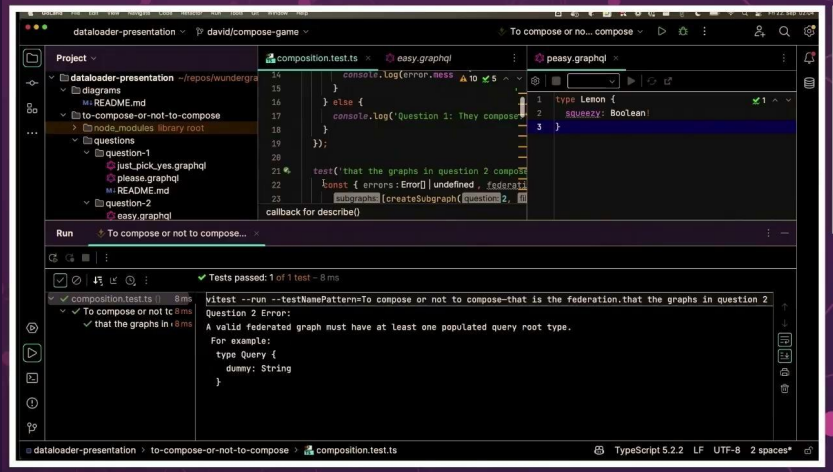
WUNDERGRAPH

Federation, Reversed: A Consumer-First Future with Fission

David Stutt
WunderGraph Founding Engineer

New conf; who dis?

GraphQL Conference 2023 “Will it Compose?”



GraphQLConf 2023

hosted by GraphQL Foundation

A valid federated graph

Federation in a nutshell

- Created and made public around 2019 by Apollo GraphQL.
- Many services federate to become one unified API interface.
- “Federation directives” describe how the composition algorithm should (try to) compose.

What we're talking about today

"Federation directives" describe how the composition algorithm should (try to) compose.

This means the unified API interface is not what's designed. It is at the mercy of the composition algorithm. Do the constituents (subgraphs) produce the desired result? Are they even compatible?

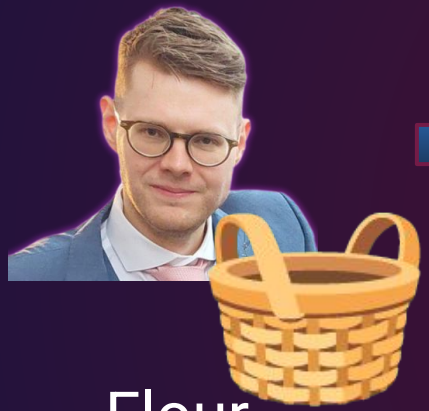
The “Cake Problem”

David, every cake maker in the entire UK has suddenly disappeared under suspicious circumstances. Can you bake my wedding cake?

Please can you bake my wedding cake?

The “Cake Problem”

The “Bottom Up” approach



?

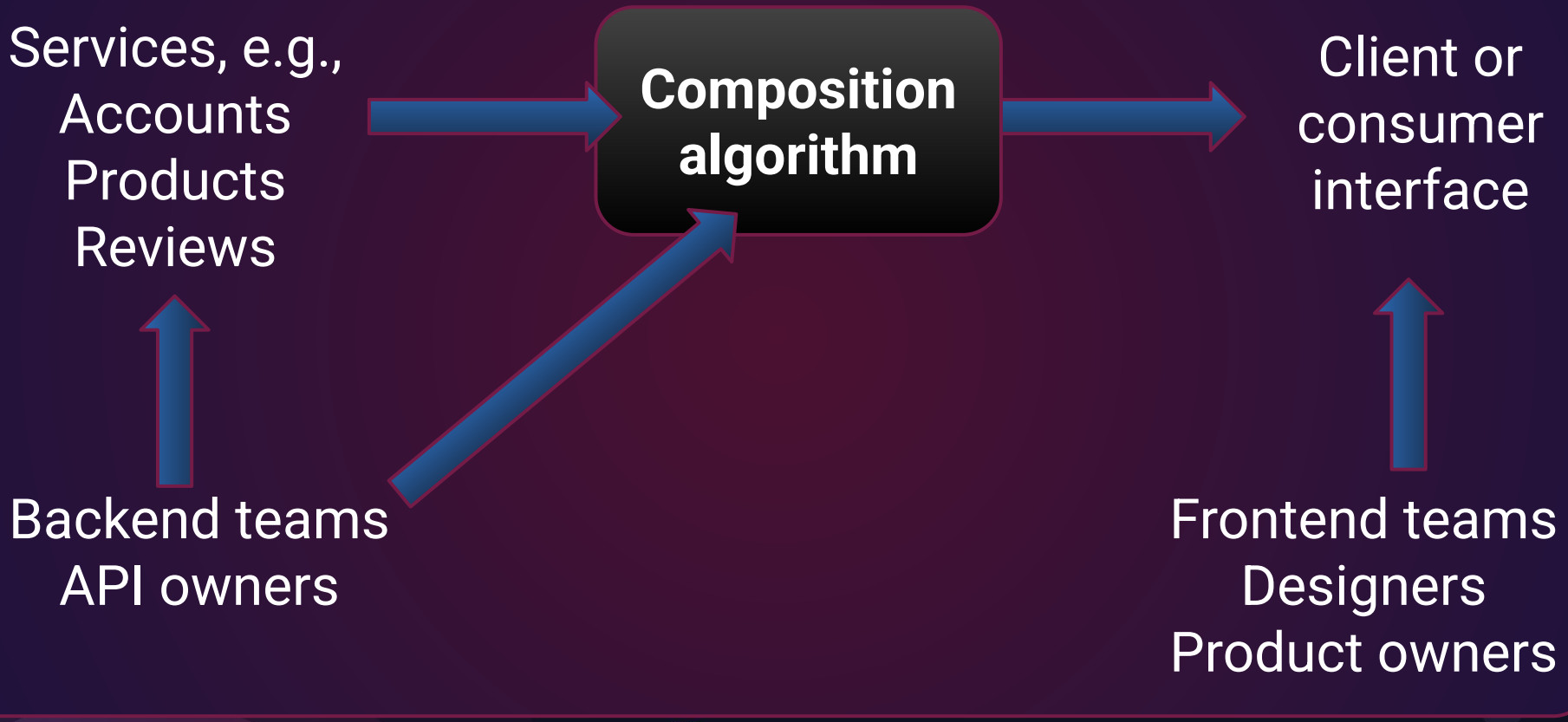
?

Flour
Sugar
Eggs
Butter

Flour
Sugar
Eggs
Butter

Cake?
Cookies?
Pancake?
Scone?
Brioche?

The “Cake Problem”



The “Cake Problem”

So, is there a better way to bake our cake?

Cake

225g Flour
225g Sugar
4 Eggs
225g Butter

The “Top Down” approach

- Start with the interface that is *actually* desired and will *actually* be consumed.
- The ability to involve non-technical colleagues and stakeholders.
- Limit how many people need to know how the “black box” works and the required threshold of that knowledge.

WUNDERGRAPH

Introducing Fission

Introducing “Fission”

- “Fission” is the algorithm that fissures, or breaks apart a federated GraphQL API interface into constituent services.
- The goal of Fission is to abstract the rules of Federation as much as possible.
- WunderGraph Hub is powered by Fission to provide powerful GraphQL API collaboration regardless of technical ability.

Why abstraction? The @external directive

- Marks a field that it cannot be resolved in “this” subgraph.
- Marks a field as *conditionally* resolvable in “this” subgraph (with @provides).
- Marks a field as appearing only to satisfy an interface.
- A relic of legacy (Version 1) syntax.

How Fission works: The @provides directive

- The @provides directive declares a field that is *usually* unresolvable in the subgraph *can* be resolved if (and only if) requested under a certain path.
- In Federation Version 2, the field being provided must not be unconditionally resolvable (i.e., it/an ancestor must be declared @external).

How Fission works: The @provides directive

- The goal is to produce a valid @provides field set (selection set).
- A valid field set contains a leaf field (Enum or Scalar) that is unresolvable in the providing subgraph or one of its selection set ancestors is unresolvable in the providing subgraph.

How Fission works: The @provides directive

- Fission produces a list of fields available across subgraphs belonging to the providing type from which the user can choose.
- New lists of fields are generated using the named type of the last selection.
- Eventually, the list produces a leaf if it’s either 1. undefined or @external in the subgraph itself; 2. one of its selection ancestors is undefined or @external in the subgraph.

How Fission works: The @provides directive

```
function getPotentialSelections(node: Node): Array<Node>:
var potentials: Array<Node> = [];
for field in node.fields:
    if field.isLeaf &&
   (!field.isExternal || !field.isAncestorExternal):
       continue;
  potentials.push(field);
return potentials;
```

How Fission works: The @provides directive

- From each each provided leaf node, walk up the field set “branch”.
- If a node is encountered that is not defined in the providing subgraph, add it (with the @external directive).
- Once at the root (the providing type), move on to the next “branch”.

How Fission works: The @provides directive

```
for field in fieldSetBranch:
    if subgraph.hasNode(field.typeName):
        continue;
    subgraph.addExternalNode(
        field.typeName
    );
```

Conclusion - Fission & Hub

- Allows colleagues of varying technical ability and understanding *to collaborate together*.
- Allows much of the complexity of Federation *to be abstracted away*, removing confusion or uncertainty.
- Allows you to produce the final consumed API *you actually want*.

Thank you for listening!

Questions?

LinkedIn: https://linkedin/in/daviditstutt

GitHub: https://github.com/aenimus

WunderGraph Hub: https://hub.wundergraph.com