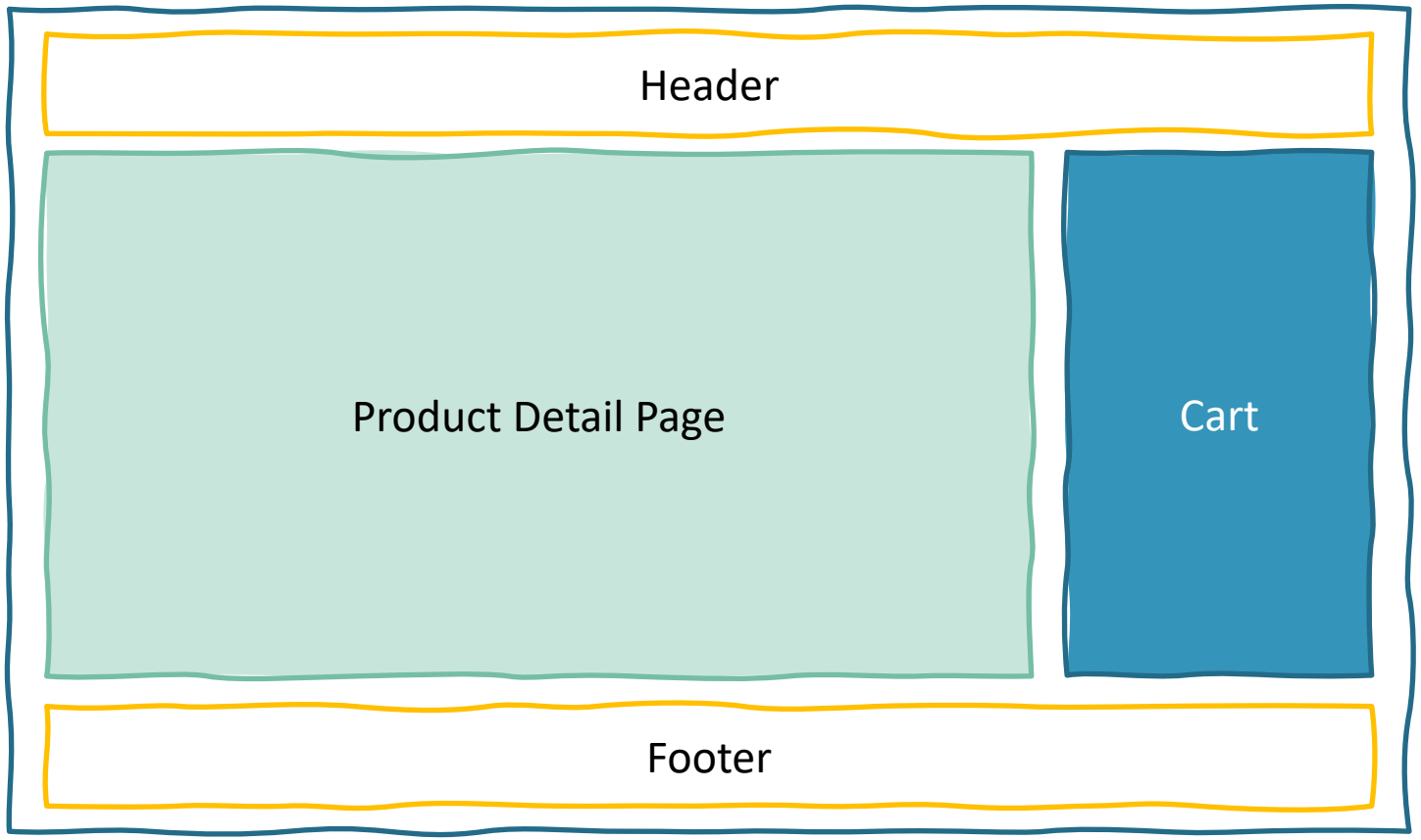
# Changes in the Build and Deployment Process



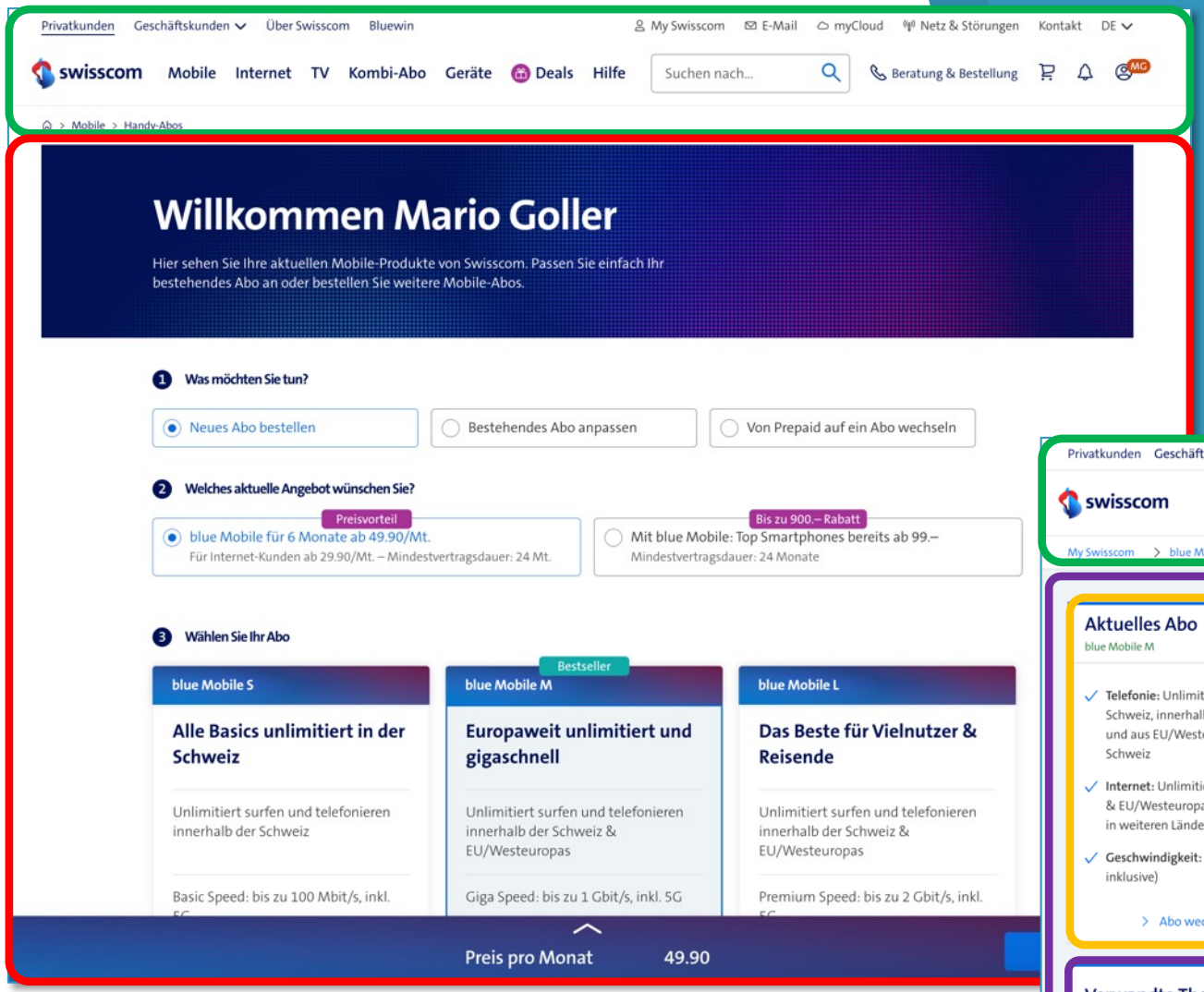

https://www.trendmicro.com

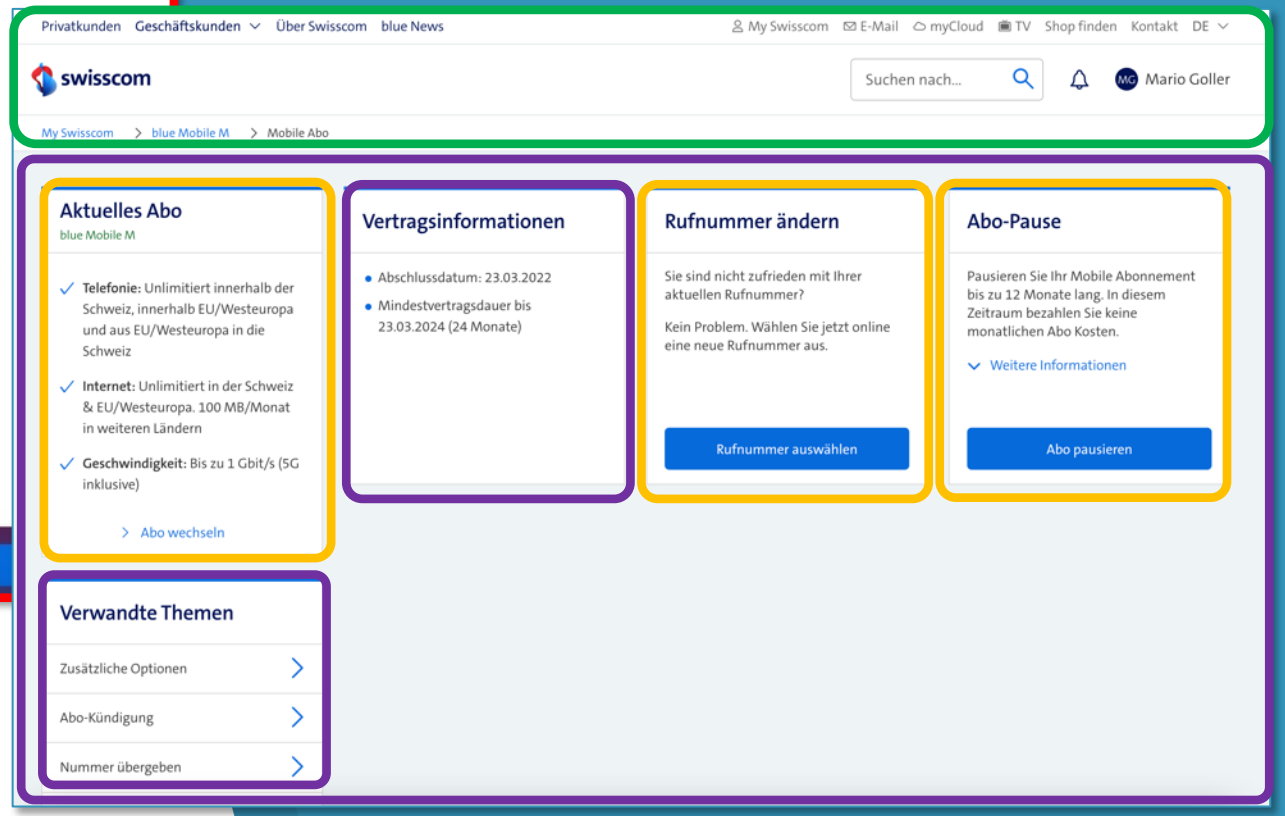
# How to identify Micro Frontends in your Application

- Defnied around Domains, Subdomains
- Creating a Bounded Context

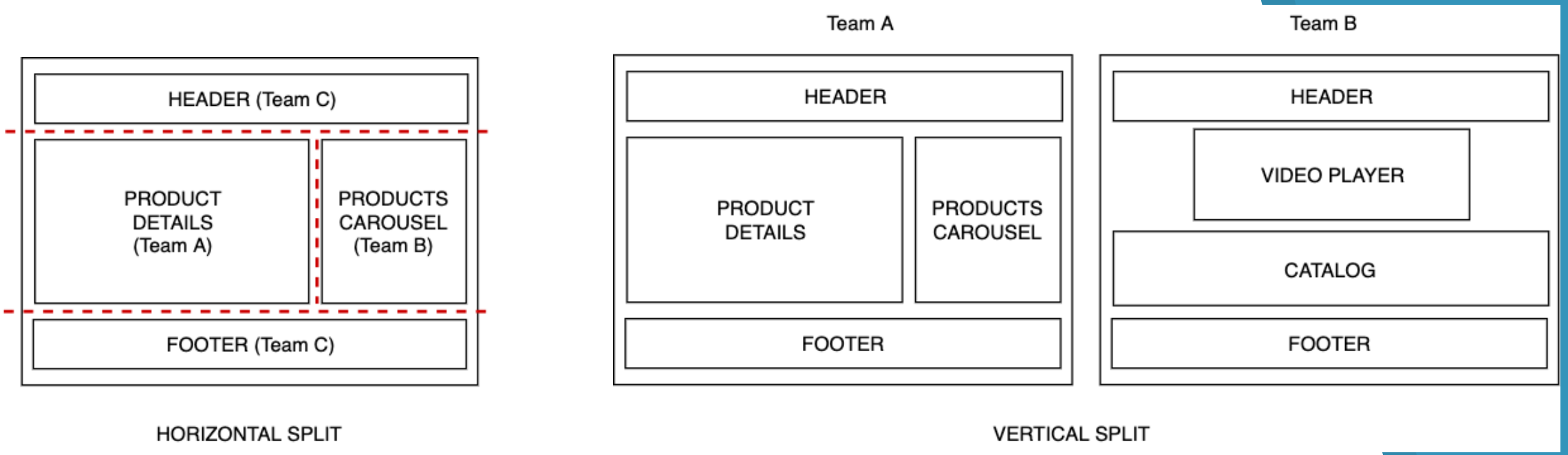


# Customer Portal @ Swisscom



# Splitting your Application




## Things we need to consider

- Composition of fragments
- Routing between views
- Communication between fragments

https://medium.com/@lucamezzalira/micro-frontends-decisions-framework-ebcd22256513

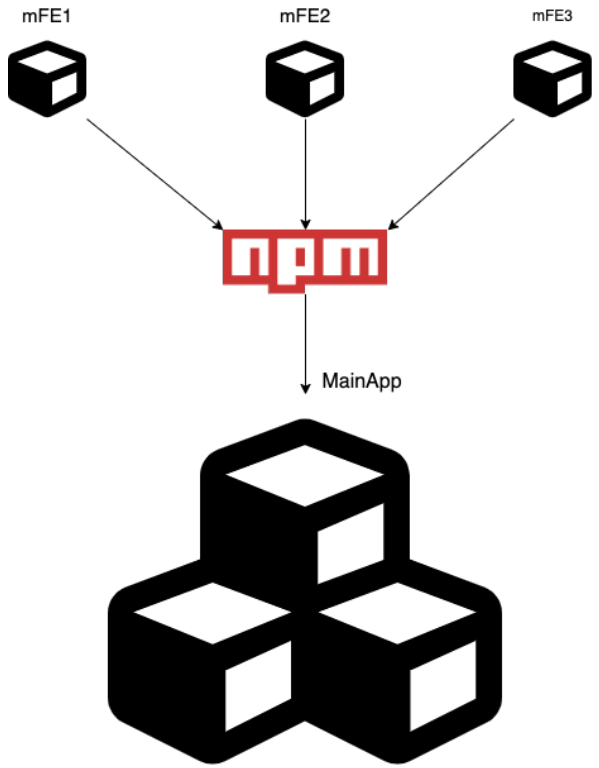
# Integrating Multiple Frontends

**Build-time Integration**


**Runtime Integration**

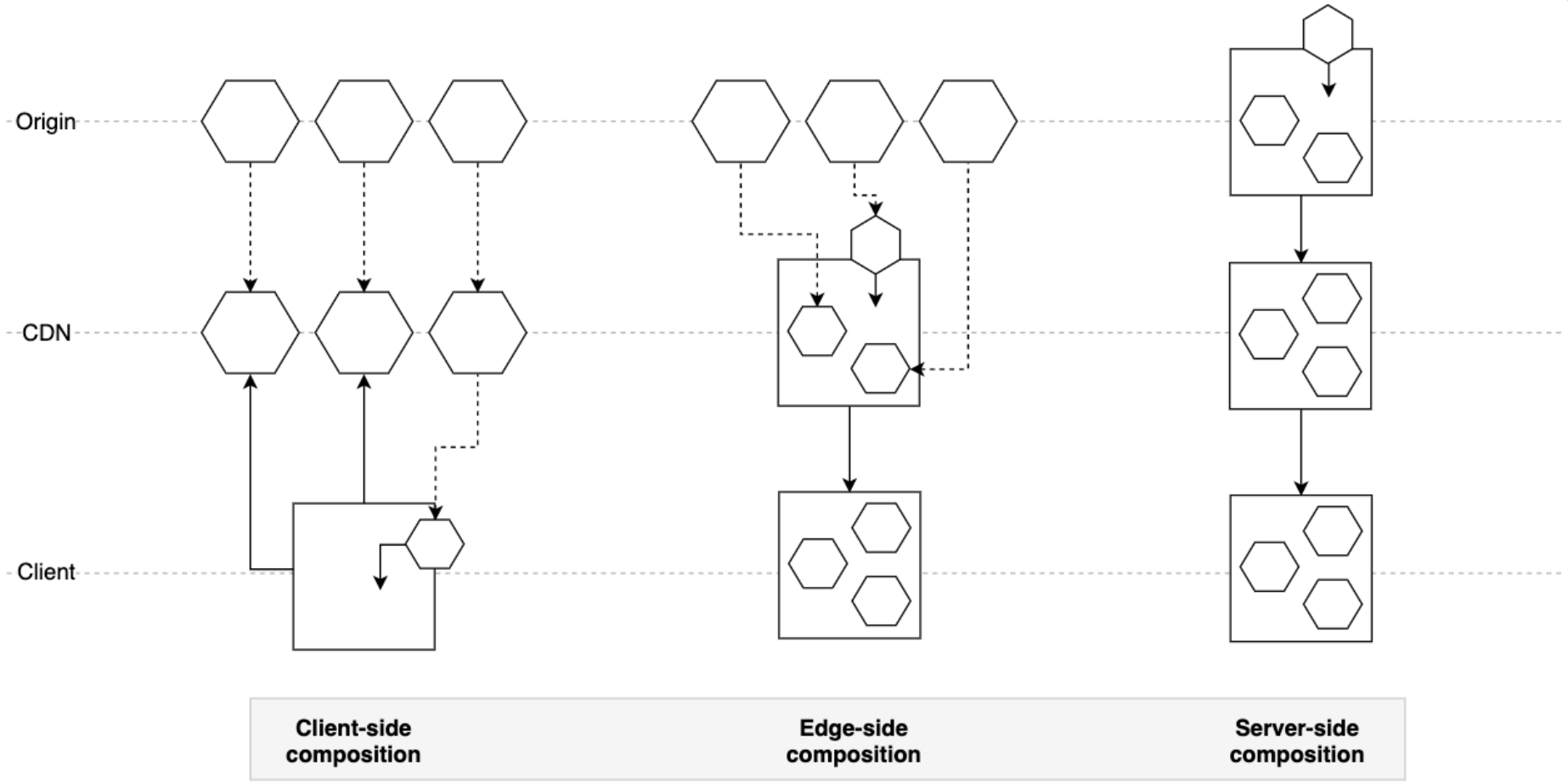
# Build-Time Integration

- Defining and bundling dependend UI parts at build time
- Each micro-frontend is a package
- Package is integrated to the Application Shell (MainApp)
- Whole Application is built and deployed

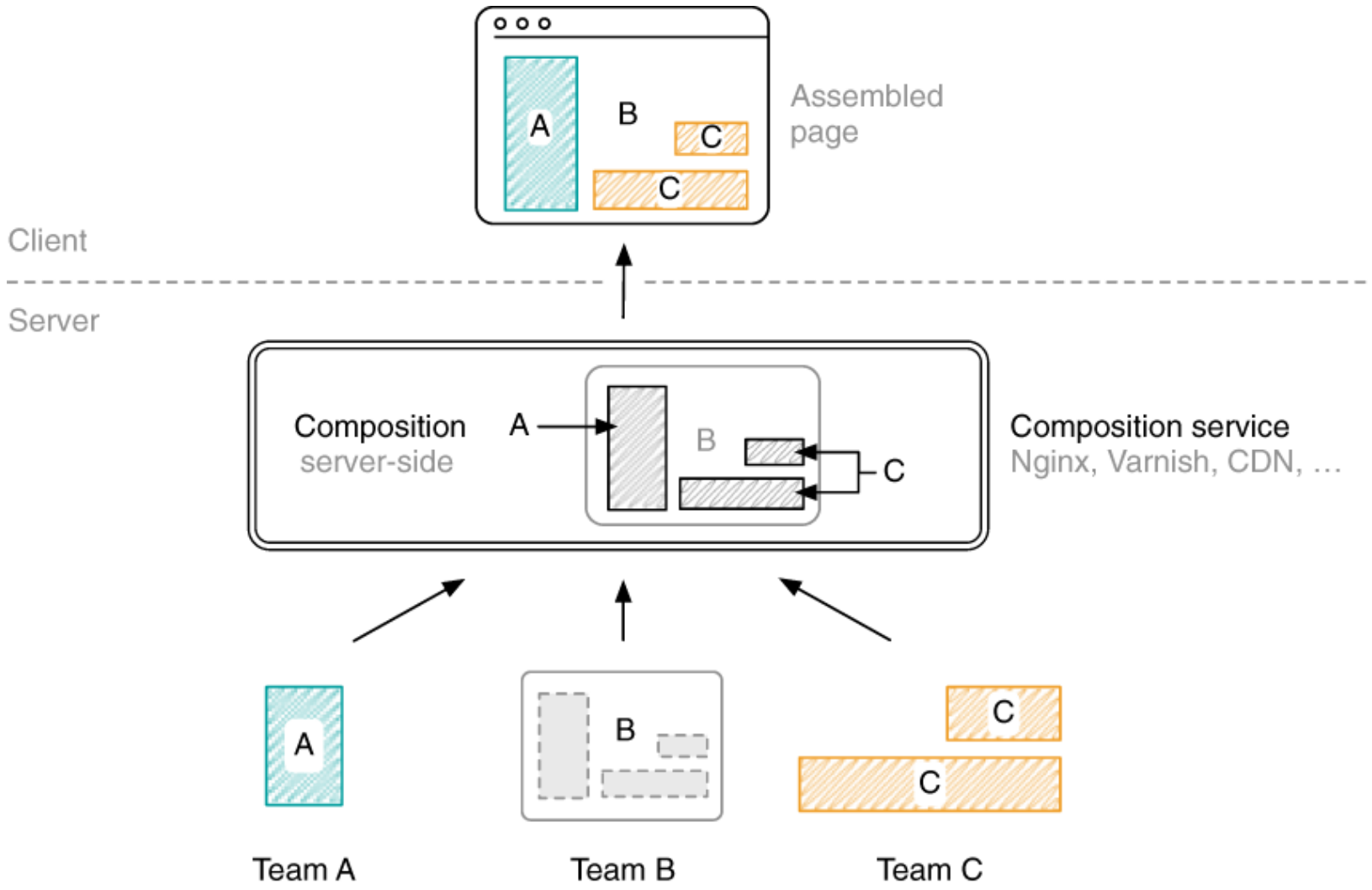


# Runtime Integration

- Load and integrate dependend UI parts (fragments/views) when the page gets requested by the client



# Server Side Composition




Micro Frontends in Action (Michael Geers)

# Server Side Includes (SSI)

NGINX configuration

```
...
server {
  listen 3000;
  ssi on;
  ...
}
```

- server-side template composition technically works well with SEO
- it can’t load the new Micro-Frontend for the client without a full page reload at most of the time

Main webpage markup

```
...
<aside class="decide_recos">
  <!--#include virtual="/shop/fragment/recommendations/notebooks" -->
</aside>
...
```

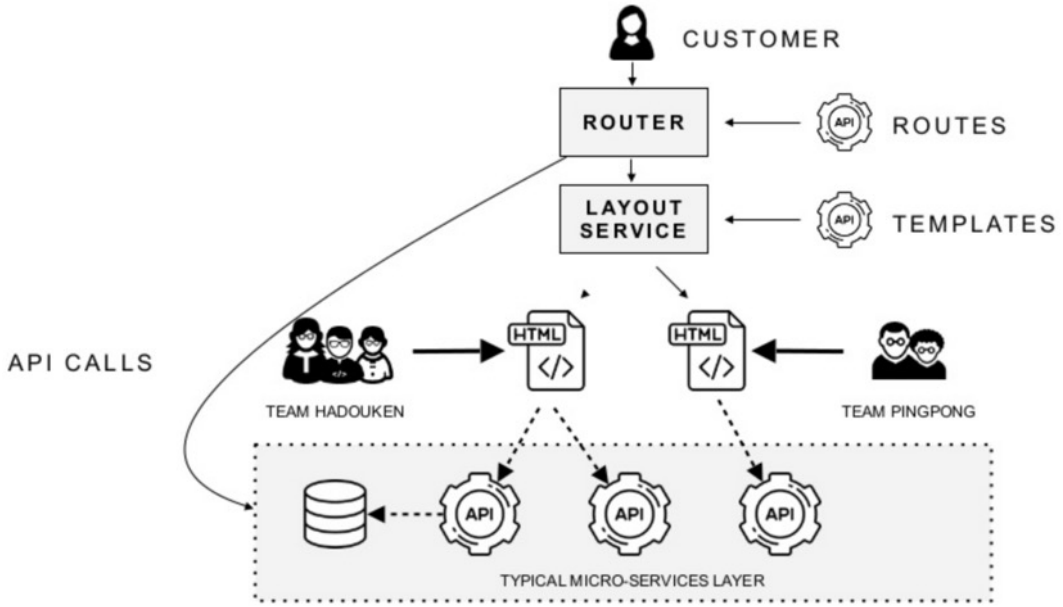
# Other solutions

## Edge Side Includes (ESI)

- ESI is a simple markup language that allows the inclusion of fragments from other URLs.

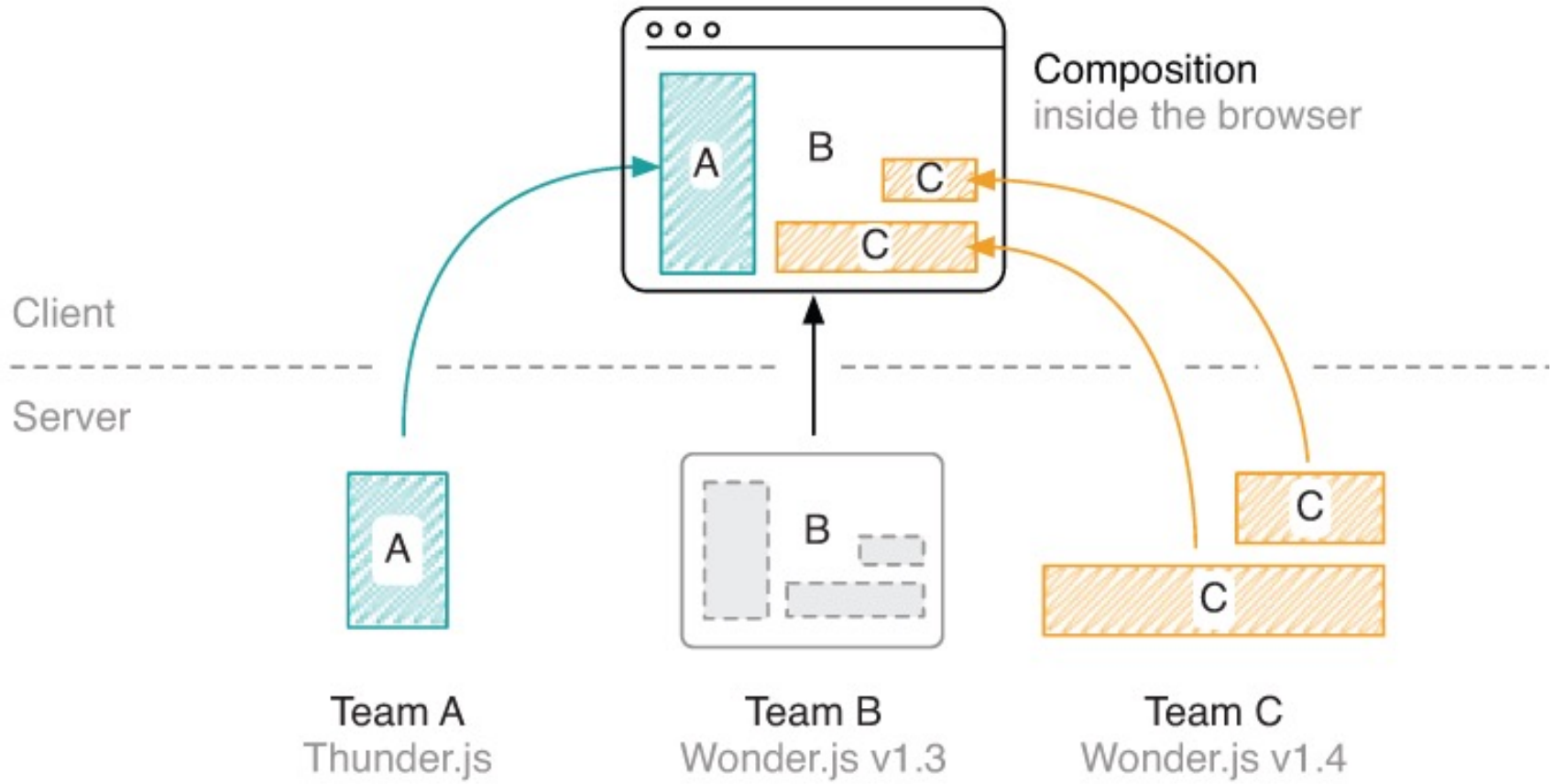
## Zalando Tailor.js

- Tailor is a Node.js based fragment service open-sourced by Zalando and runs as stand-alone service
- It is part of Project Mosaic - umbrella project for frontend modularization




https://levelup.gitconnected.com

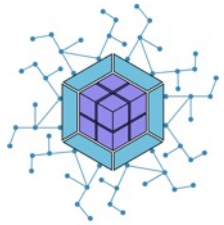
# Client Side Composition



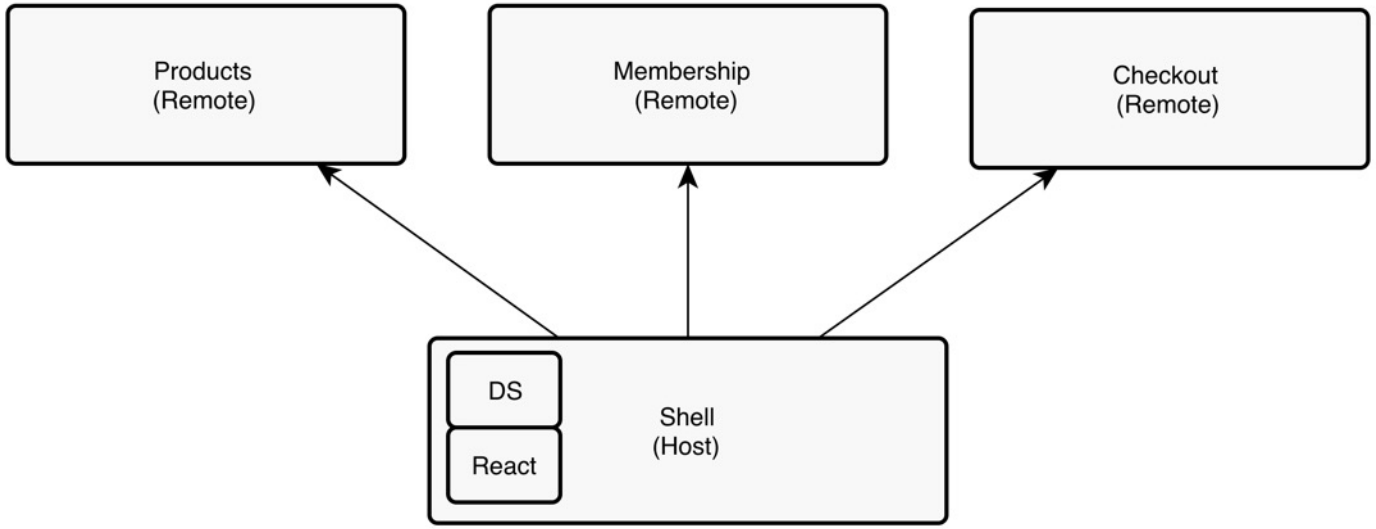

Micro Frontends in Action (Michael Geers)

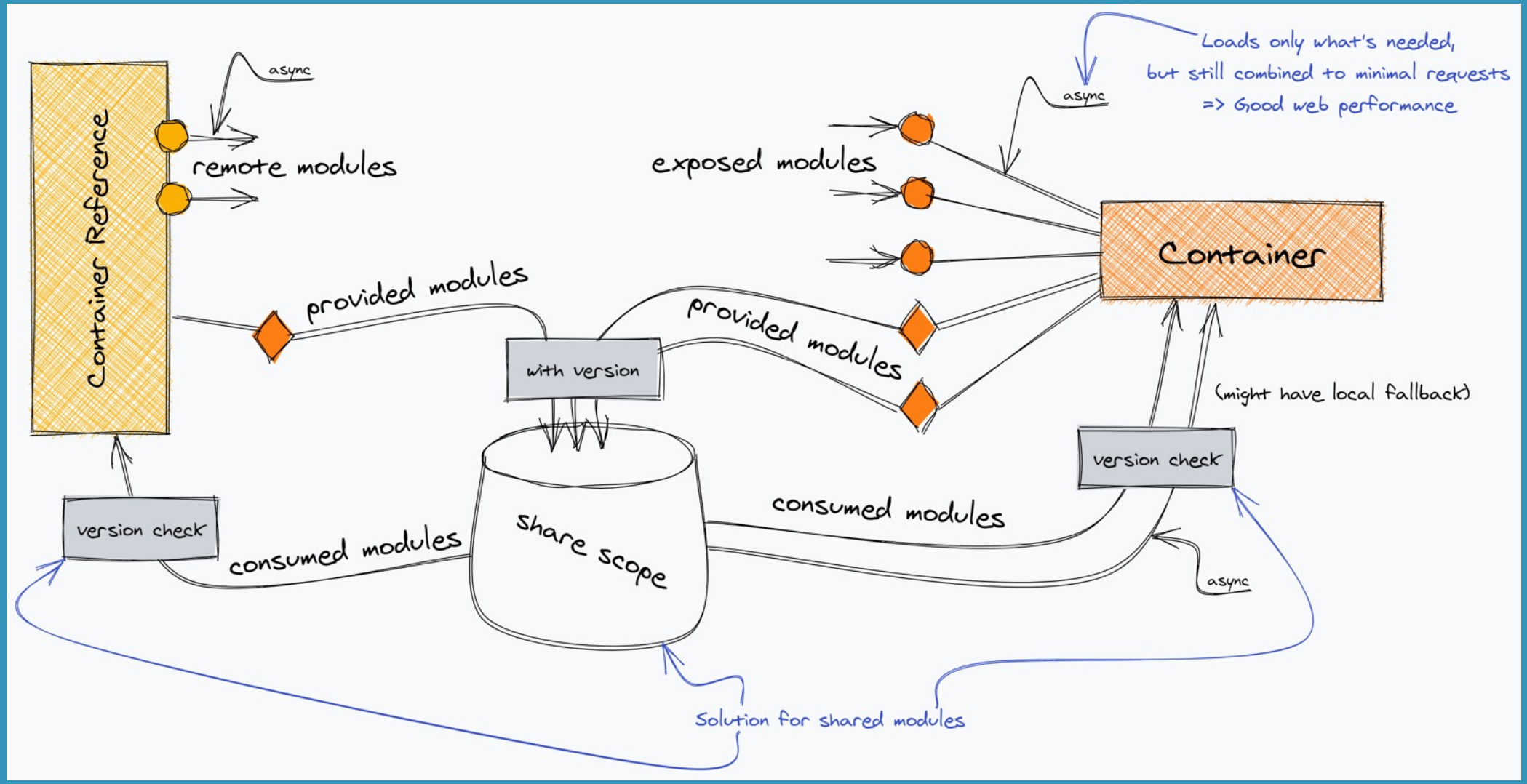
# Using Webpack 5 Module Federation

https://webpack.js.org/plugins/module-federation-plugin/



- each part of the frontend would be a separate build
- These builds are compiled as "Containers”
- Containers can be referenced by applications or other containers
- Container is the **remote** and the consumer of the container is the **host**.
- The "remote" can **expose modules** to the "host”
- The "host" can use such modules ("remote modules”)





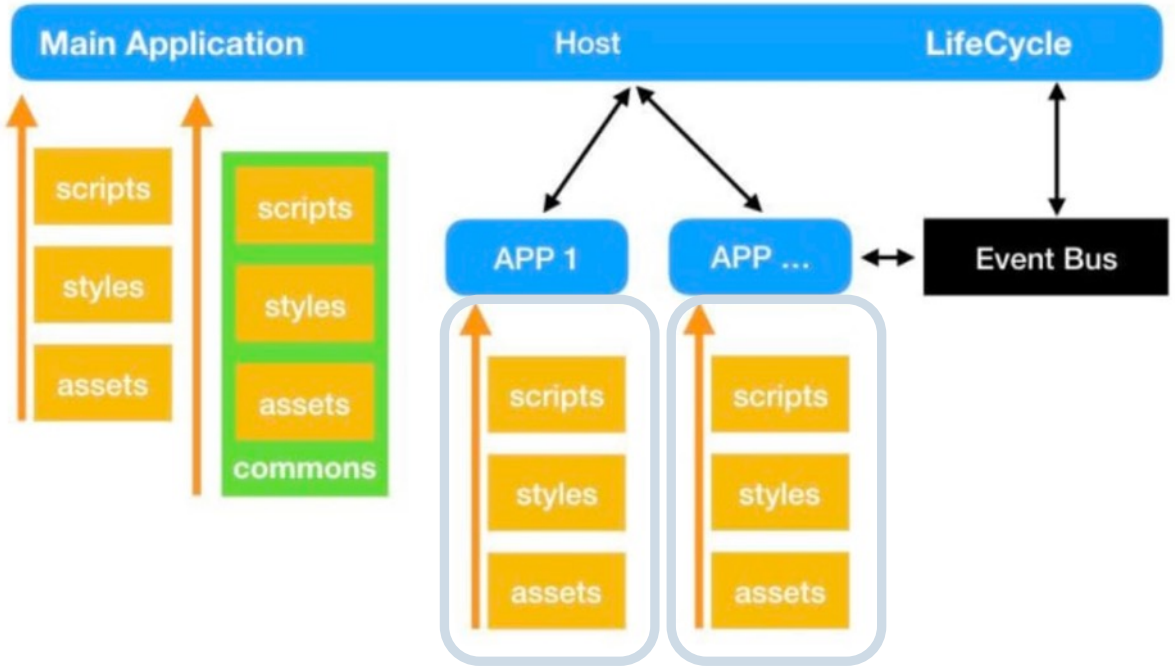

https://github.com/sokra/slides/blob/master/content/ModuleFederationWebpack5.md

# Demo

# *Challenges with Micro-Frontends*

# Web Performance

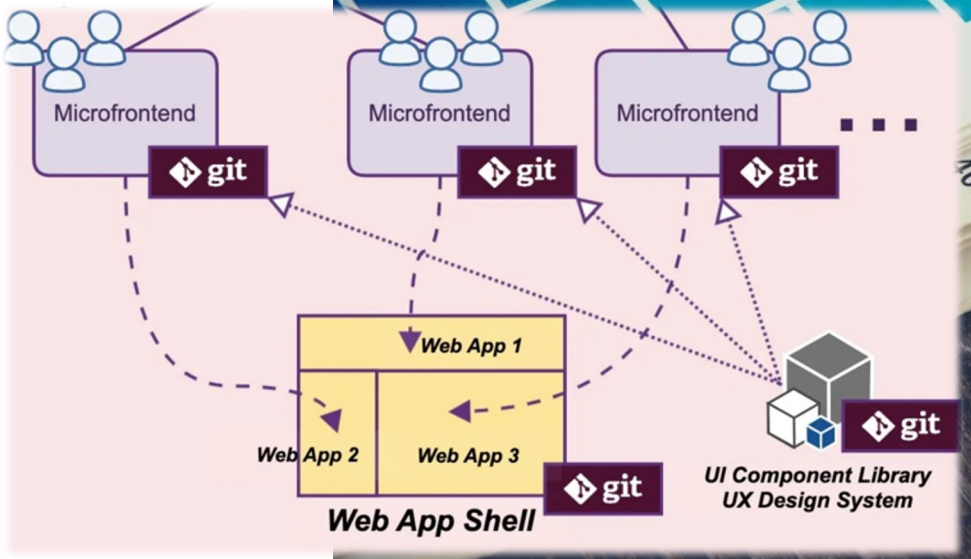
- **Bundle size -** Each micro application includes its own copy of the framework and the entire toolkit.
- **Loading Time** - Multiplies the amount of data downloaded by the user
- Vendor packages can solve this problem (However, it violates the idea of team independence)



# UI/UX Consistency

- **Design System**
  - Design Tokens (fonts, colors, …)
  - (Re-)Usable Components (buttons, …)
  - Responsive Design (Breakpoints, Grid, …)
  - Accessibility

- **Shared fundamentals**
  - Basic Styles & Typography
  - CSS Resets
  - Tracking
  - Error Handling



# Conclusion

## When to use micro frontends

- ✓ Huge code base where different teams are contributing to
- ✓ Code ownership get messy
- ✓ Deployment is delayed because of other part of the application
- ✓ You have to use different FE frameworks

## When not to use micro frontends

- ❖ Small organization, small number of teams
- ❖ Backend is owned by dedicated centralized teams
- ❖ everyone works on separate modules rather than together towards one product
- ❖ Software Modularization is already managed very well (aka. Modulith)


Q
&
A

*mario.goller@swisscom.com*

*@gollerMario*


Photo by Matt Jones on Unsplash