Divide and Conquer with Micro-frontends

- *Vishal Sharma*

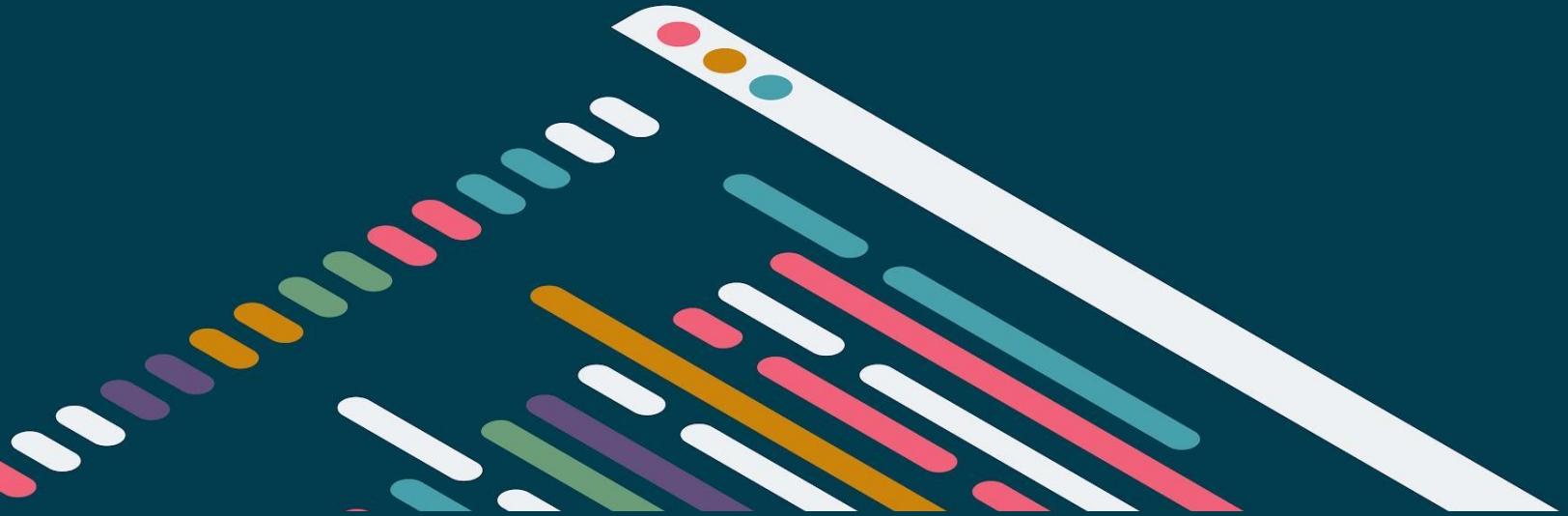
Who am I?

Vishal Sharma

Lead Consultant, Thoughtworks

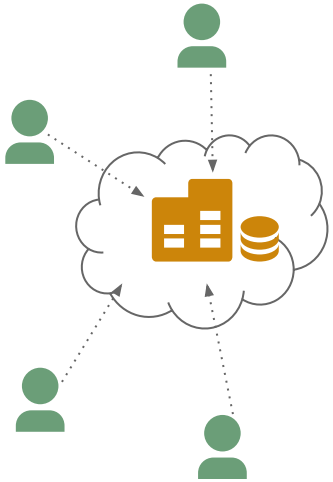
https://www.linkedin.com/in/vishal-sharma-29749bb6/

https://medium.com/@sharvishi9118

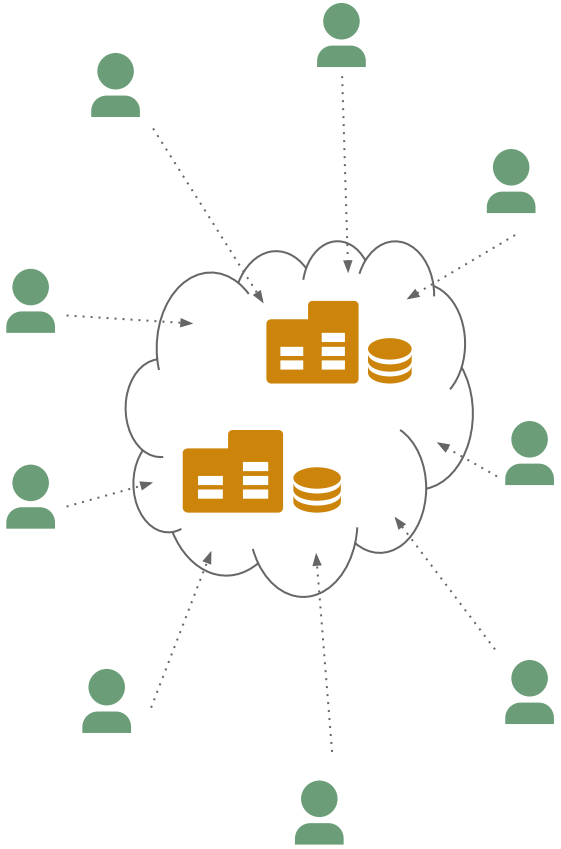


Why do we need Micro-frontends?

Scaling application

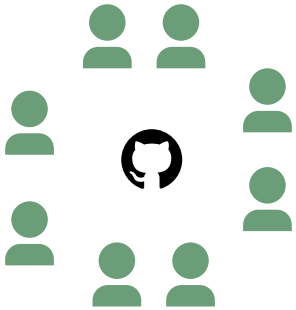


- Increased performance
- Reduced Latency
- Managing high volumes

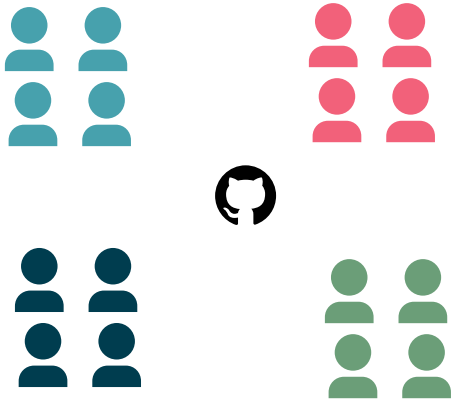


Scaling application teams

Vertical scaling of teams

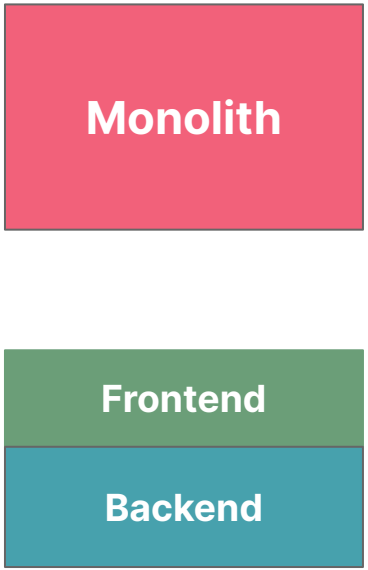


Horizontal team scaling



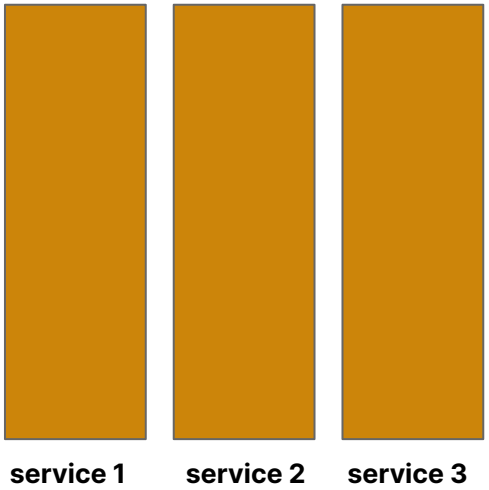
Microservices for the backends and Micro-frontends for the frontend

Monoliths



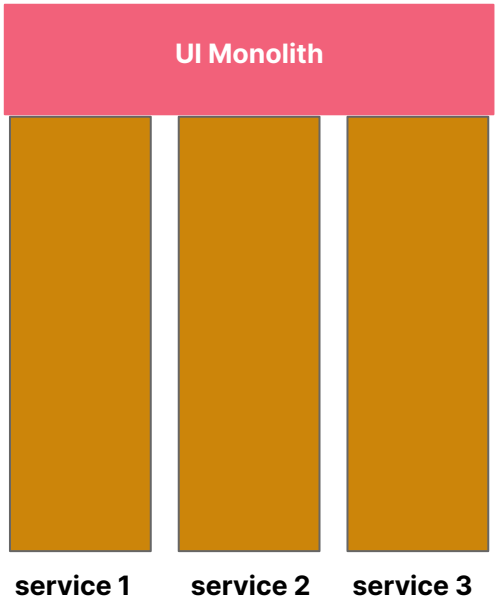
Monoliths aren’t a bad design choice

Microservices



- Strong module boundaries
- Independant deployments
- Diversified Technologies

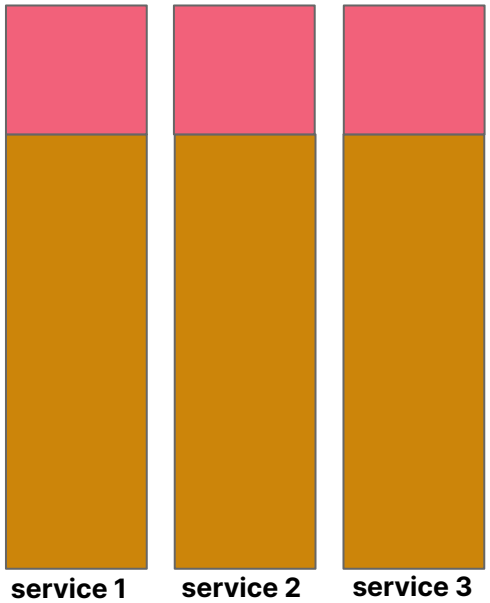
MicroServices + UI Monolith



There was mostly a monolith UI, which made independent deployments less true.

Independent deployments aren’t independent releases.

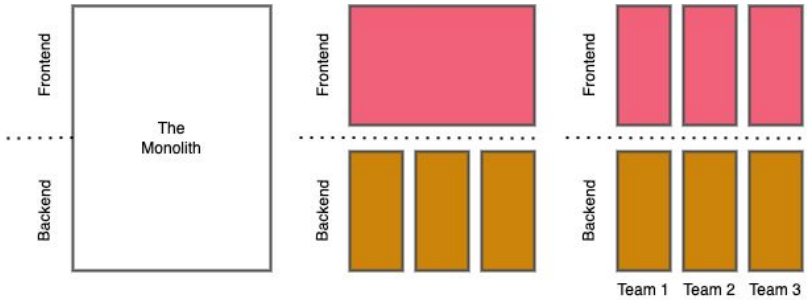
Micro-frontends



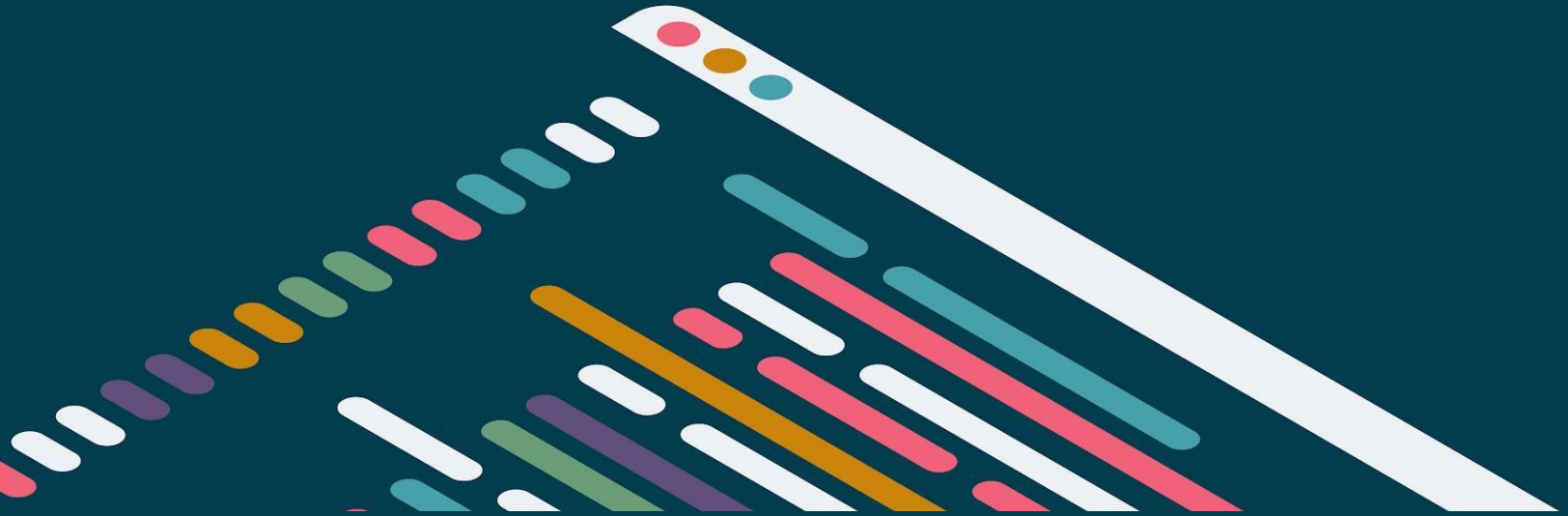
Micro frontends is an architectural design pattern to break down monolith UI to a smaller and manageable deployable units.

Microservices’ promises + Independent releases.

What you get from Micro-frontends



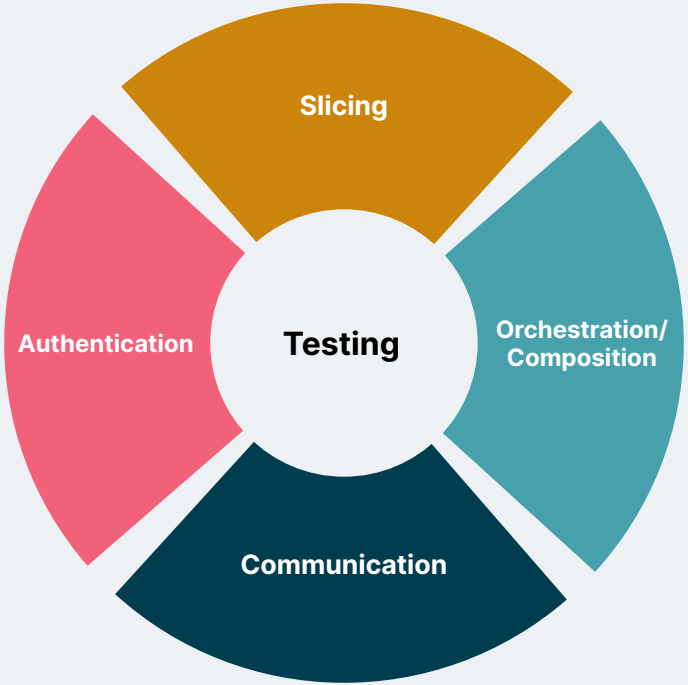
- Independent teams
- Independent releases
- Simple, decoupled codebases
- Incremental upgrades



Key decisions when starting Micro-frontends

Key Questions that need to be answered

1. **Slice**: How to break an application to separate pieces
2. **Composition**: How do multiple micro-frontends come together to form a single application.
3. **Communication**: How to communicate between micro-frontends
4. **Authentication**: How different MFEs can get authenticated
5. **Testing**: How should we test the micro-frontends



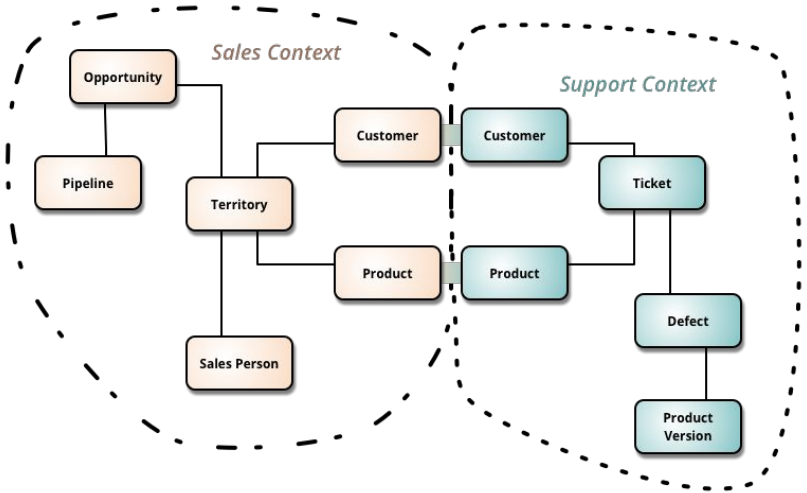
Slicing the Micro-frontends

- Don’t start thinking it as components.
- Page level vs Cross sectional
- Case by case, create a framework to ideate and slice.

Bounded Context Pattern

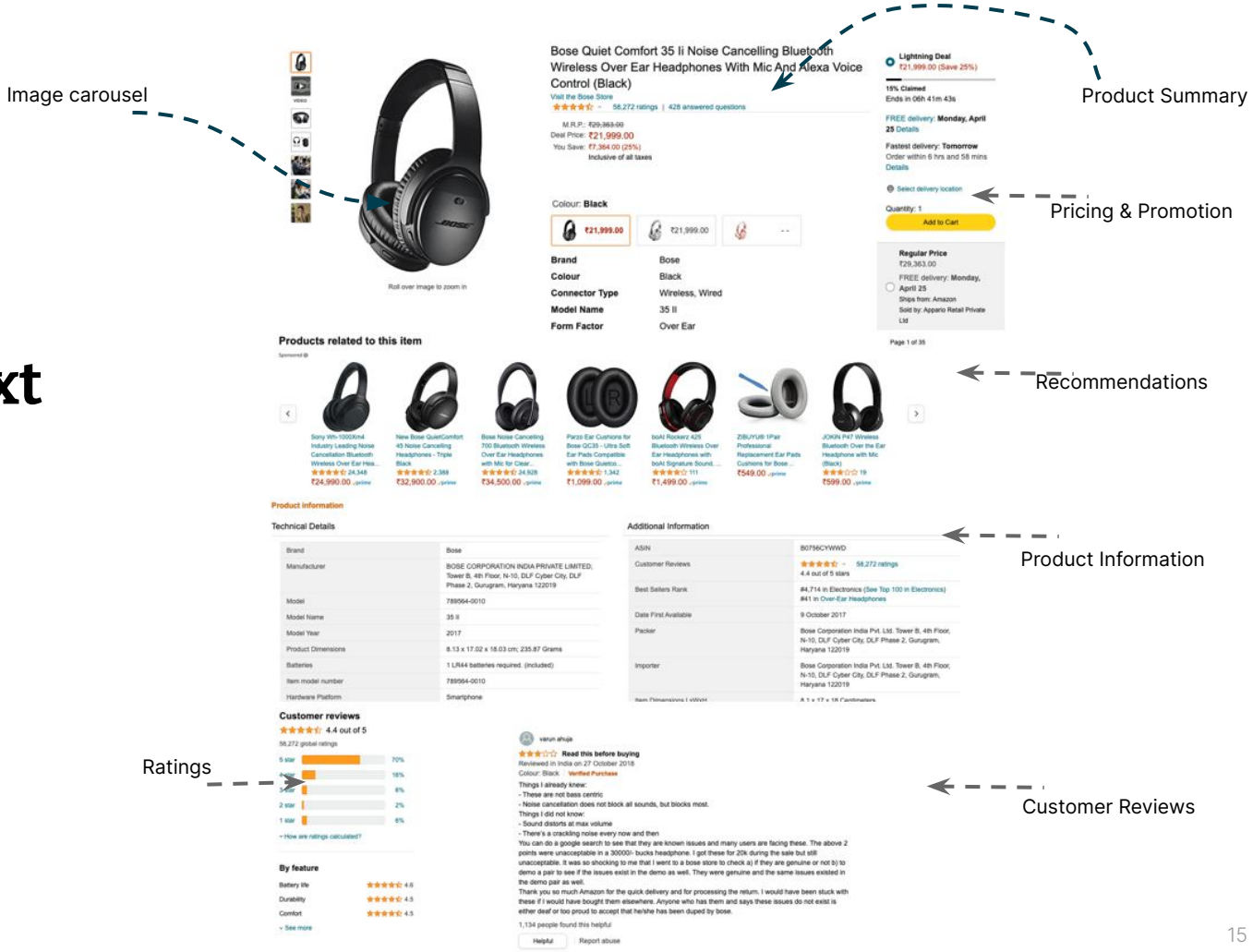
“Total unification of the domain model for a large system will not be feasible or cost-effective”

- *Eric Evans (DDD)*



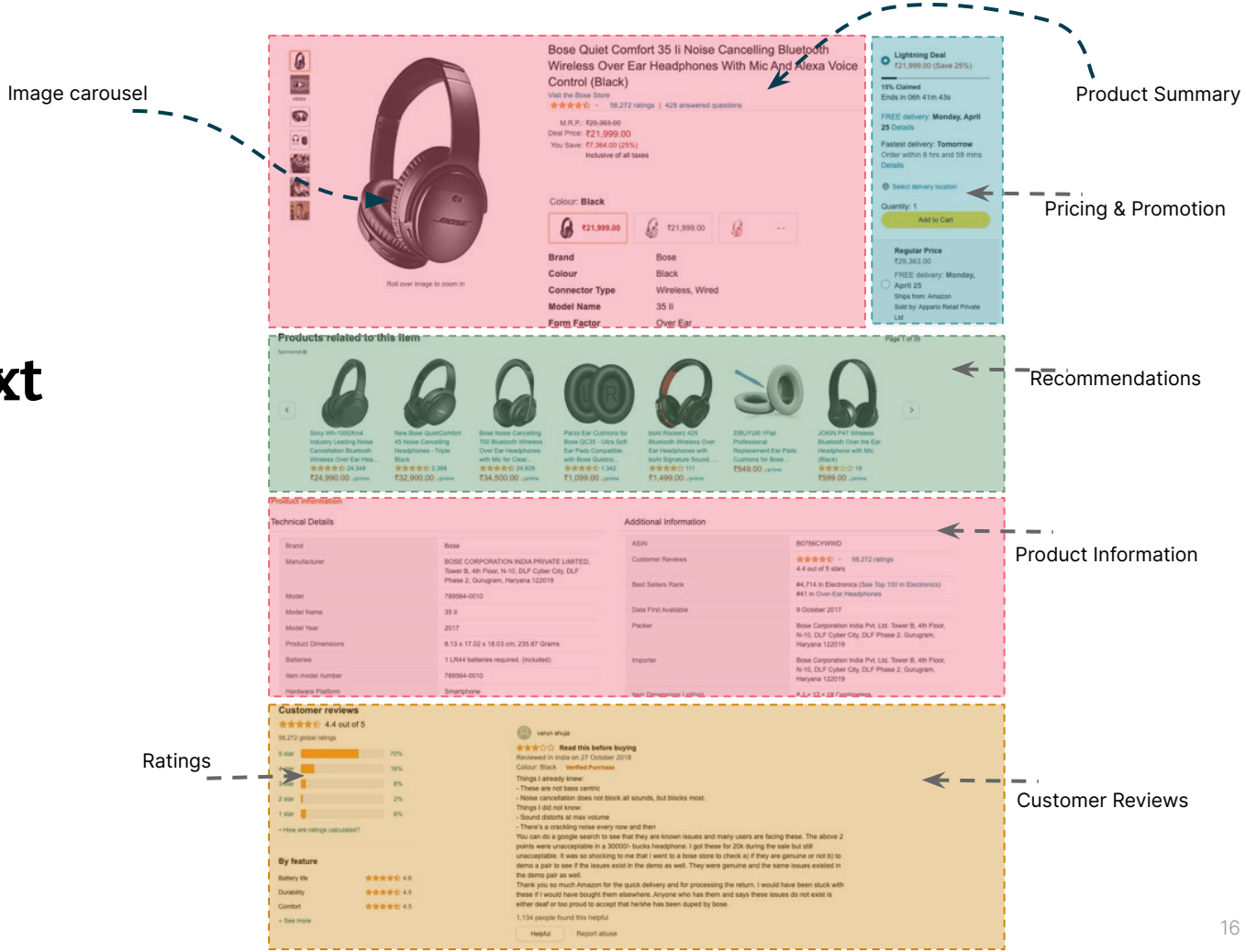
Bounded Context Example:

Product Detail Page

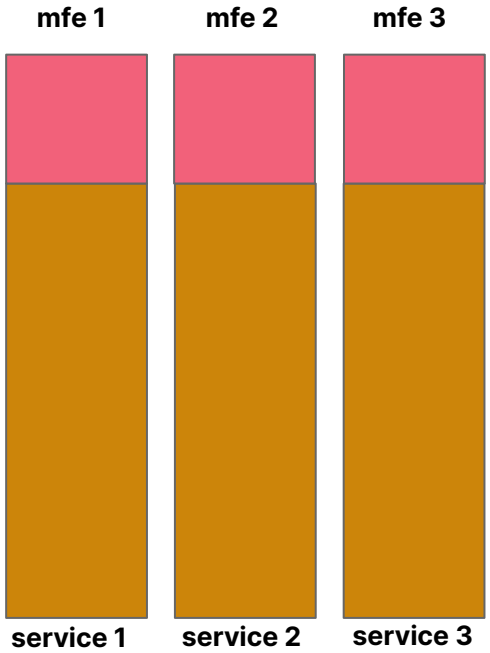


Bounded Context Example:

Product Detail Page

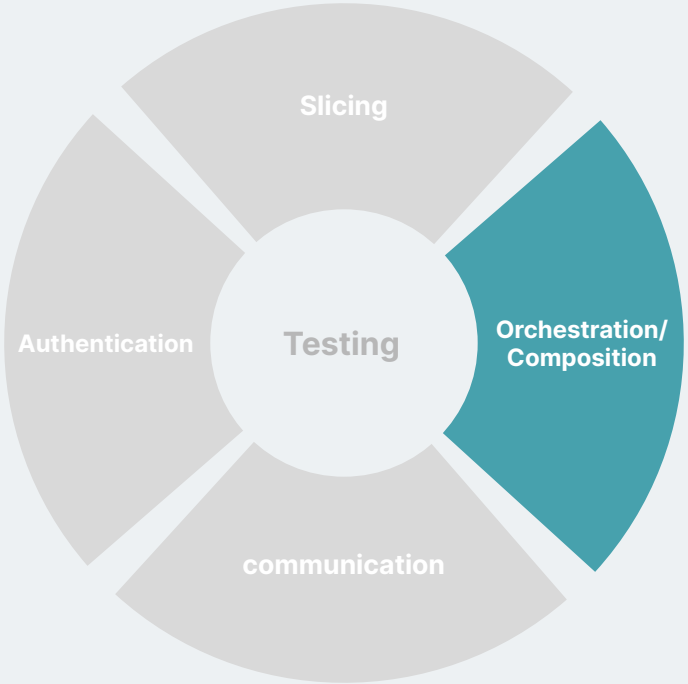


Sub domain teams

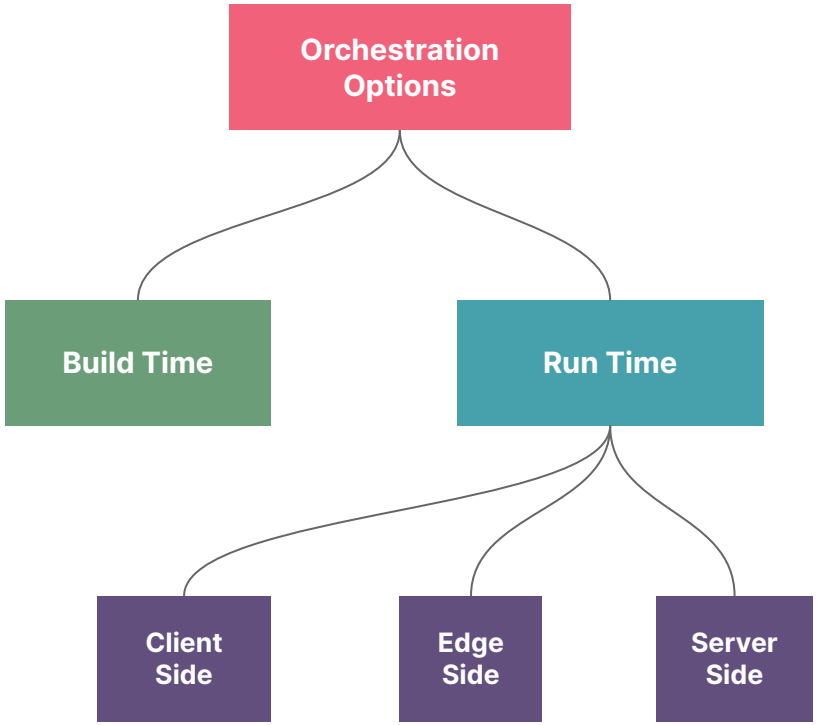


Key Questions that need to be answered

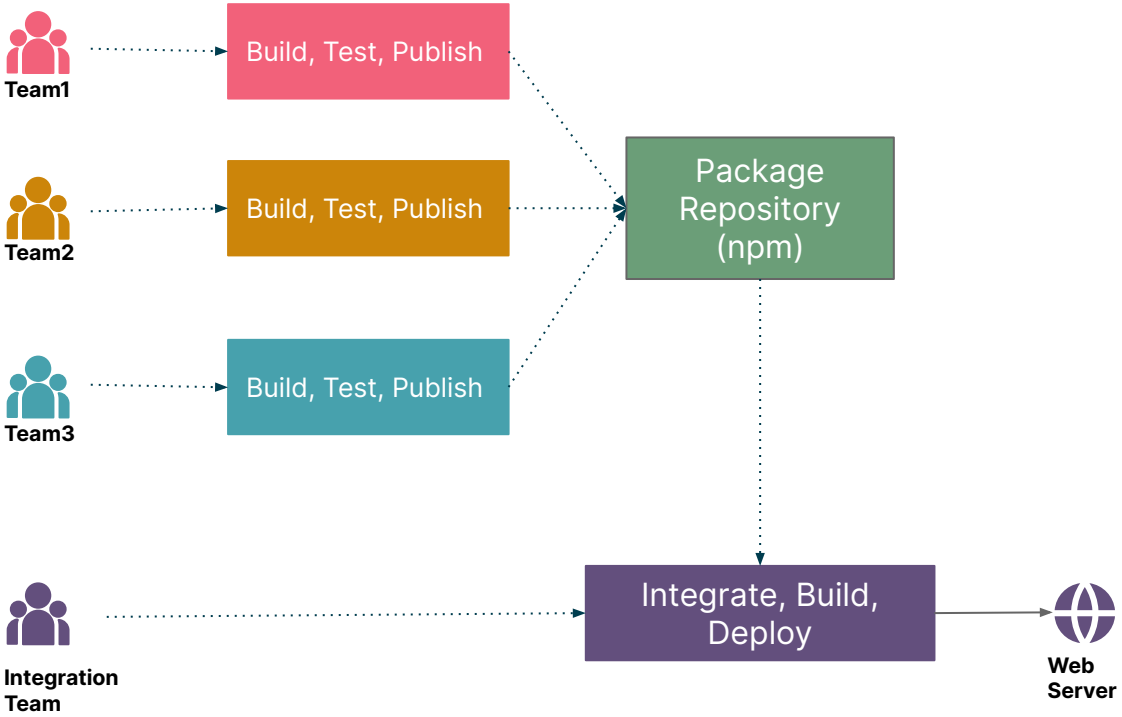
1. **Slice**: How to break an application to separate pieces
2. **Composition**: How do multiple micro-frontends come together to form a single application.
3. **Communication**: How to communicate between micro-frontends.
4. **Authentication**: How different MFEs can get authenticated
5. **Testing**: How should we test the micro-frontends.



Orchestrating Micro-frontends



Build time composition



aka
“Distributed Monolith”*

** - use it with extreme caution, can become worse than a monolith*

Run time composition

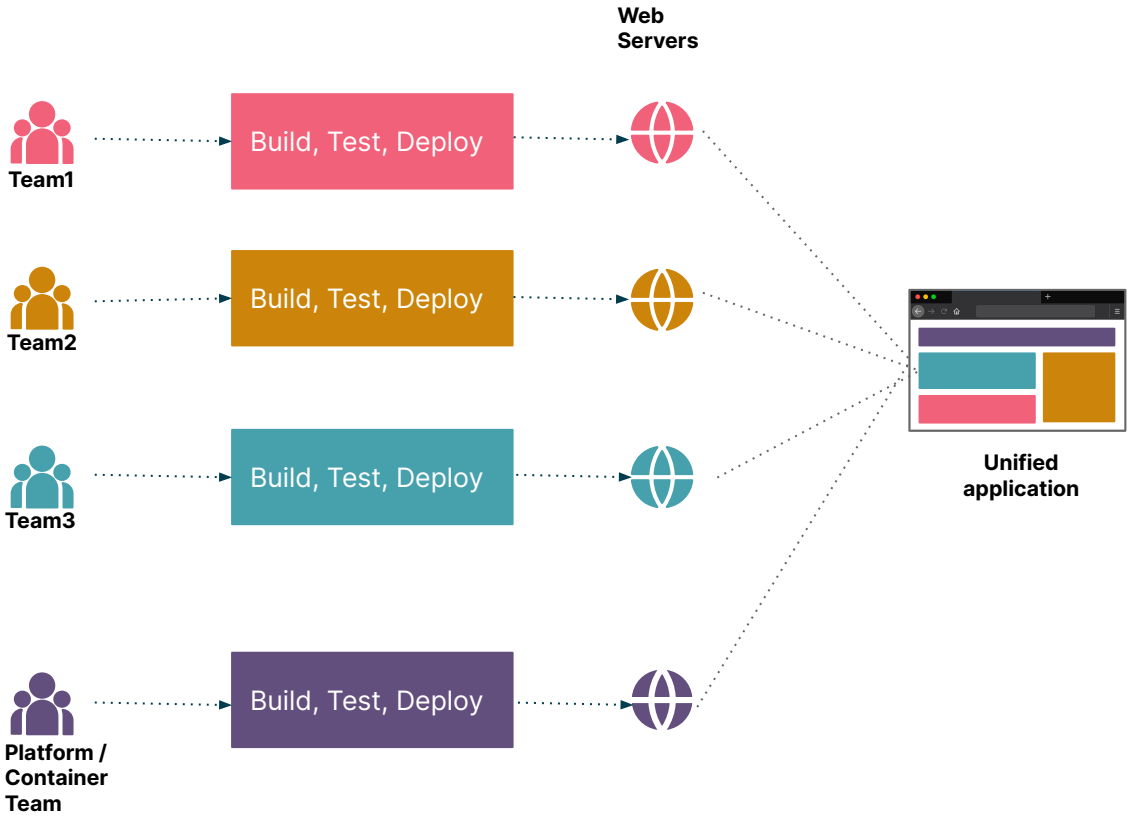
Different micro-frontends gets unified dynamically into a single page (or view) based on the user's request.

1.
Client Side Composition

2.
Edge Side Composition

3.
Server Side Composition

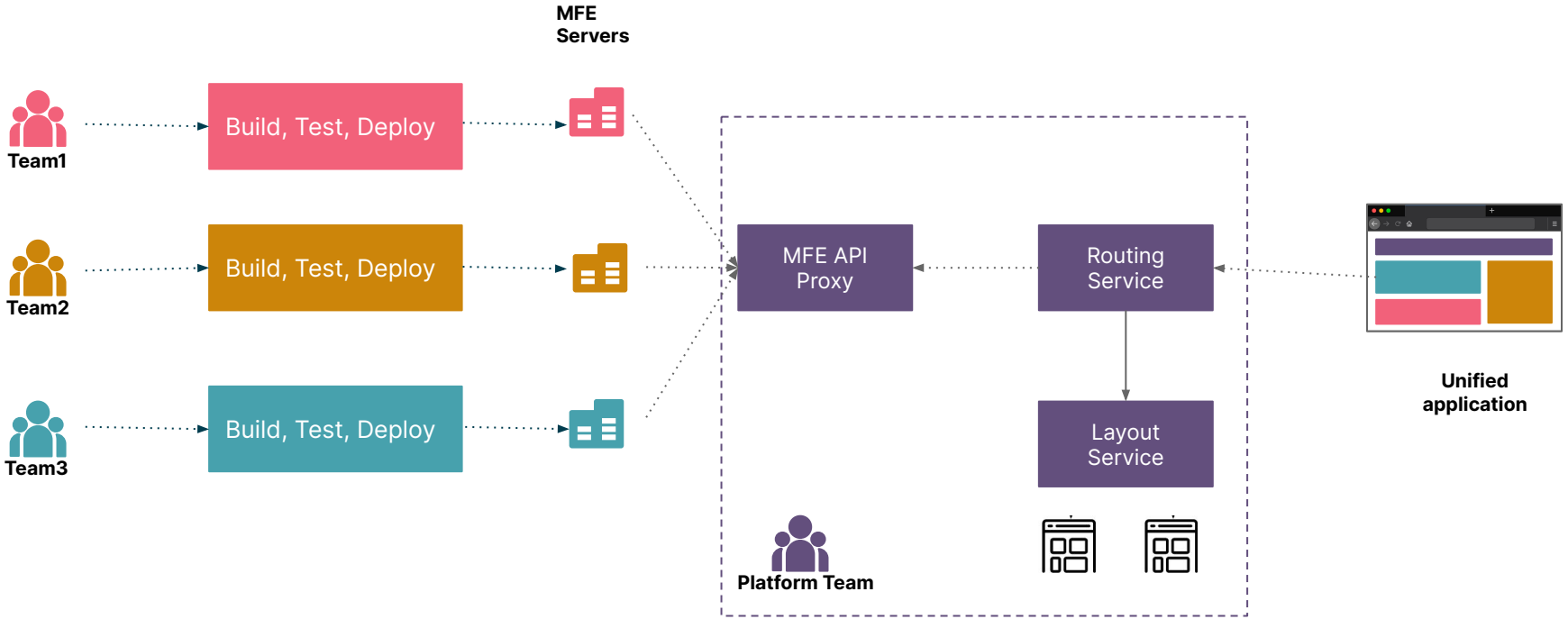
Client Side Composition



Client Side Composition - Options

- Iframe based integration - luigi-project.io
- Webpack plugins - module-federation-plugin
- SingleSpa Framework - single-spa.js.org
- Client side transclusions - h-include
- Or Write your 20 lines of JS :)

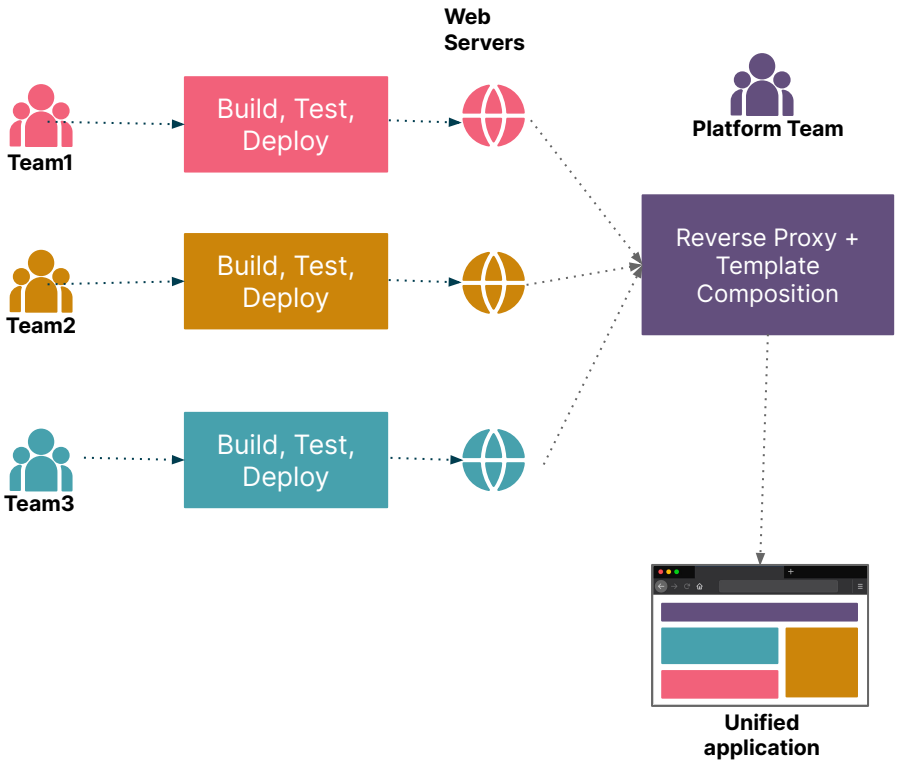
Server Side Composition



Server Side Composition - Options

- Ara Framework - ara-github
- Tailor - project-mosaic
- Hypernova - airbnb-hypernova
- Open components - open-components
- Piral - piral.io
- Podium - podium-lib

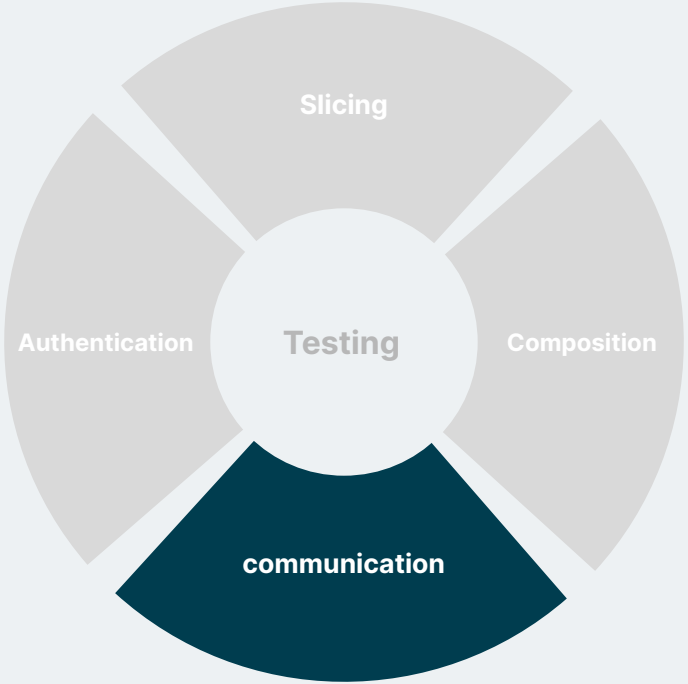
Edge Side Composition




```
<html>
  <head/>
  <body>
    <div id="content">
      <div id="header">
        <img src="/images/logo.jpg" />
        **<esi:include src="/shopping_cart" />**
      </div>
      <div id="items"/>
    </div>
  </body>
</html>
```

Key Questions that need to be answered

1. **Slice**: How to break an application to separate pieces
2. **Compose**: How to build a page / functionality using different pieces
3. **Communication:** How do the micro-frontends communicate with each other.
4. **Authentication**: How different MFEs can get authenticated
5. **Testing**: How should we test the micro-frontends.



Micro-frontends communication

Storage based

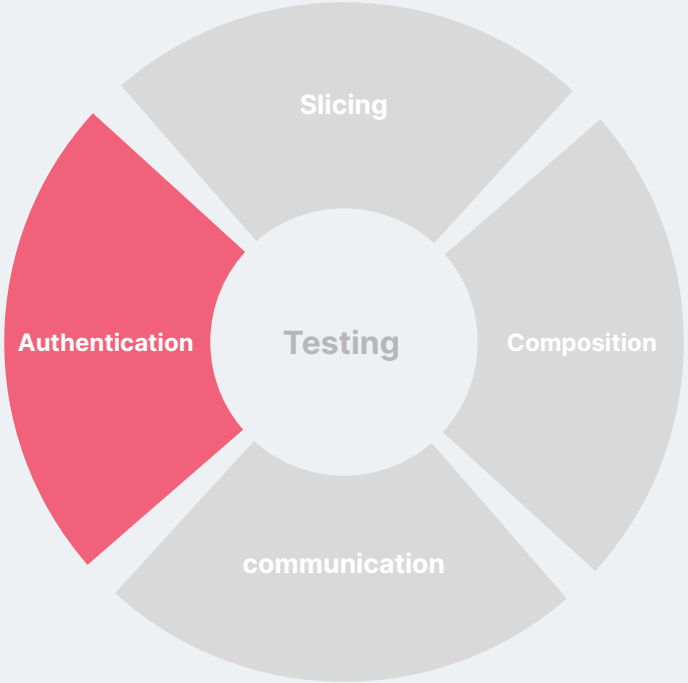
Using a global context or window objects it's possible to communicate with Micro-frontends

Event based

Using JS eventing we can communicate across Micro-frontends.

Key Questions that need to be answered

1. **Slice**: How to break an application to separate pieces
2. **Compose**: How to build a page / functionality using different pieces
3. **Communication:** How do the micro-frontends communicate with each other.
4. **Authentication**: How different MFEs can get authenticated
5. **Testing**: How should we test the micro-frontends.

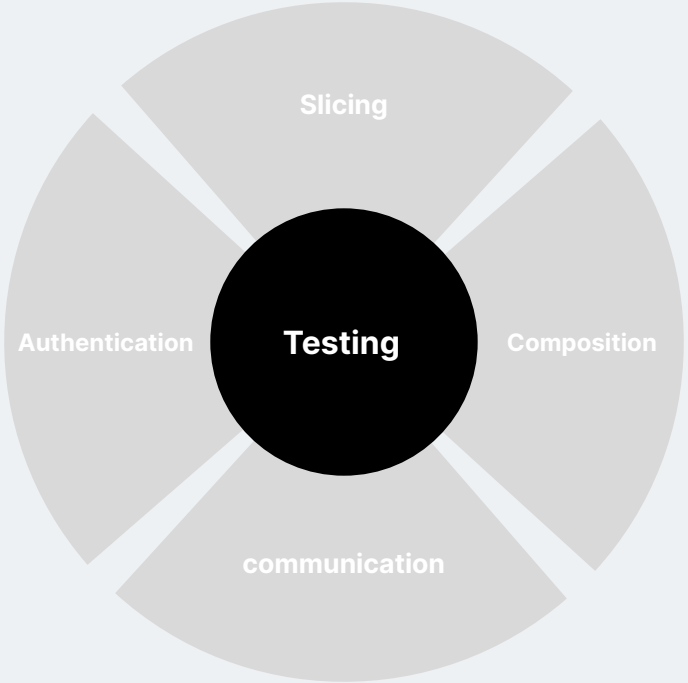


Authenticating the Micro-frontends

- How will each authenticated micro frontend run independenty

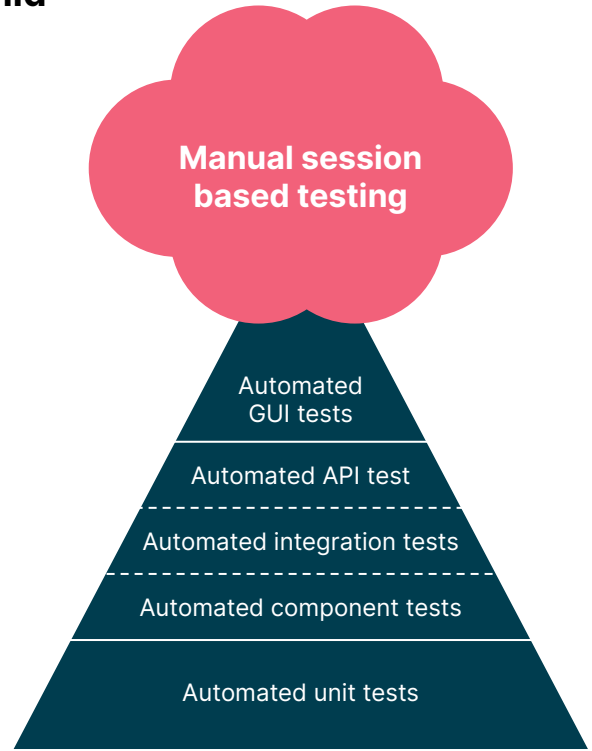
Key Questions that need to be answered

1. **Slice**: How to break an application to separate pieces
2. **Compose**: How to build a page / functionality using different pieces
3. **Communication:** How do the micro-frontends communicate with each other.
4. **Authentication**: How different MFEs can get authenticated
5. **Testing**: How should we test the micro-frontends.

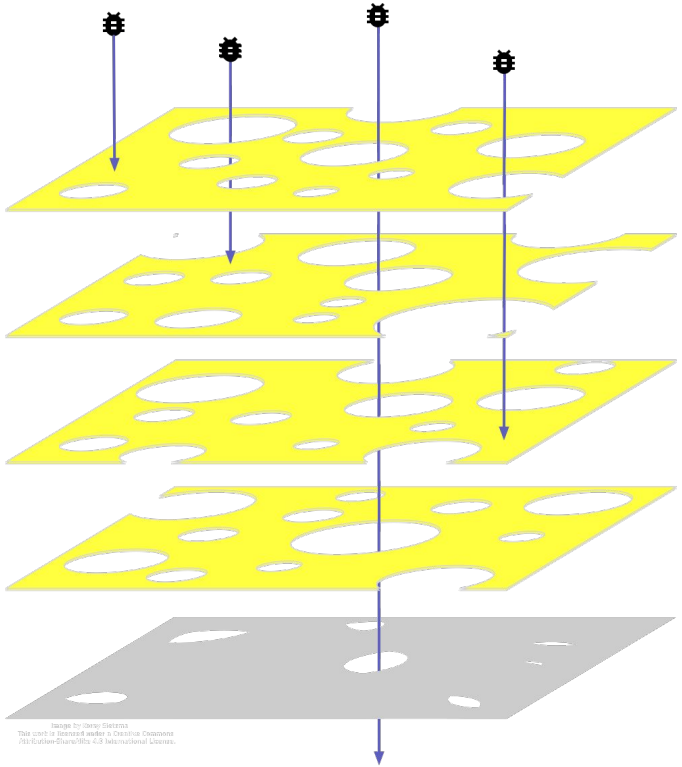


Testing Strategies

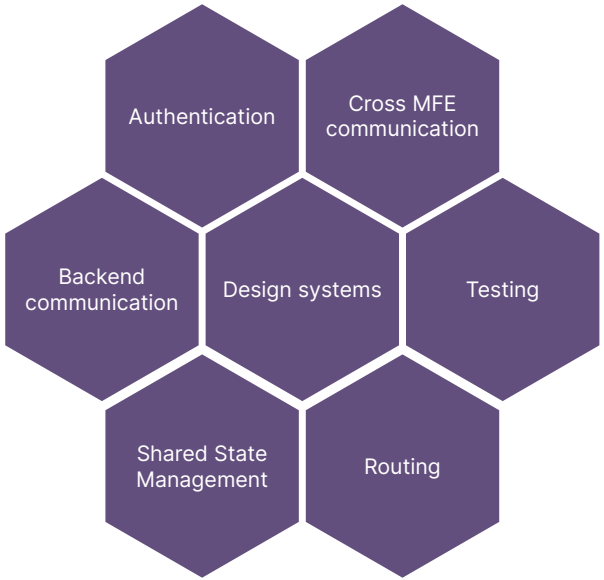
Pyramid



Swiss Cheese - Korny's Blog



Technical aspects as per your choice of integration approach



Team formation

- Team structure and software structure should align
- Follow domain driven design. “bounded context” principles
- Vertical team structure: One team, one application/MFE
- Build a cross functional team

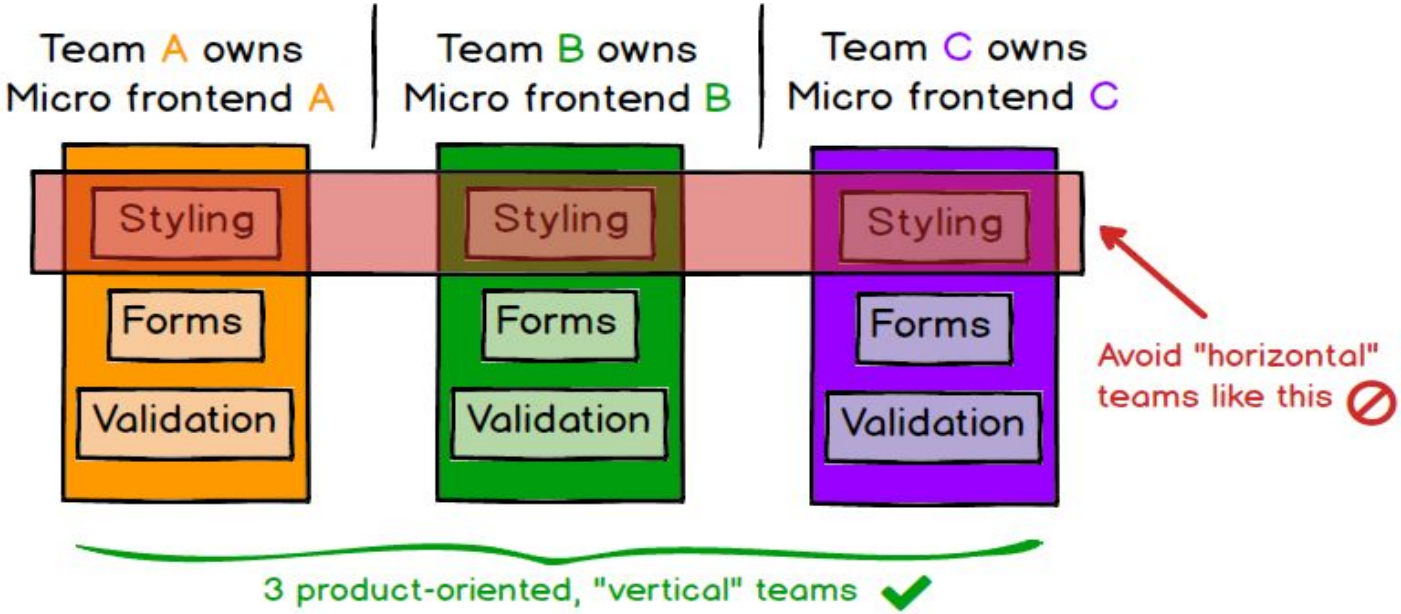
Program level

- Architects, Leads, Product Owners
- Business Analysts
- Quality Analysts

Team level

- Team Leads, Product owners
- Developers, Business Analysts
- Quality Analysts

Independent Deployment is possible only using vertical teams



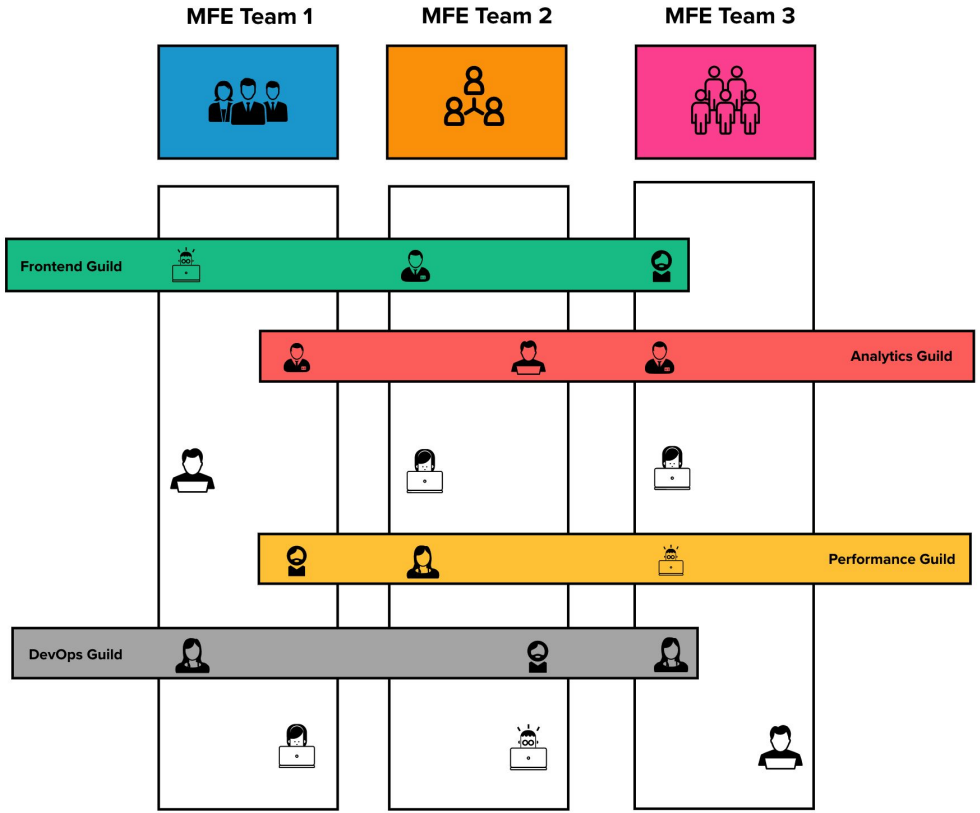
Source: https://martinfowler.com/articles/micro-frontends.html Author: Cam Jackson

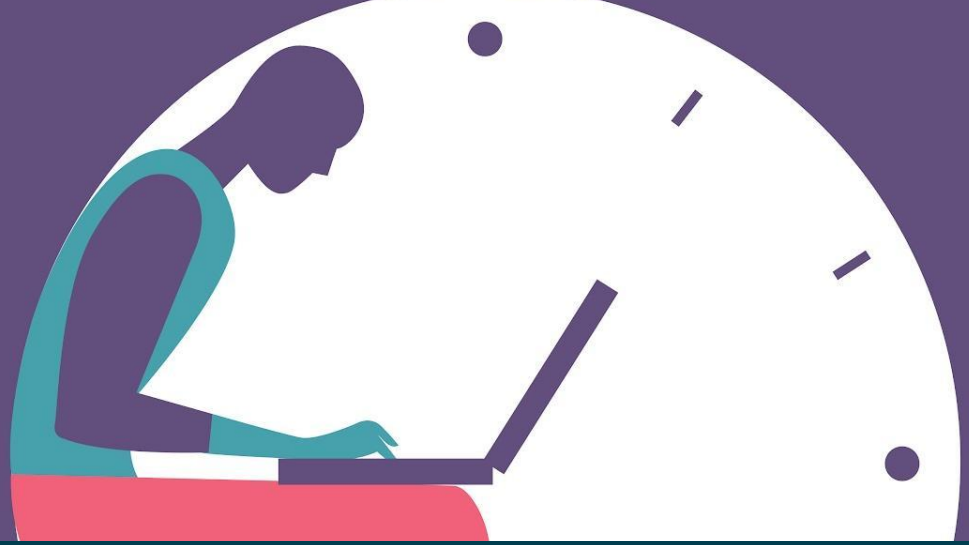
Shared guidelines

- All teams should speak the same vocabulary
- Agree on terms of responsibility and ownership
- Should adhere to global agreements and conventions
- Maintaining ADRs and documentation
- Maintain a global project scaffolding (but make it optional)

Team guilds

Forum for like minded people from different teams to share and discuss





To micro-frontend or not ?

Decision making factors for MFEs

It takes relatively longer time to:

- Provision and setup MFE Infrastructure
- Set up MFE codebases with shared dependencies
- Set cadence, workflow between teams, setup guilds
- Time and cost

Decision making factors for MFEs

Practical factors to consider:

- Project scale and timeline fairly large
- There is clear separation of sub domains
- Team size and distribution
- Migrating from a monolith

Alternate approach

You may not need to start with an MFE from day 1

1.

Start with a monolith

2.

Keep clear separation between sub domains for easy separation

3.

Slowly migrate to MFE architecture

Micro-frontends is a marathon. Not a sprint

© 2022 Thoughtworks

References

https://martinfowler.com/articles/micro-frontends.html

https://www.thoughtworks.com/en-in/about-us/events/webinars/microfrontend

https://www.youtube.com/watch?v=iZ-wIViaefc

https://livebook.manning.com/book/micro-frontends-in-action/micro-frontends-in-action/

https://micro-frontends.org/

Talk in Vodqa on Testing Strategy https://www.youtube.com/watch?v=1UEd5IJQ8G0

https://github.com/e-commerce-platform-with-microfrontends

Thank You

Vishal Sharma