Conquering Micro-services with GraphQL

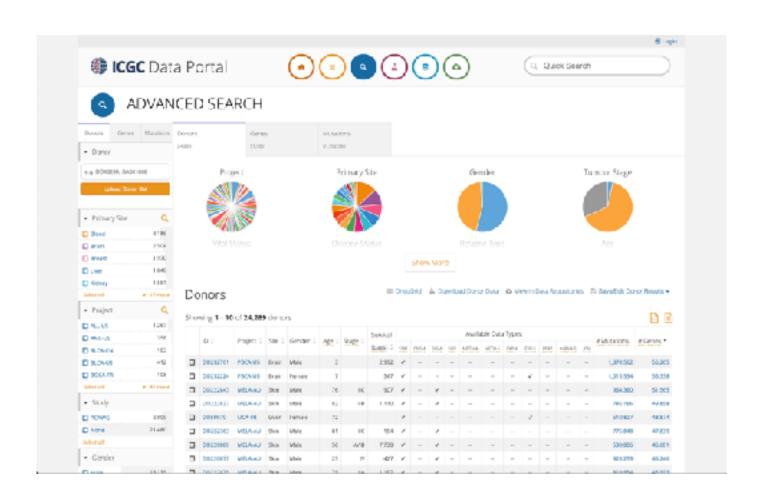
Our plan for Argo

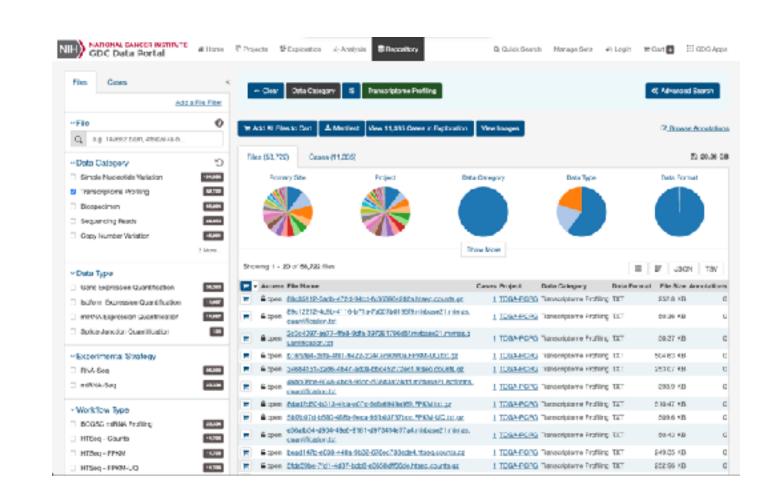


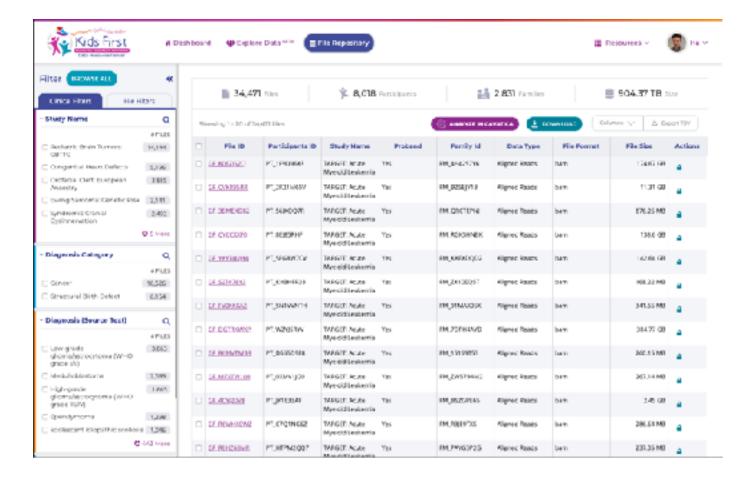






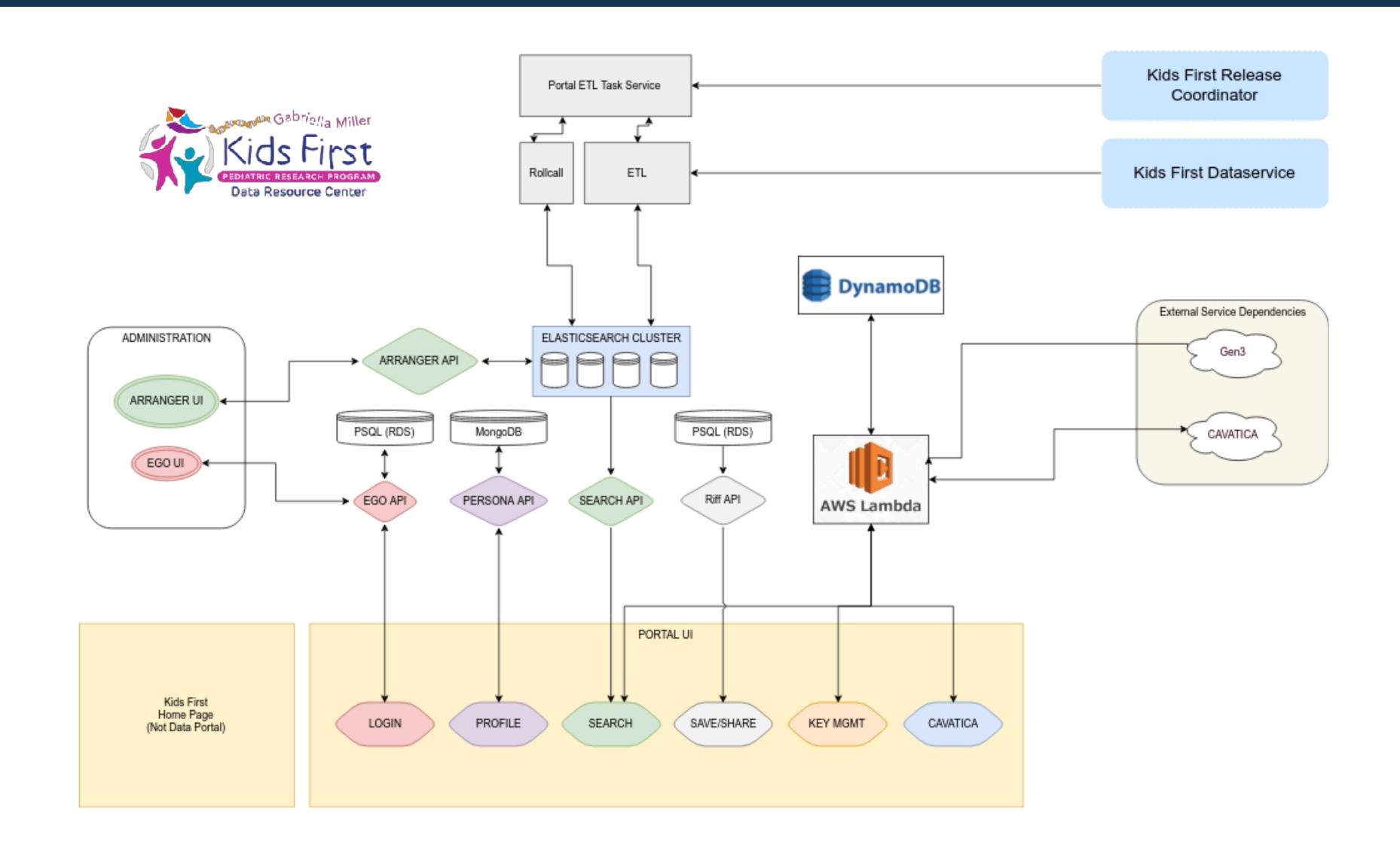




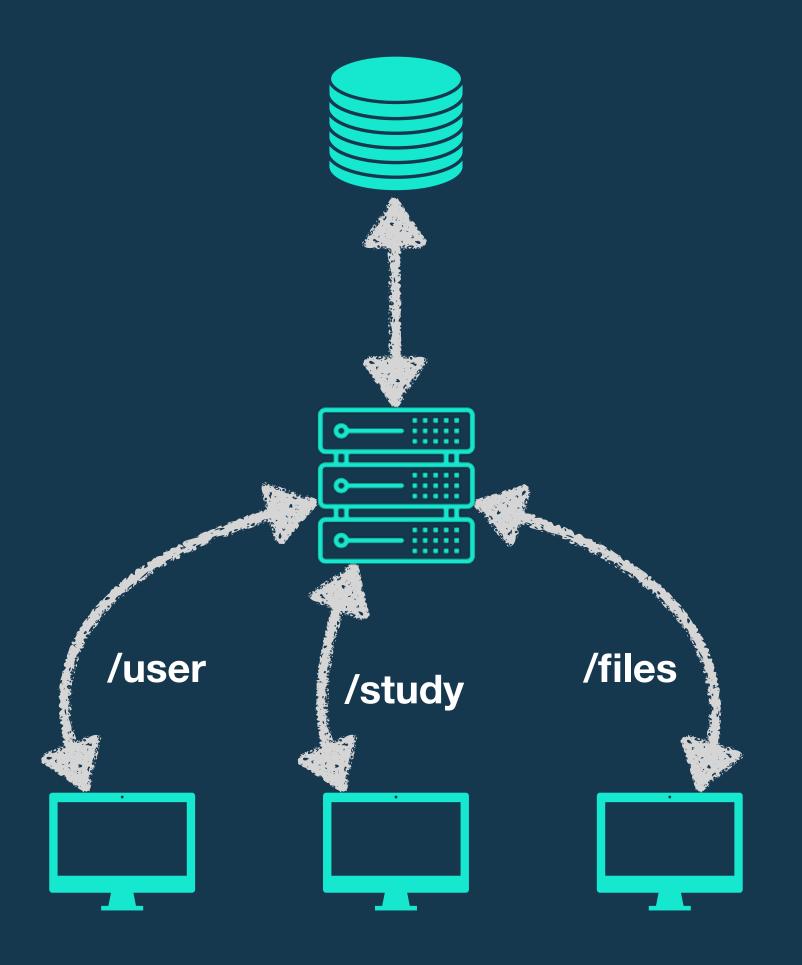


and more...

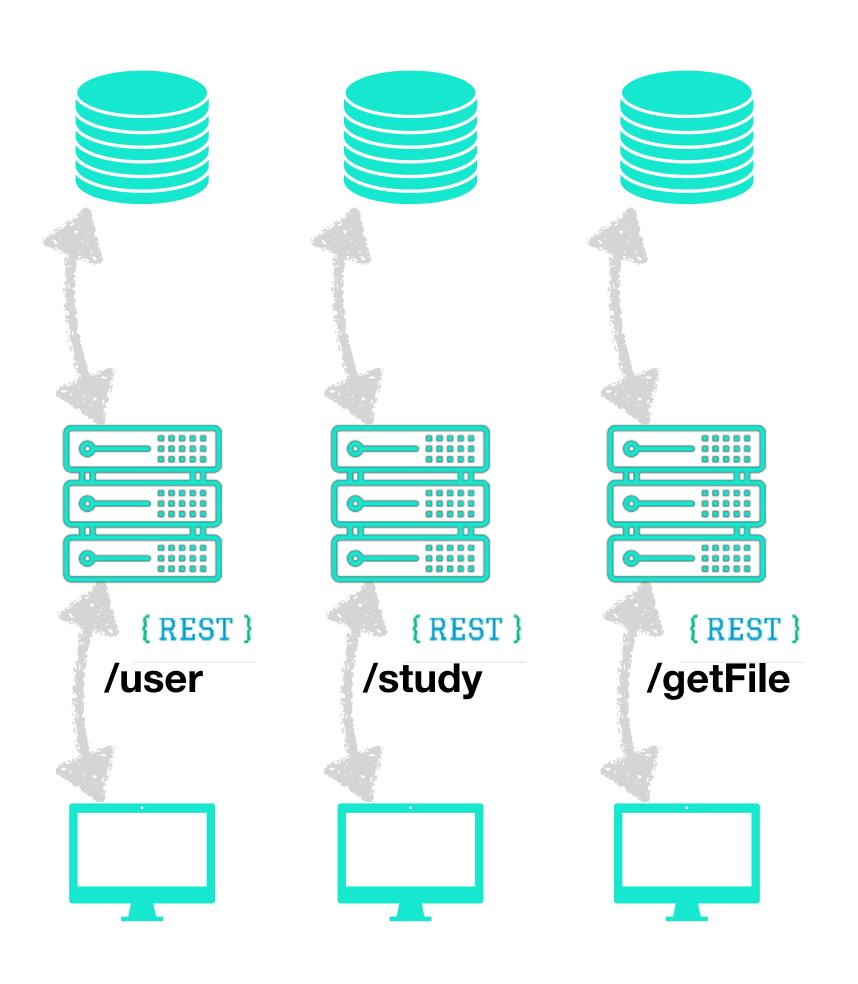
Behind the scene



Monolithic System



Micro-...what?

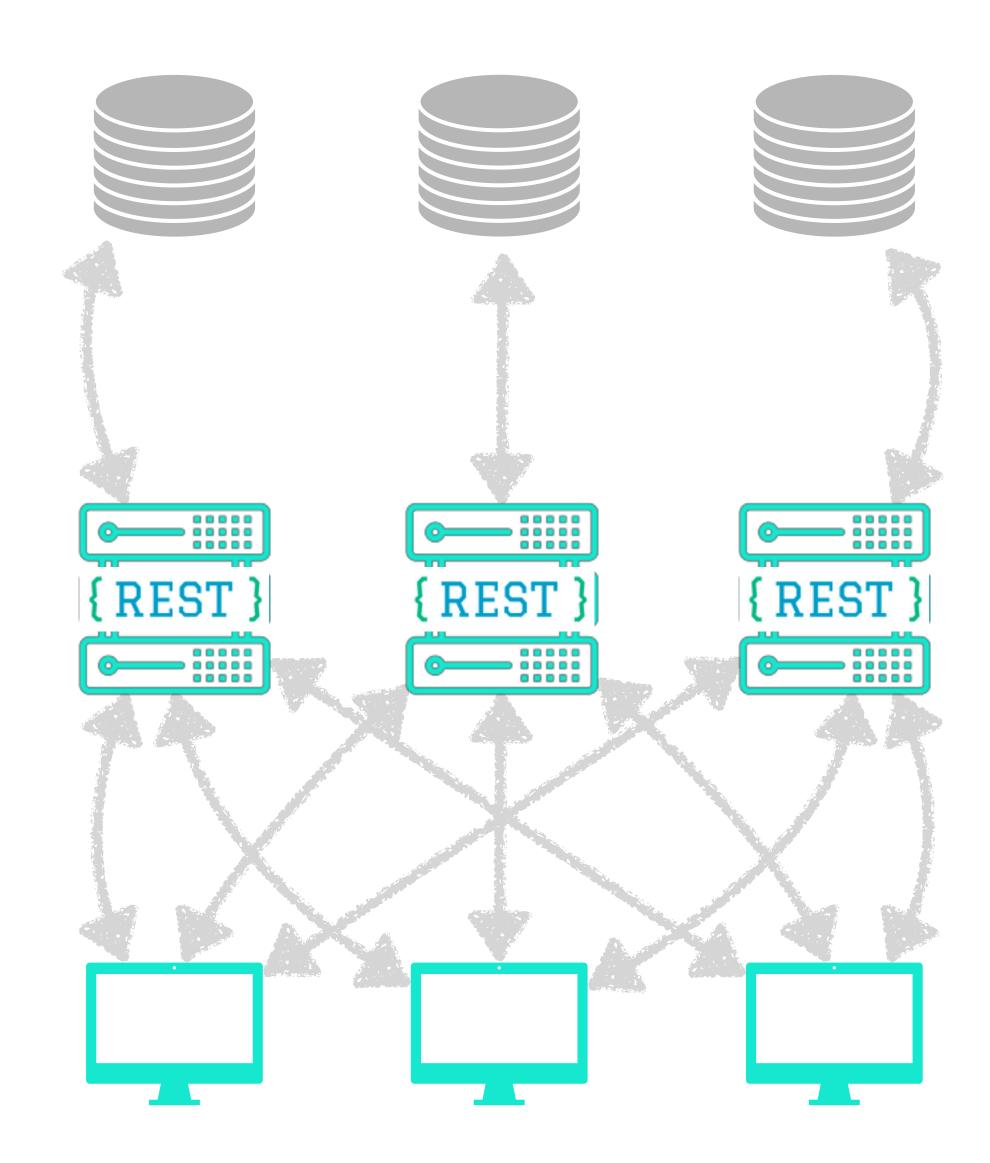


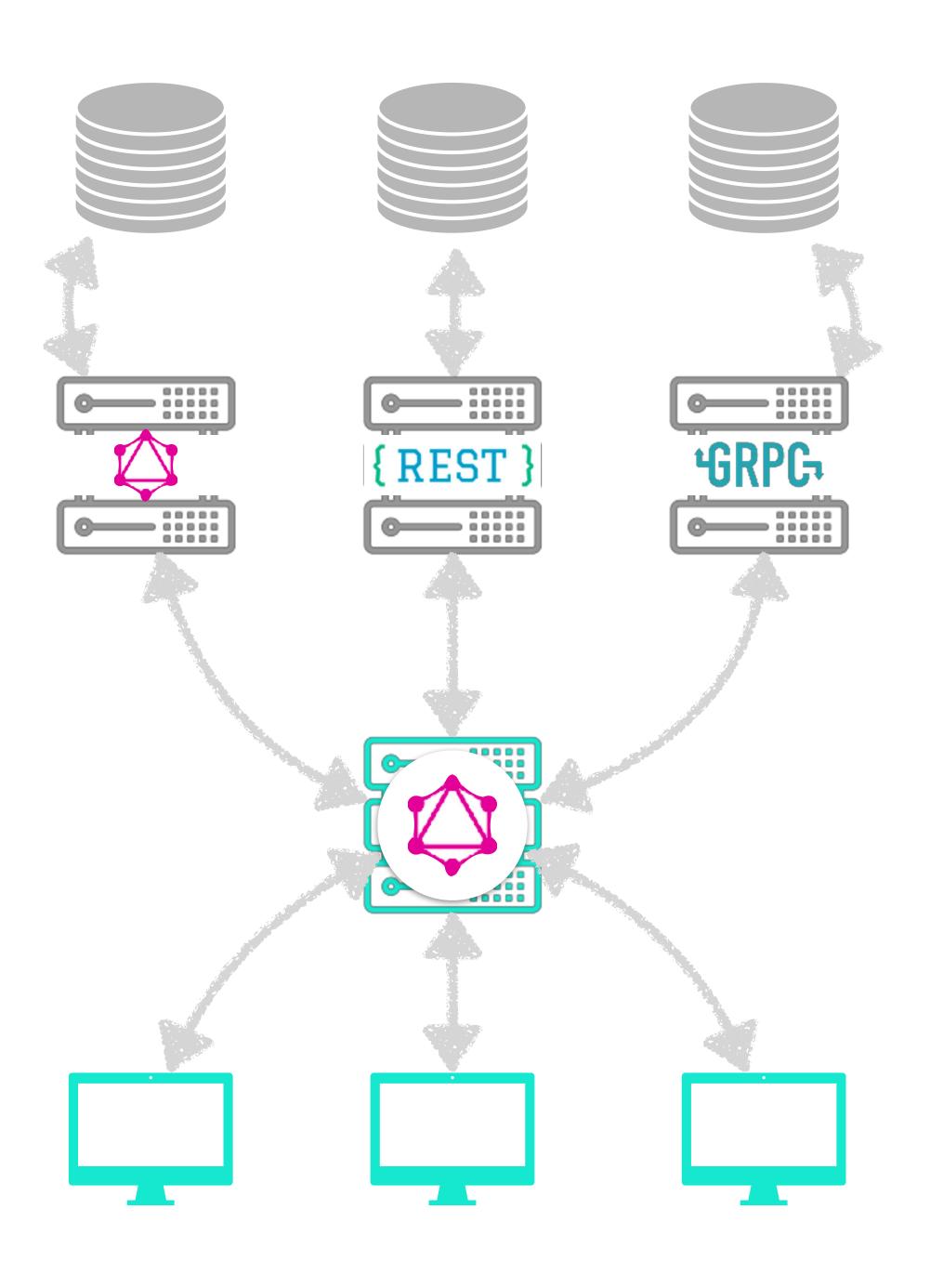
Micro-services System

- Functionally decoupled
- Independently deployable
- Independently scalable

Some challenge

- No place to handle cross-cutting concerns
- Front-end handles heavy business logic
- Inefficiencies with REST:
 - Over & under-fetching
 - Lack of API schema -> manual documentation





The GraphQL Solution

- •An aggregation layer to handle cross-cutting concern
- ·Why GraphQL?
 - Explorable schema —> selfdocumenting
 - Schema Stitching:
 - •Modelling underlying services as an independent schemas
 - "Stitch" the schemas together as nodes in a graph
 - Network advantage: batched request



SONG

SCORE

Submission System

Validation

System

Data

Dictionary

Clinical SONG

Control

(DACO)

Queue

Manager

Workflow

Management

Dashboard

RDPC MEMBER

Manifest and

Transfer

Dashboard

RDPCs

RDPC Datastore

SONG-R1

SCORE-R1

Workflow

Centralized Services

ID Service

GENOMIC DATA

Submission

Portal Pages

Harmonization

Dashboard

SUBMITTERS

Submission

Dashboard

Programme

Dashboard

MIDDLE WARE

SITE

FRONTEND - ARGO

RDPC Detastore

SONG-R2

SCORE-R2

Workflow

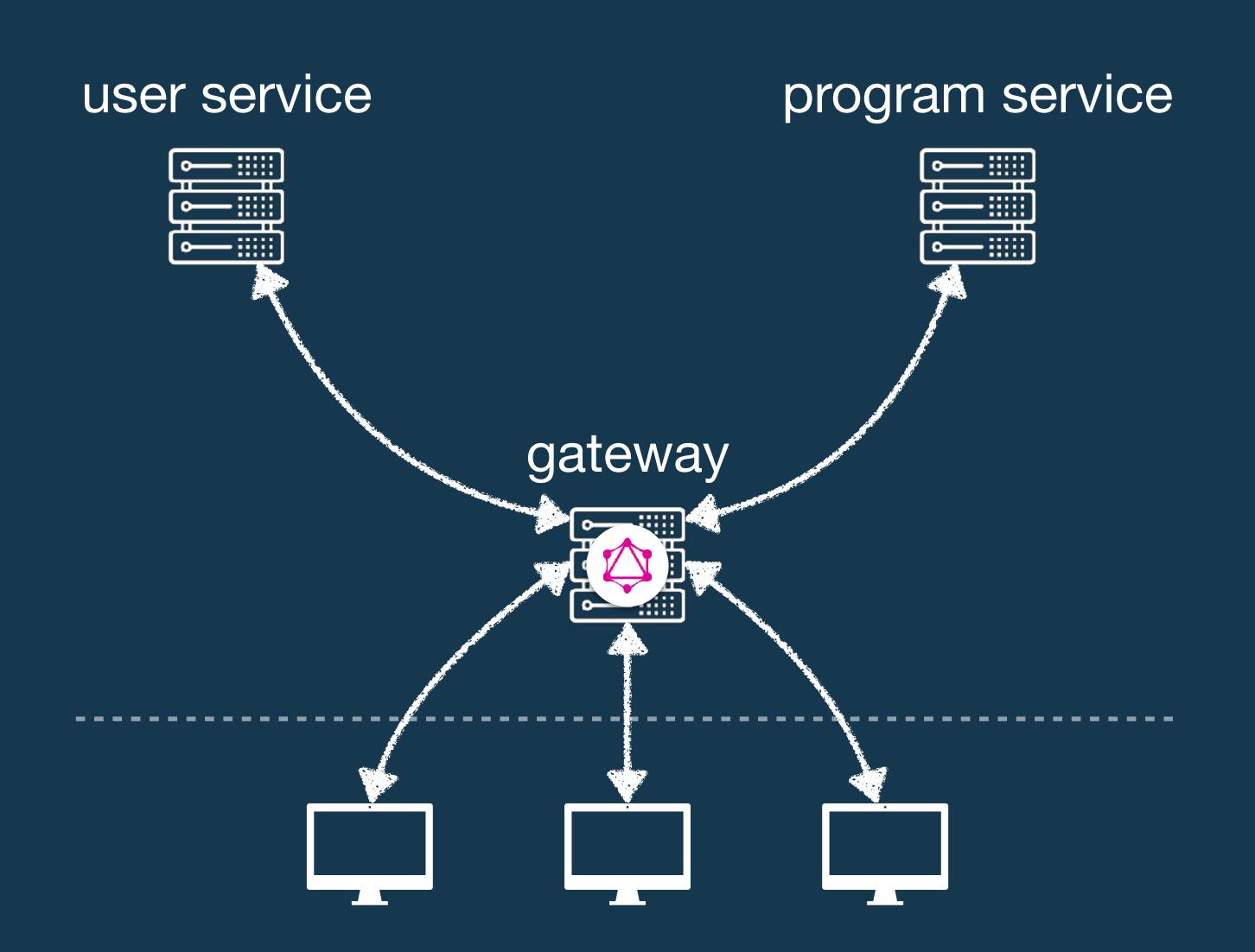
System

Location

Policy Engine

- CLINCAL DATA ----

demo



The wins

- Explorability:
 - Better developer on-boarding
 - Collaboration with non-technical team members
- Central place for integration test —> higher quality software
- Network performance

thank you