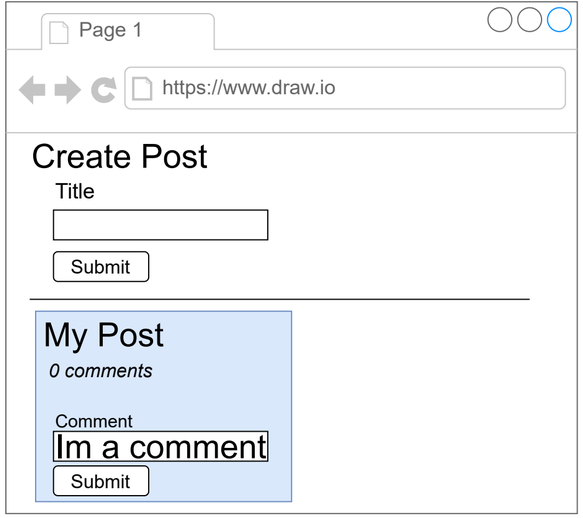
Few Notes

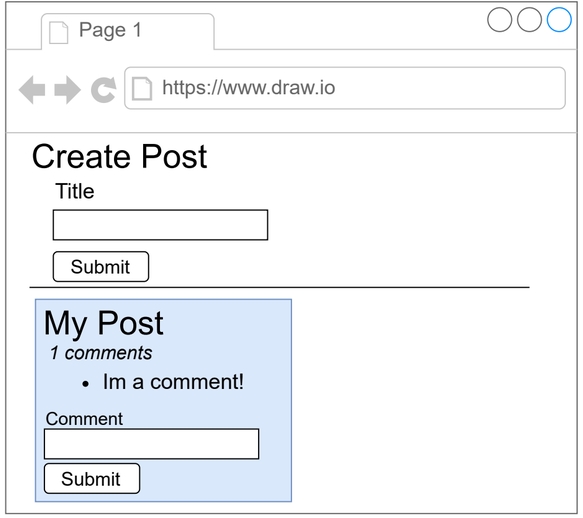
Goal #1 - Get a *taste* of a microservices architecture

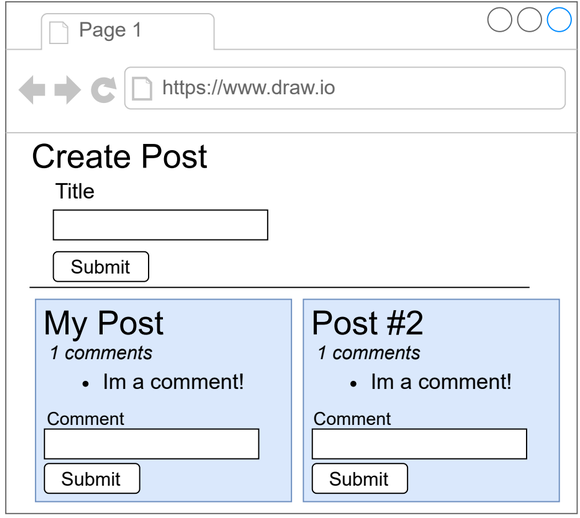
Goal #2 - Build as much as possible from scratch

Do not use this project as a template for future microservices stuff (we will build a better template later)


Page 1
https://www.draw.io
Create Post
Title
My POST
Submit

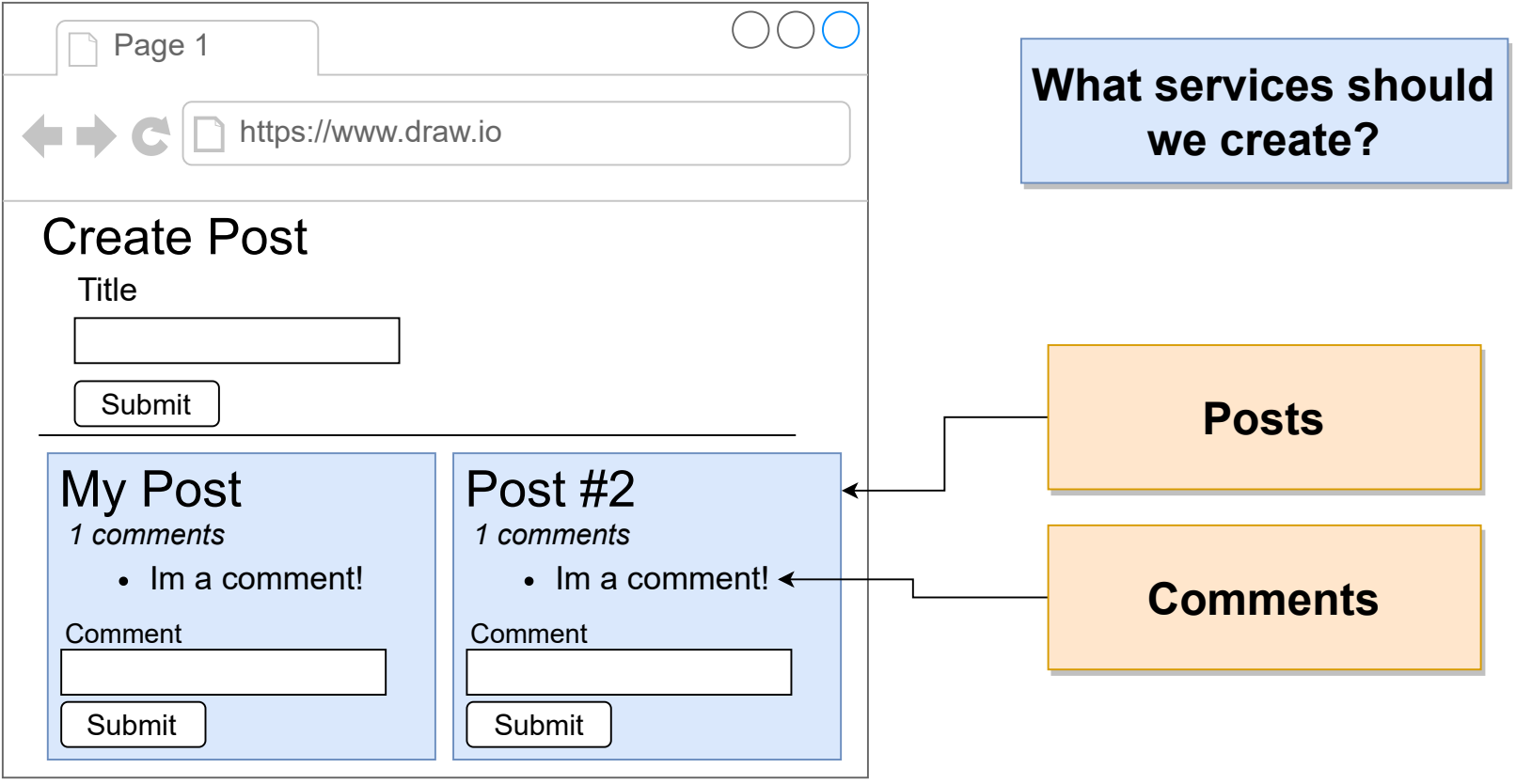

Page 1
https://www.draw.io
Create Post
Title
Submit
My Post
0 comments
Comment
Im a comment
Submit

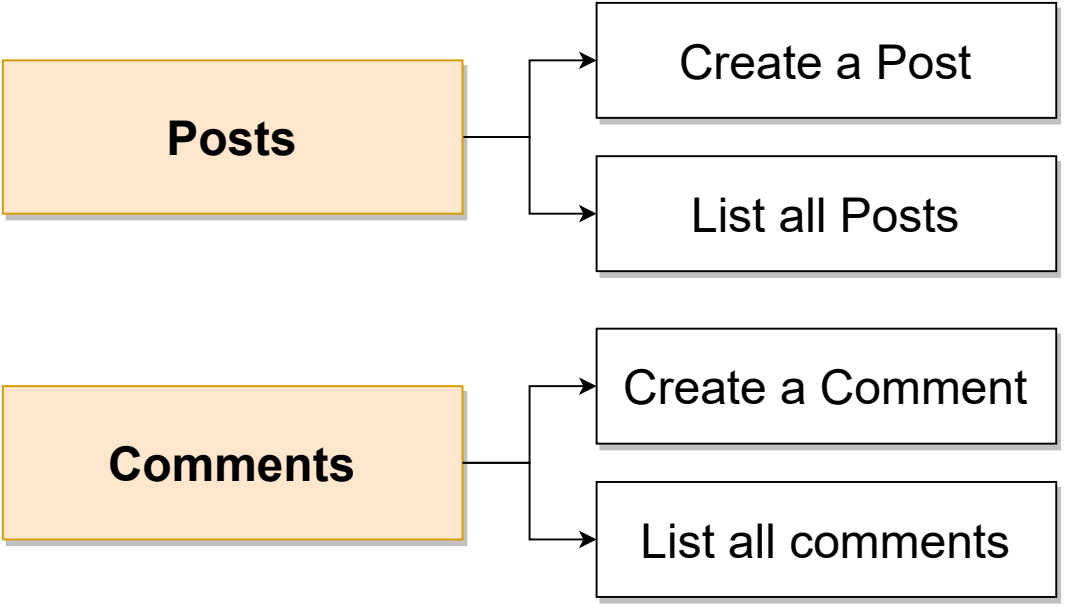

Page 1
https://www.draw.io
Create Post
Title
Submit
My Post
1 comments
Im a comment!
Comment
Submit

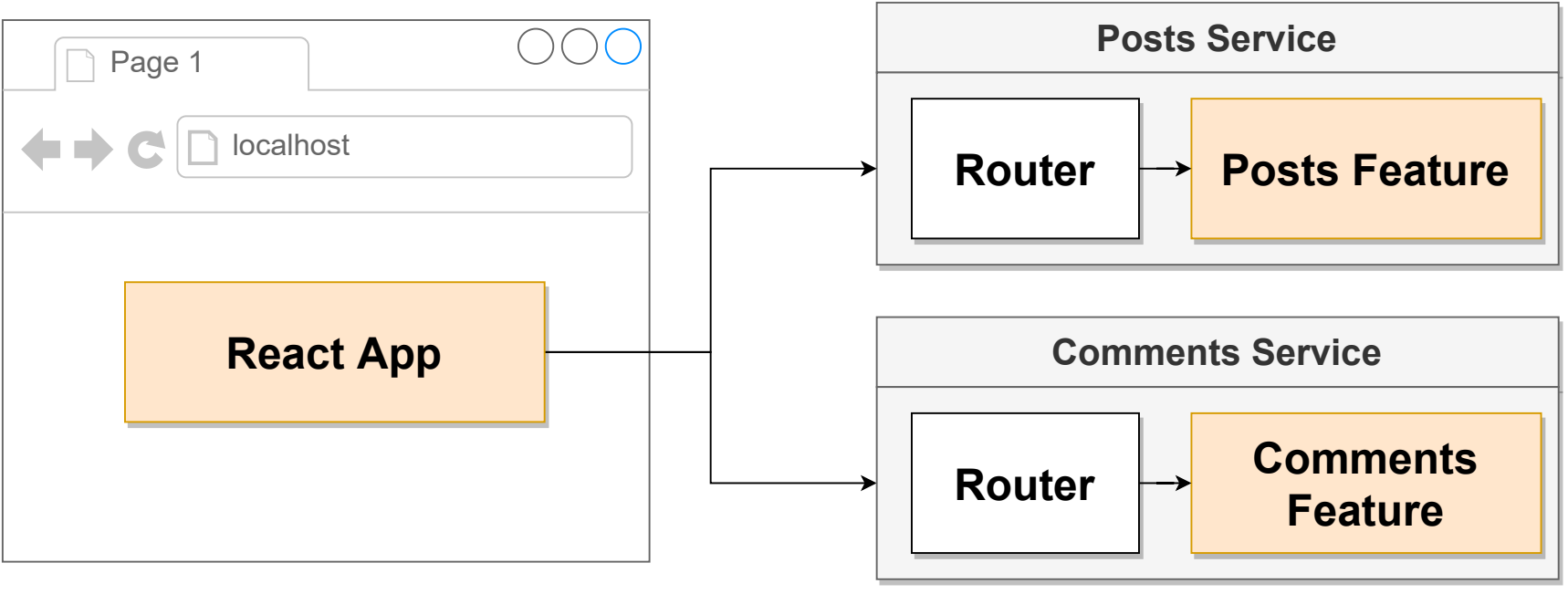

Page 1
https://www.draw.io
Create Post
Title
Submit
My Post
1 comments
Im a comment!
Comment
Submit
Post #2
1 comments
Im a comment!
Comment
Submit

What services should we create?

For now, we will create one separate service for each *resource* in our app


Page 1
https://www.draw.io
Create Post
Title
Submit
My Post
1 comments
Im a comment!
Comment
Submit
Post #2
1 comments
Im a comment!
Comment
Submit
What services should we create?
Posts
Comments


Posts
Create a Post
List all Posts
Comments
Create a Comment
List all comments


Page 1
localhost
React App
Posts Service
Router
Posts Feature
Comments Service
Router
Comments Feature

Initial App Setup

Generate a new React App using Create-React-App

Create an Express-based project for the **Posts** Service

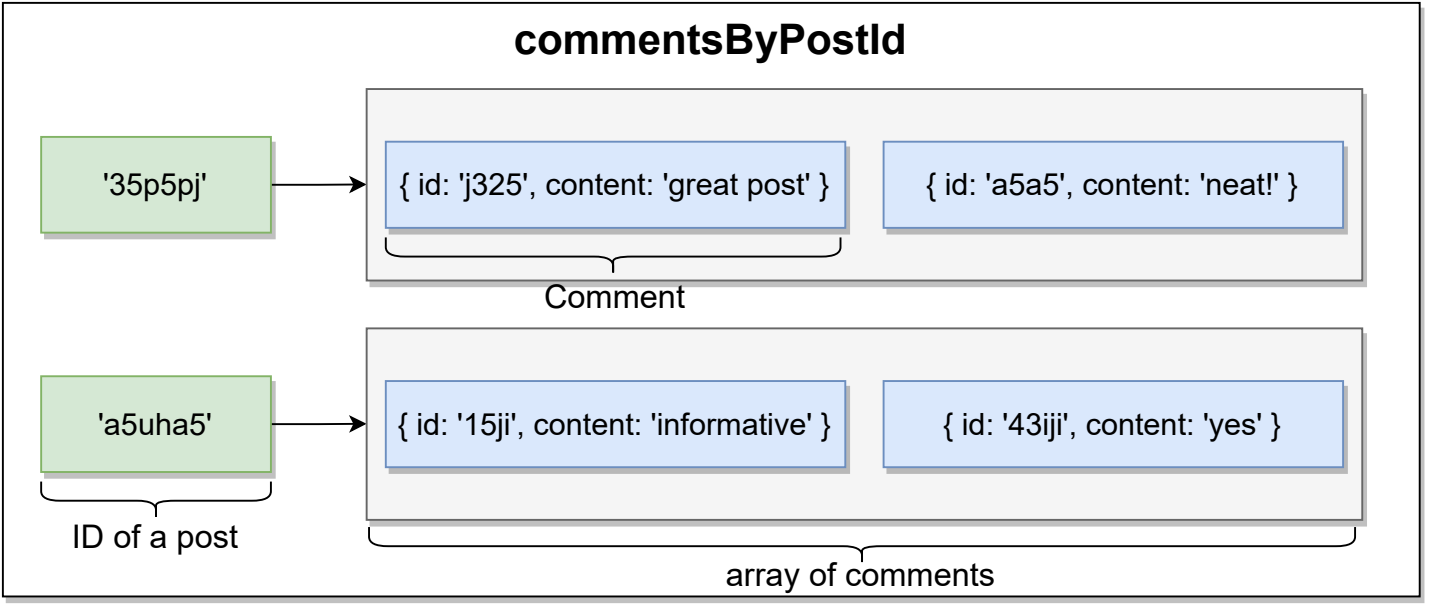
Create an Express-based project for the **Comments** Service

Posts Service

Path	Method	Body?	Goal
/posts	POST	{ title: string }	Create a new post
/posts	GET	-	Retrieve all posts

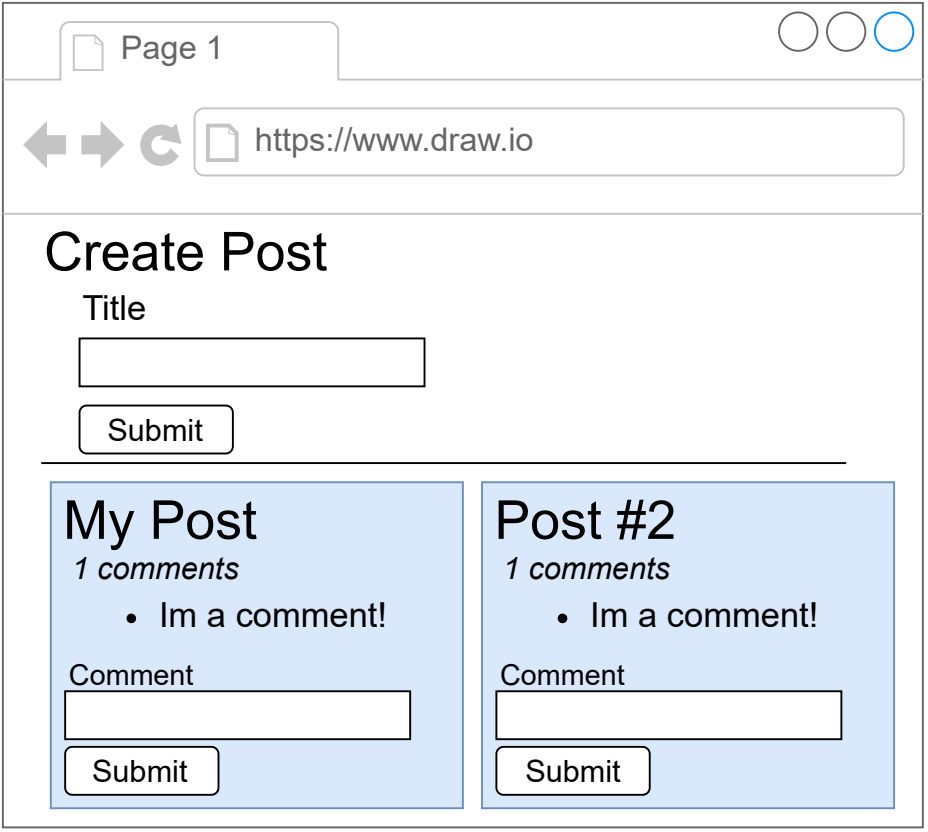
Comments Service

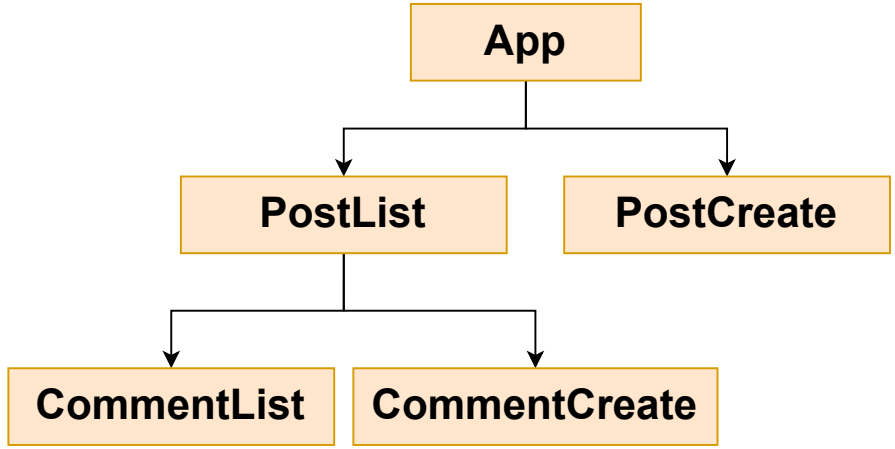
Path	Method	Body?	Goal
/posts/:id/comments	POST	{ content: string }	Create a comment associated with the given post ID
/posts/:id/comments	GET	-	Retrieve all comments associated with the given post ID

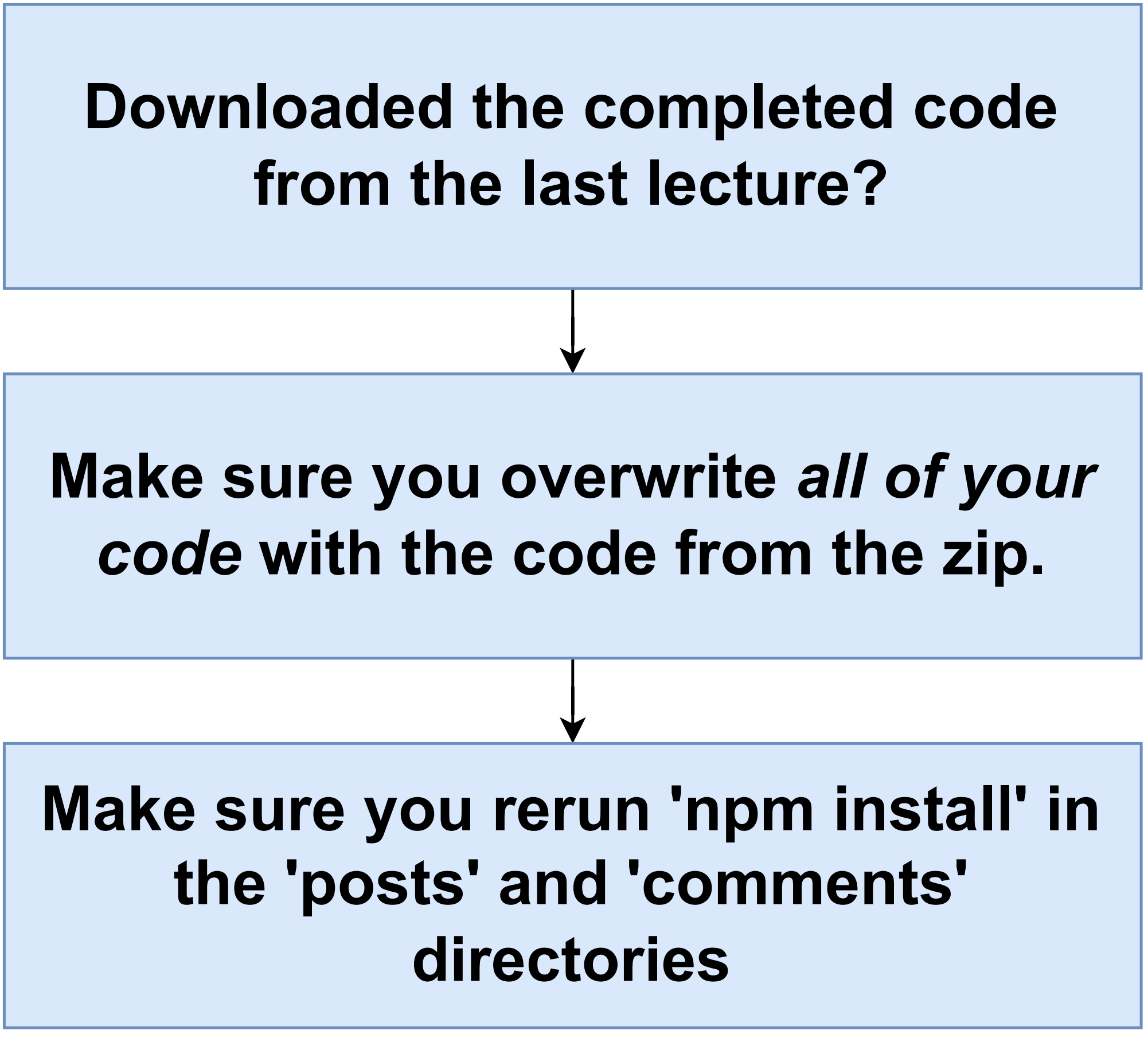

commentsByPostId
'35p5pj'
{ id: 'j325', content: 'great post' }
{ id: 'a5a5', content: 'neat!' }
Comment
'a5uha5'
{ id: '15ji', content: 'informative' }
{ id: '43iji', content: 'yes' }
ID of a post
array of comments

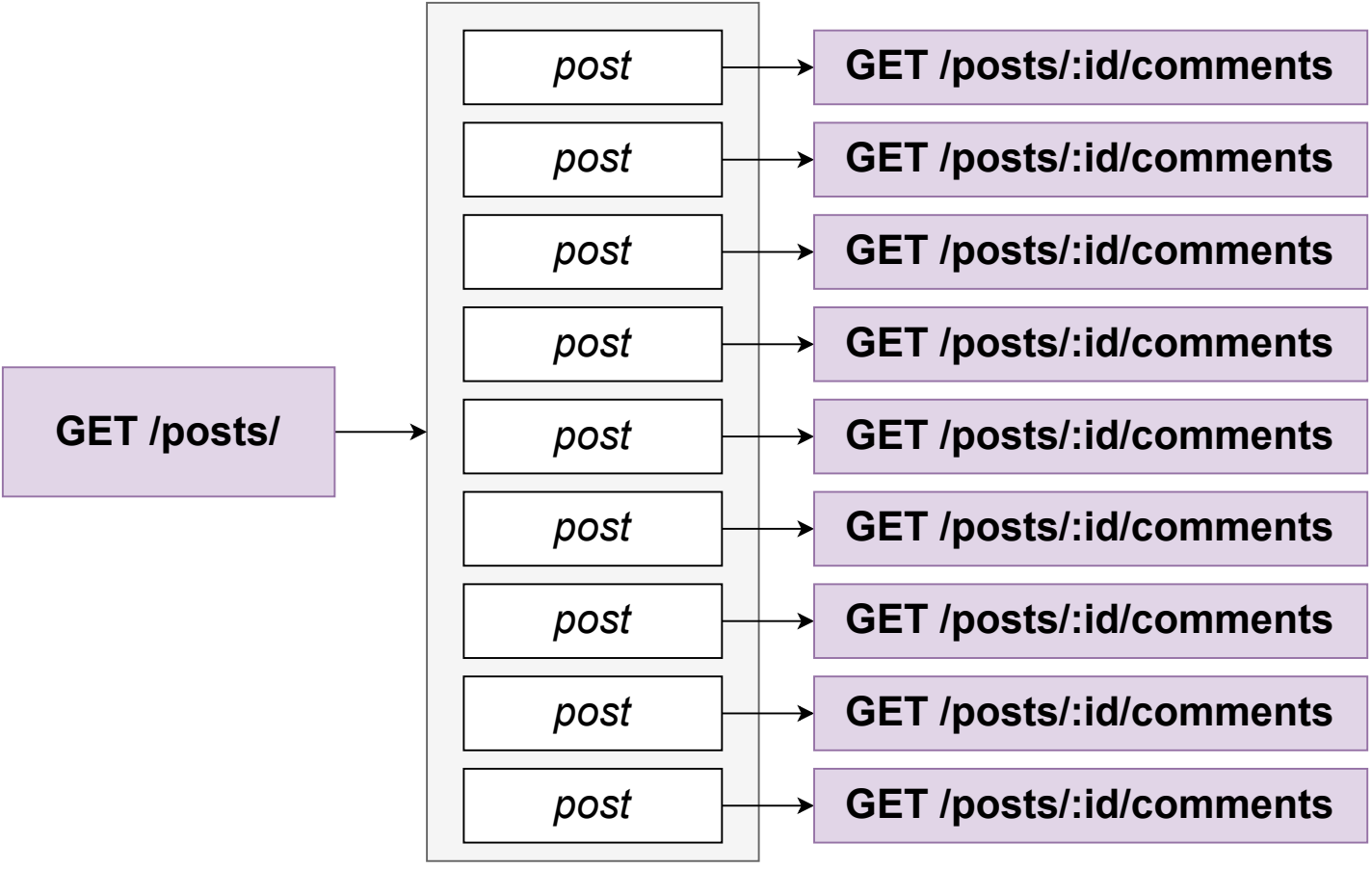
Note

If you don't know React, or don't want to work with it, no problem! Skip ahead a few videos. Completed React code is provided in a zip file

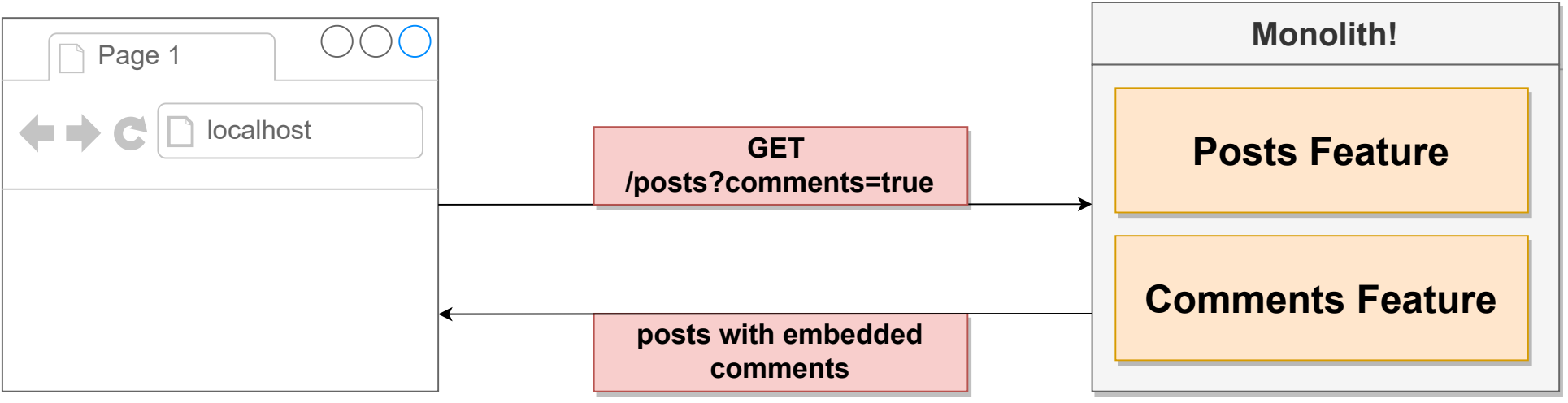

Page 1
https://www.draw.io
Create Post
Title
Submit
My Post
1 comments
Im a comment!
Comment
Submit
Post #2
1 comments
Im a comment!
Comment
Submit


App
PostList
PostCreate
CommentList
CommentCreate

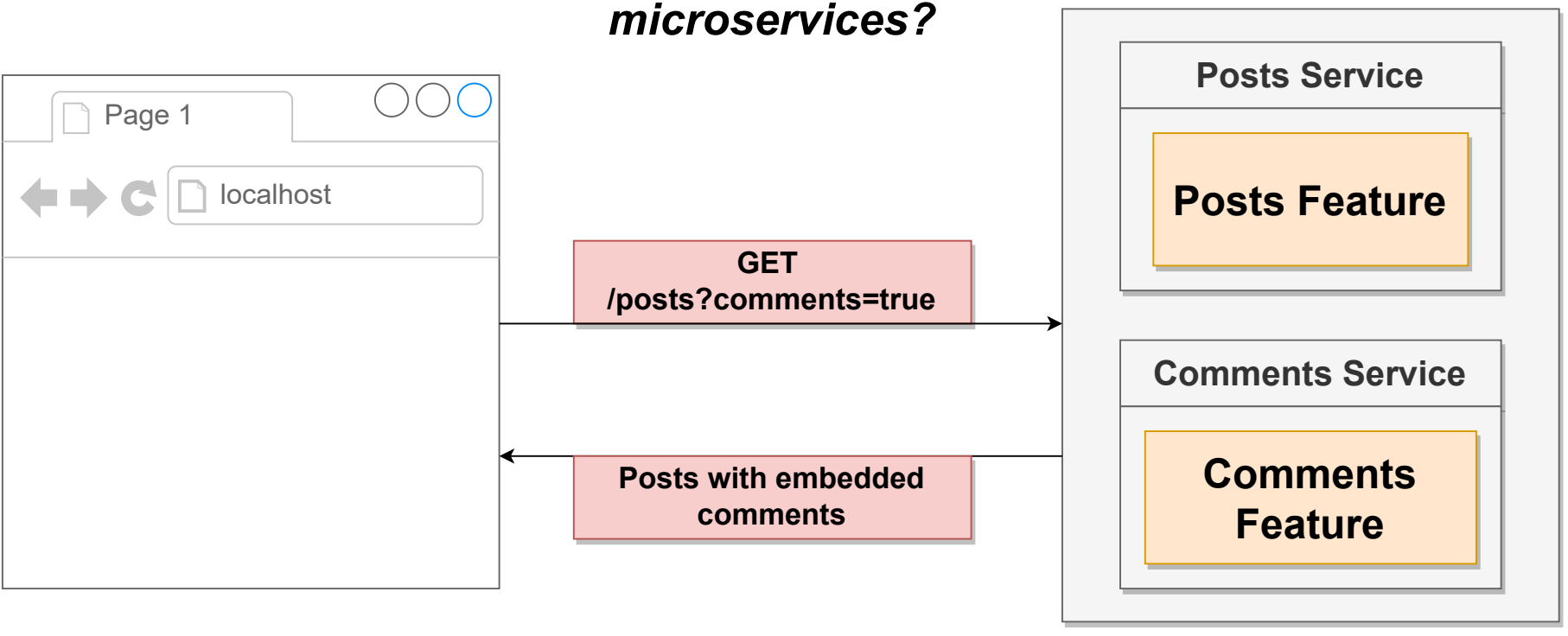

Downloaded the completed code from the last lecture?
Make sure you overwrite *all of your code* with the code from the zip.
Make sure you rerun 'npm install' in the 'posts' and 'comments' directories

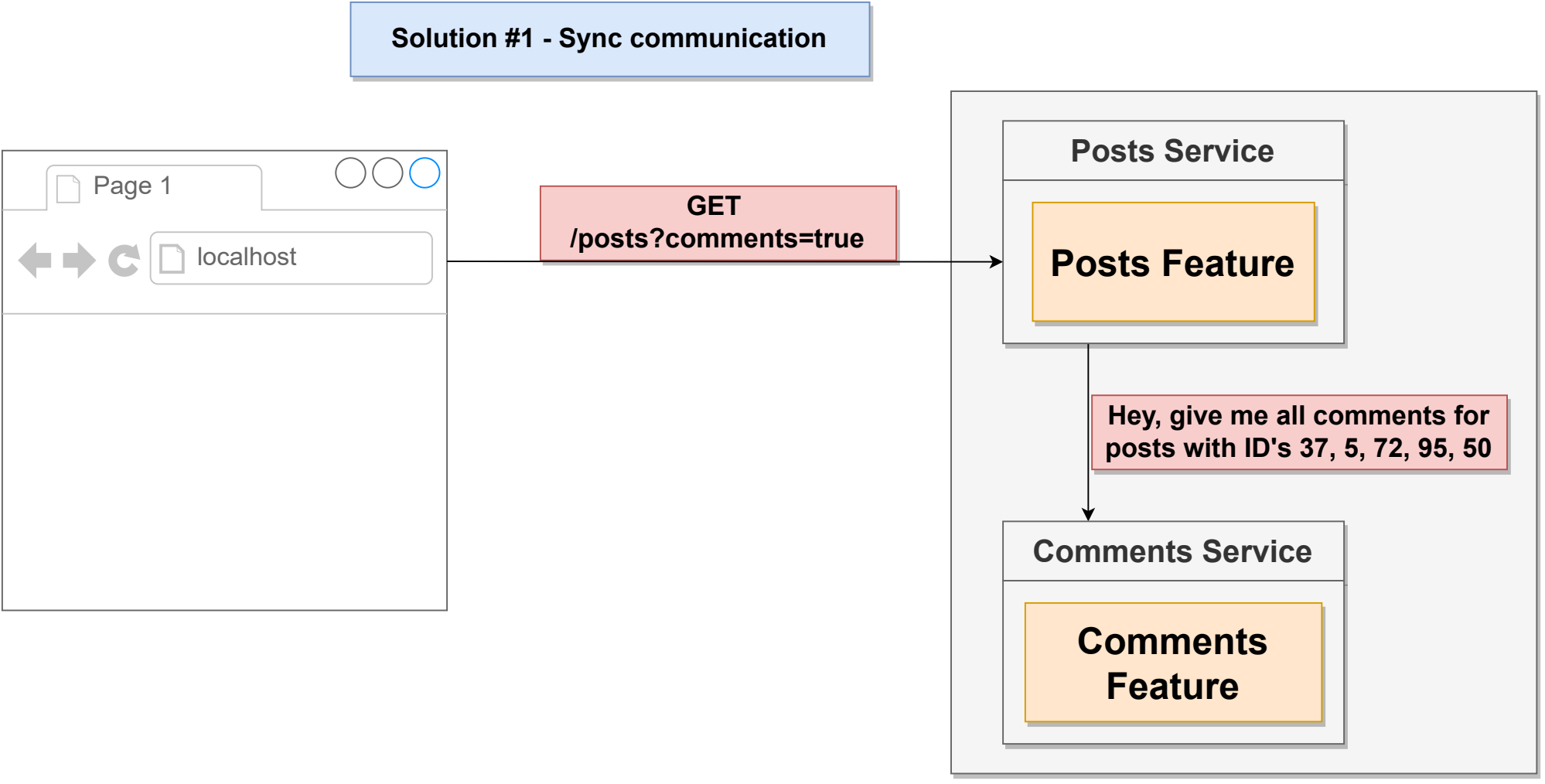

GET /posts/
post
post
post
post
post
post
post
post
post
GET /posts/:id/comments
GET /posts/:id/comments
GET /posts/:id/comments
GET /posts/:id/comments
GET /posts/:id/comments
GET /posts/:id/comments
GET /posts/:id/comments
GET /posts/:id/comments
GET /posts/:id/comments

Easily solved with monoliths!



How to solve with microservices?




Solution #1 - Sync communication
Page 1
localhost
GET
/posts?comments=true
Posts Service
Posts Feature
Hey, give me all comments for posts with ID's 37, 5, 72, 95, 50
Comments Service
Comments Feature

Notes on Sync Communication

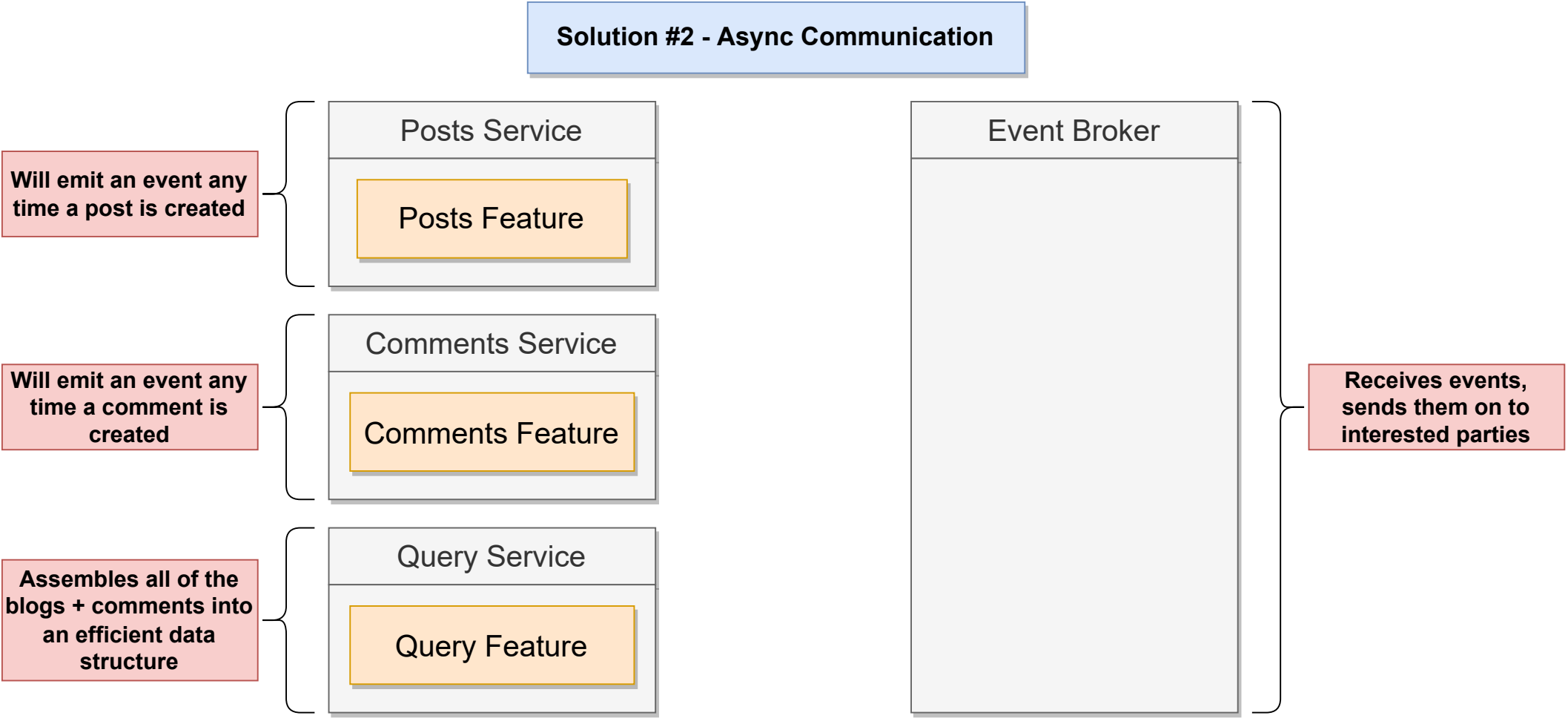
Conceptually easy to understand!

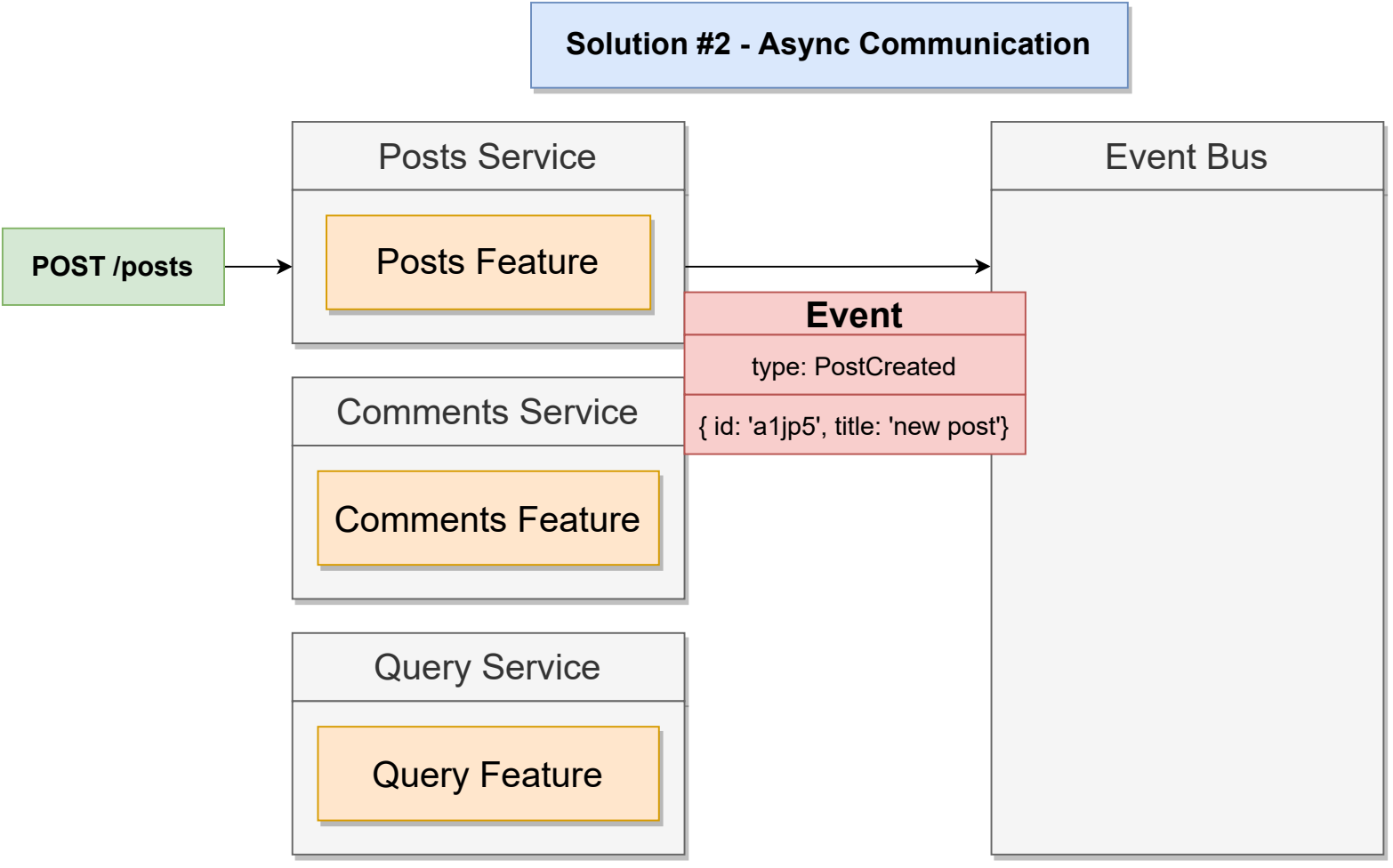
Introduces a dependency between services

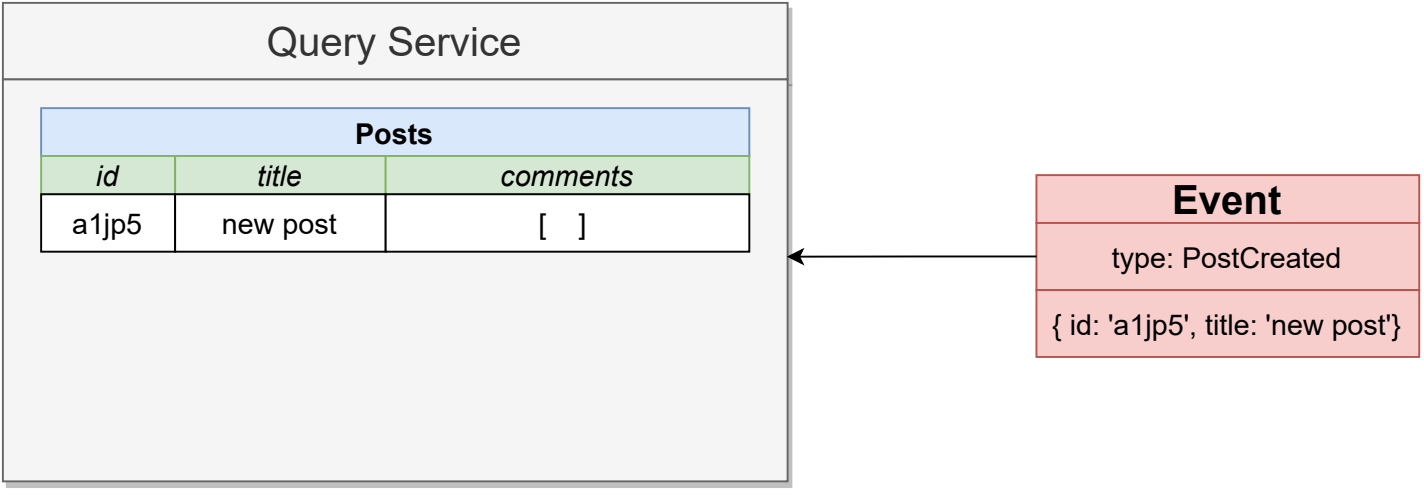
If any inter-service request fails, the overall request fails

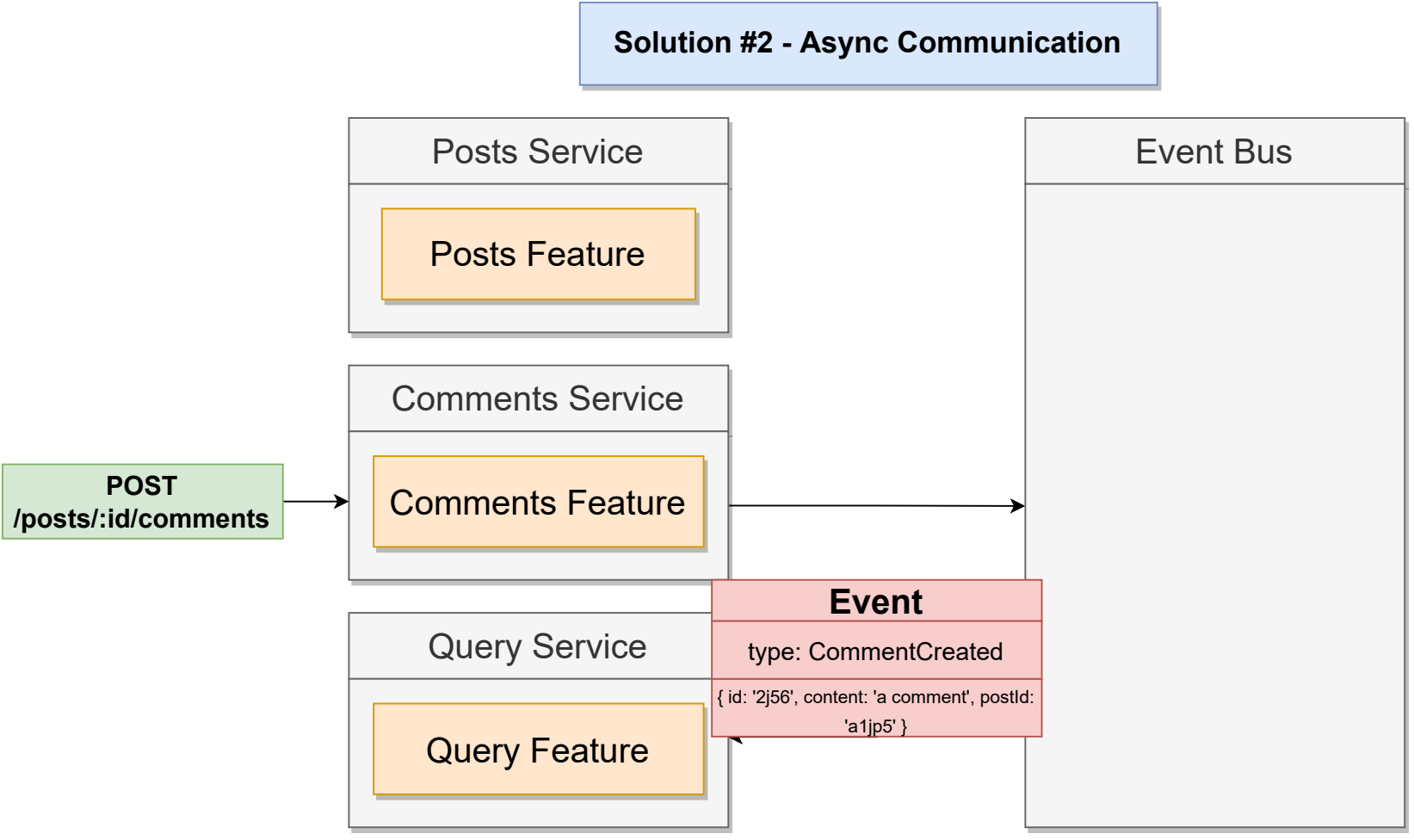
The entire request is only as fast as the slowest request

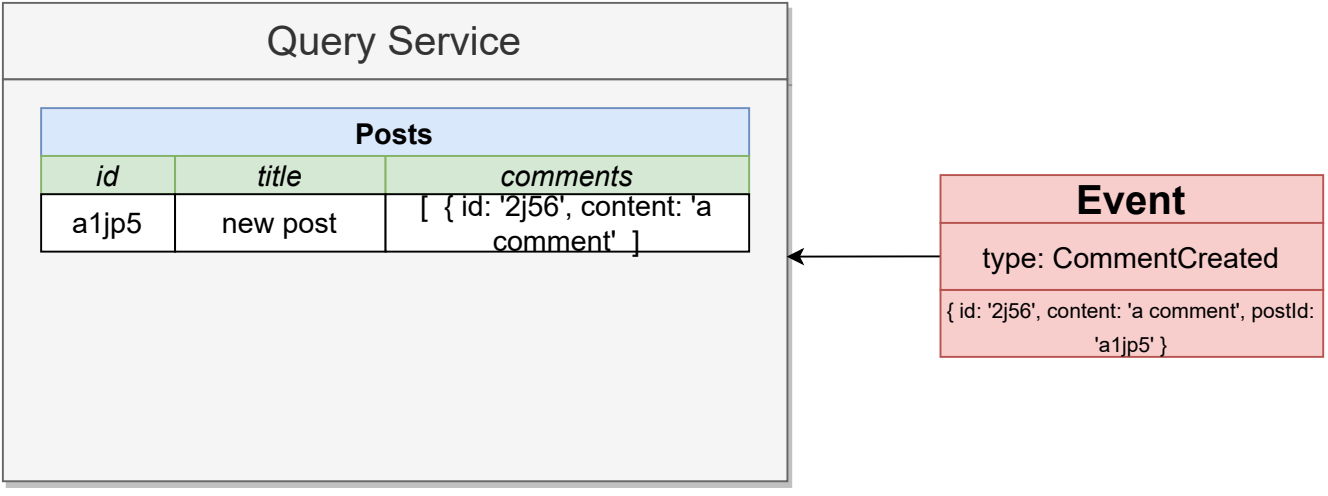
Can easily introduce webs of requests

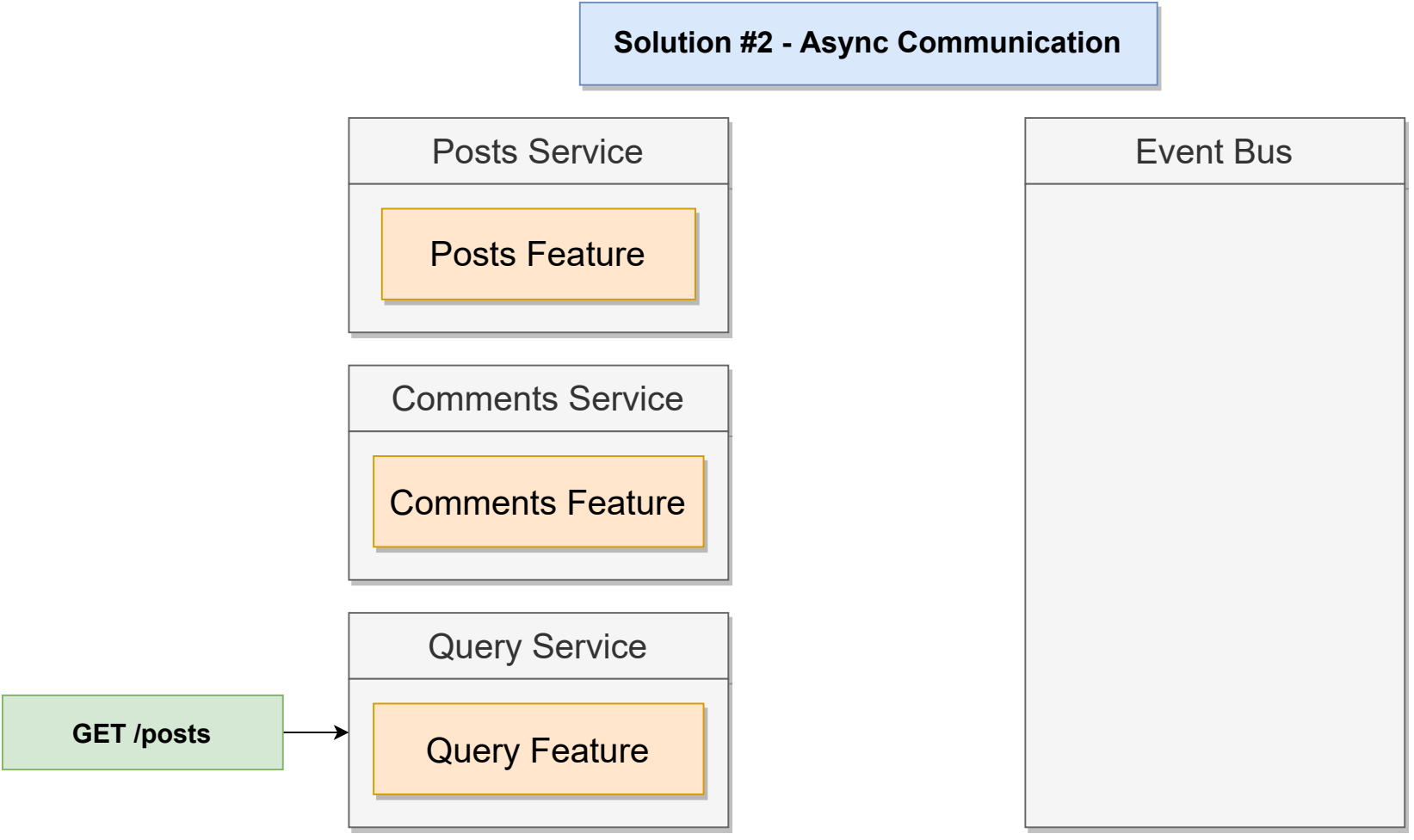

Solution #2 - Async Communication
Posts Service
Posts Feature
Will emit an event any time a post is created
Comments Service
Comments Feature
Will emit an event any time a comment is created
Query Service
Query Feature
Assembles all of the blogs + comments into an efficient data structure
Event Broker
Receives events, sends them on to interested parties


Solution #2 - Async Communication
Posts Service
Posts Feature
Event Bus
POST /posts
Event
type: PostCreated
{ id: 'a1jp5', title: 'new post'}
Comments Service
Comments Feature
Query Service
Query Feature


Query Service
Posts
id
title
comments
a1jp5
new post
[ ]
Event
type: PostCreated
{ id: 'a1jp5', title: 'new post'}


Solution #2 - Async Communication
Posts Service
Posts Feature
Event Bus
Comments Service
POST
/posts/:id/comments
Comments Feature
Event
type: CommentCreated
{ id: '2j56', content: 'a comment', postId: 'a1jp5' }
Query Service
Query Feature


Query Service
Posts
id
title
comments
a1jp5
new post
[{ id: '2j56', content: 'a comment' }]
Event
type: CommentCreated
{ id: '2j56', content: 'a comment', postId: 'a1jp5' }


Solution #2 - Async Communication
Posts Service
Posts Feature
Comments Service
Comments Feature
Query Service
Query Feature
GET /posts
Event Bus

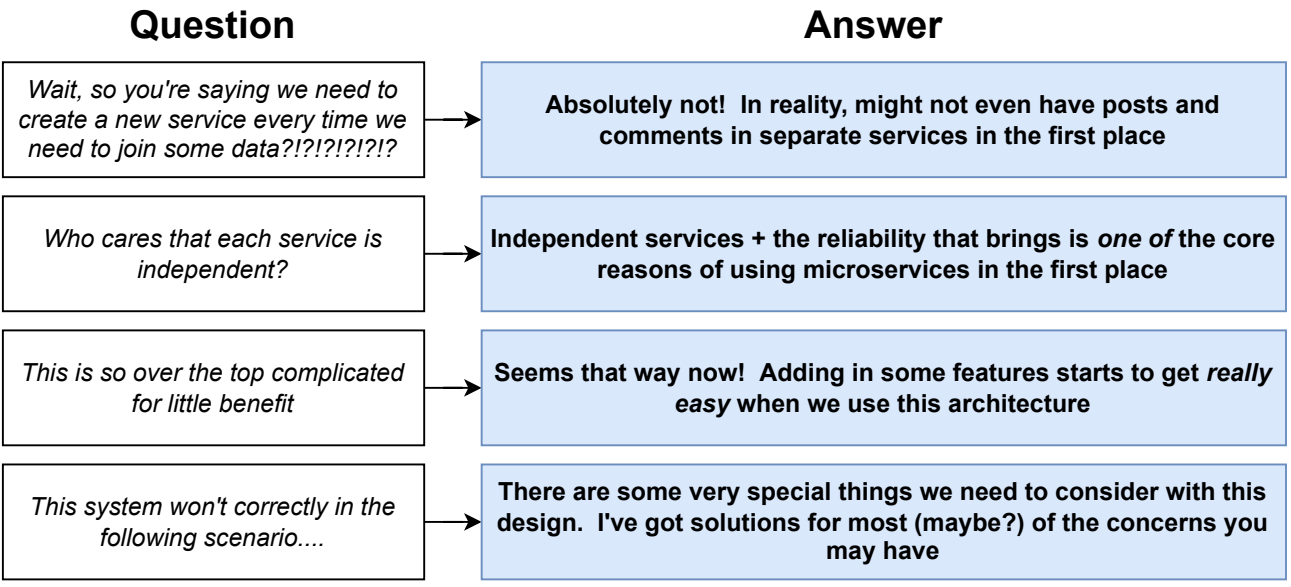
Notes on Async Communication

Query Service has zero dependencies on other services!

Query Service will be extremely fast!

Data duplication.

Harder to understand


Question
Answer
Wait, so you're saying we need to create a new service every time we need to join some data?!?!?!?!?!?
Absolutely not! In reality, might not even have posts and comments in separate services in the first place
Who cares that each service is independent?
Independent services + the reliability that brings is one of the core reasons of using microservices in the first place
This is so over the top complicated for little benefit
Seems that way now! Adding in some features starts to get really easy when we use this architecture
This system won't correctly in the following scenario....
There are some very special things we need to consider with this design. I've got solutions for most (maybe?) of the concerns you may have