TRACK
Microservice - The First Decade

SESSION

The Human Side of Airbnb's Microservice Architecture

Jessica Tai
Tech Lead Manager on the Users Platform Team @Airbnb

Jessica Tai discusses lessons learned by Airbnb from its migration to microservices, covering cross-team collaboration strategies, designing observability access control, and planning for unified APIs.

Architecture evolution

Migration hits and misses

What's next

Architecture evolution

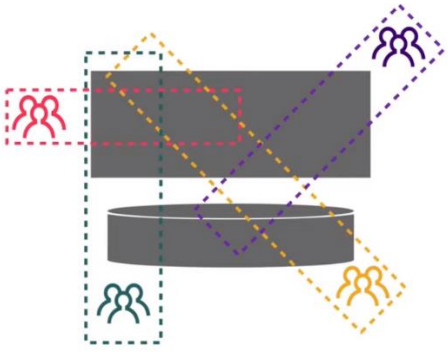
Monolith
2008 - 2017

More full-stack engineers

Features completed within teams, less dependencies

Confusing team ownership + unowned code

Monolith
2008 - 2017

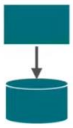


Slow deploys resulted in slower developer velocity.

Data fetching service

Microservices

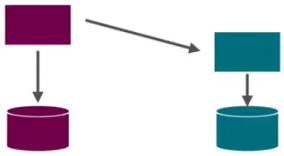
2017 - 2020



Business logic data service

Microservices

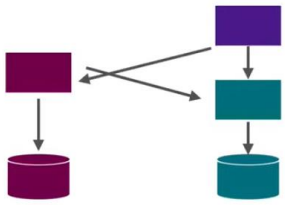
2017 - 2020



Write workflow service

Microservices

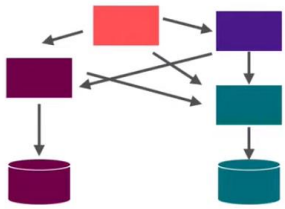
2017 - 2020



UI aggregation service

Microservices

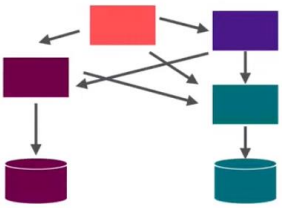
2017 - 2020



API gateway

Microservices

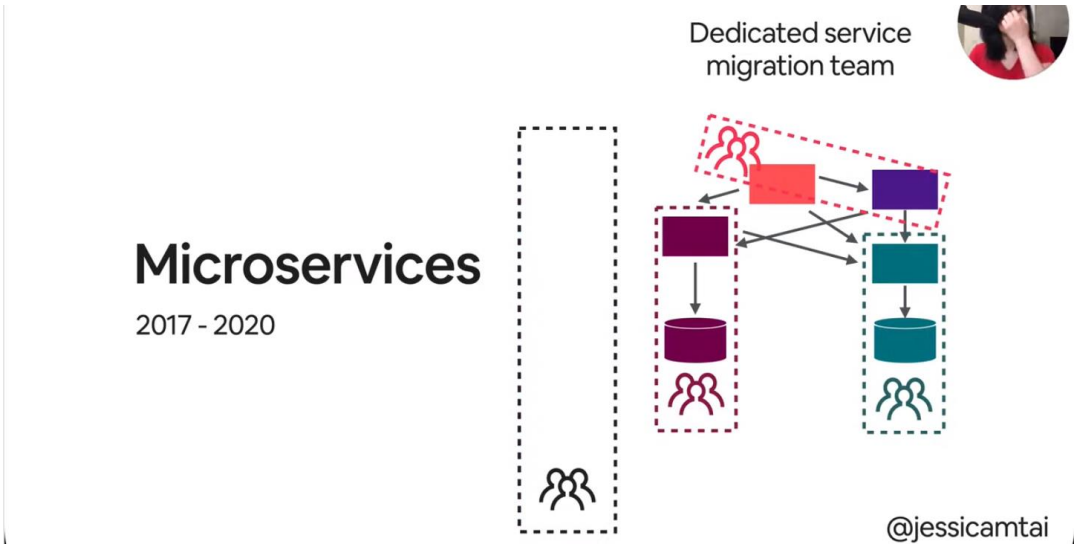
2017 - 2020



Each service had one owning team

More backend only teams & services

Dedicated service migration team



Microservices

2017 - 2020

@jessicamtai

Features required changes from multiple services & teams

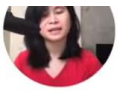
Hundreds of services and dependencies were difficult for humans to manage.

Micro + macroservices

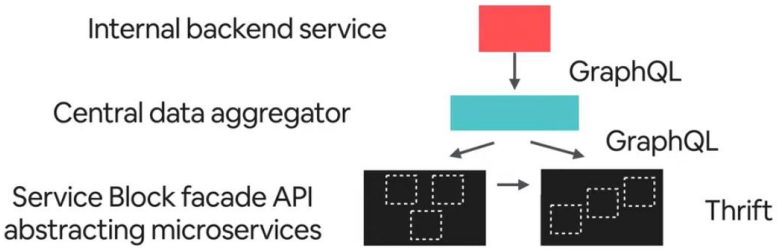
2020 - present

See previous QCon Plus talk for details!

Micro + macroservices

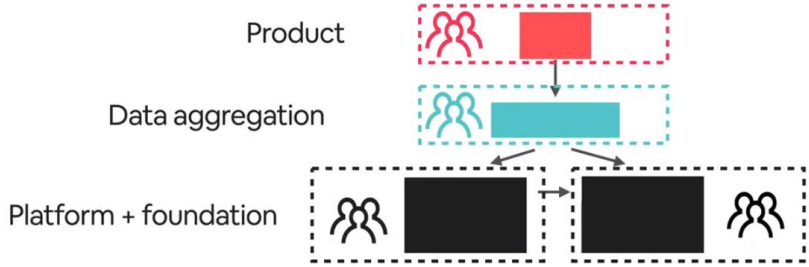


Unified APIs



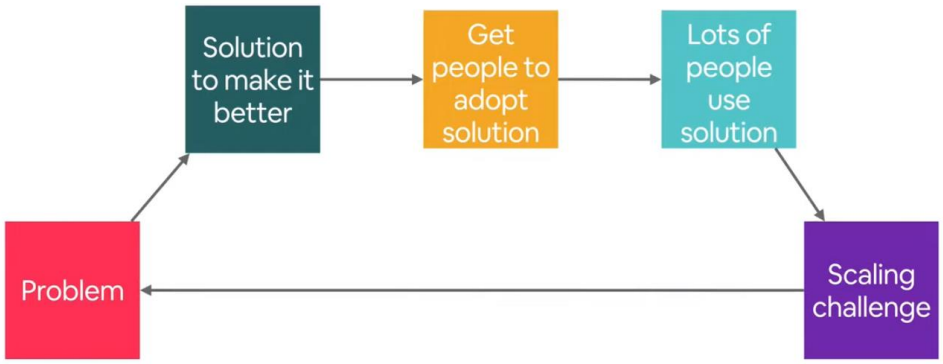
Micro + macroservices

Specialized teams

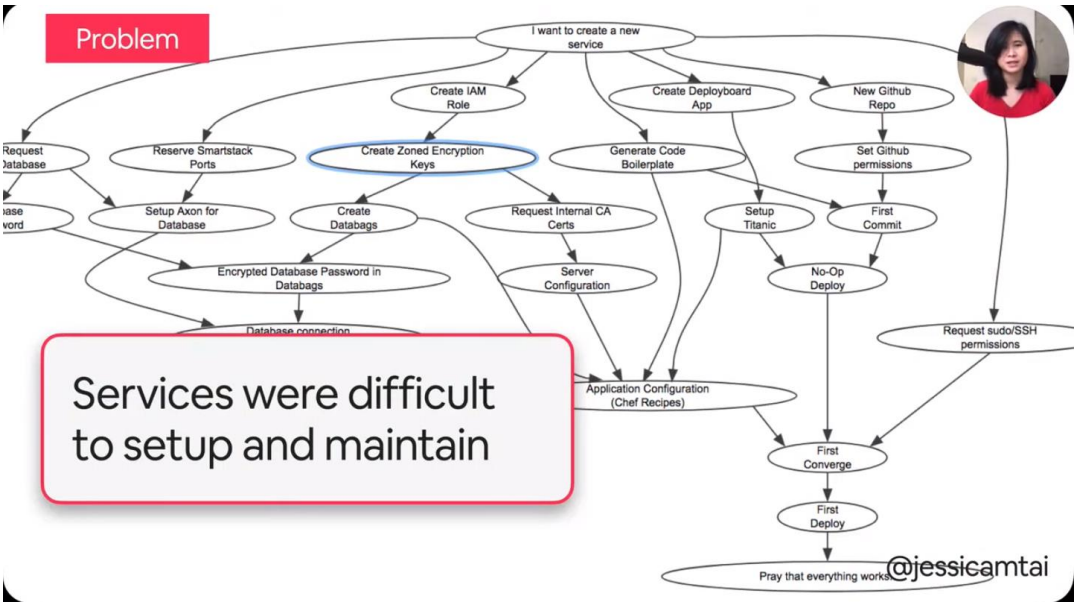


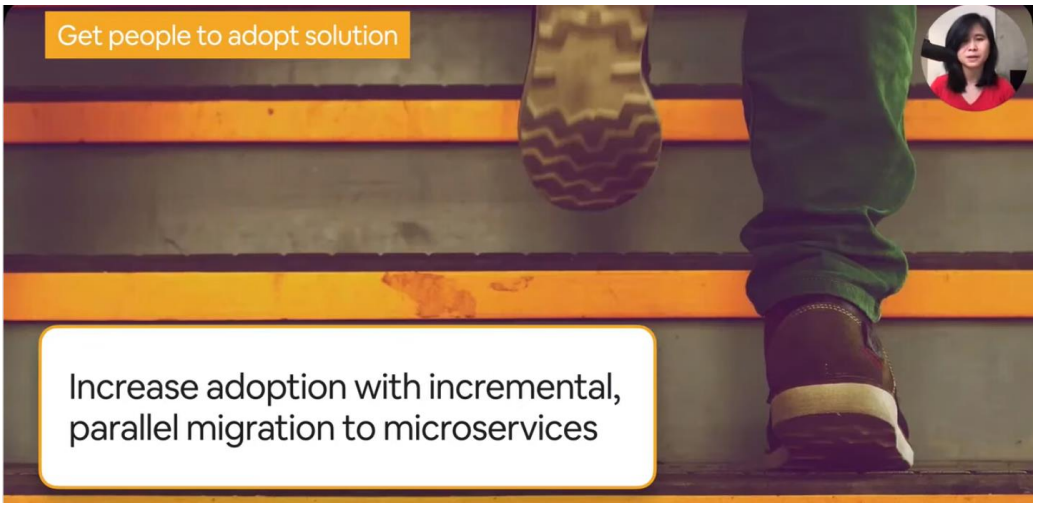
Migration hits and misses

Migration challenge journey

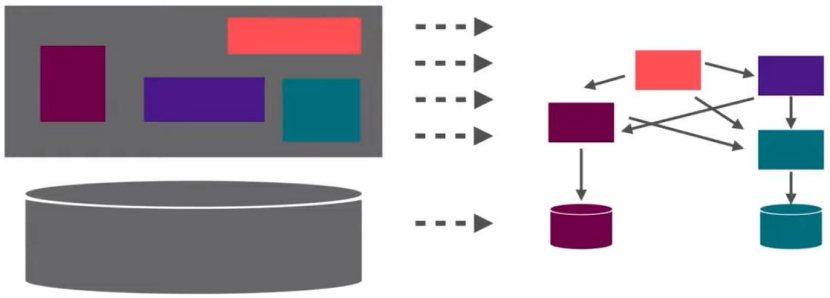


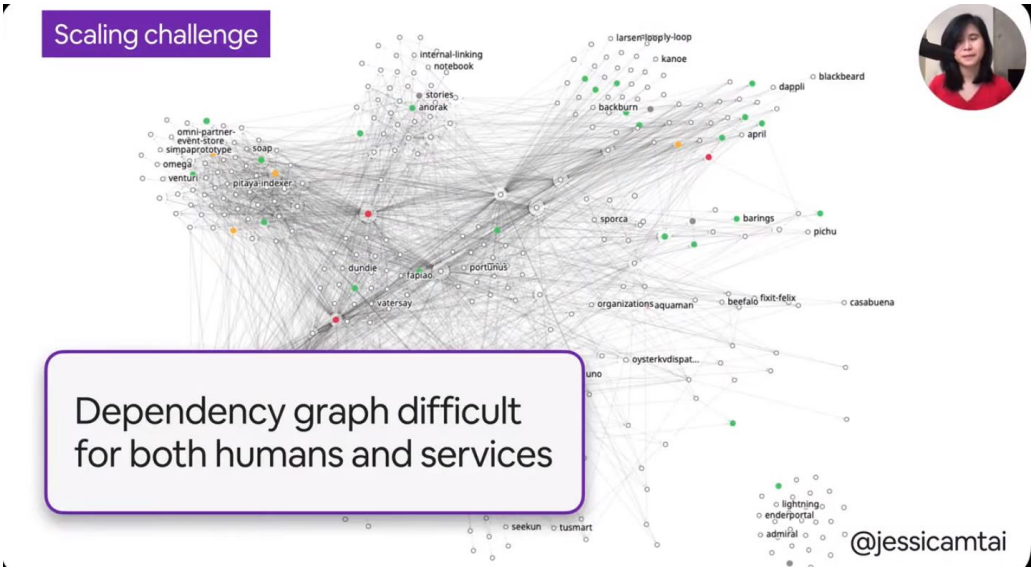
From Monolith to Services



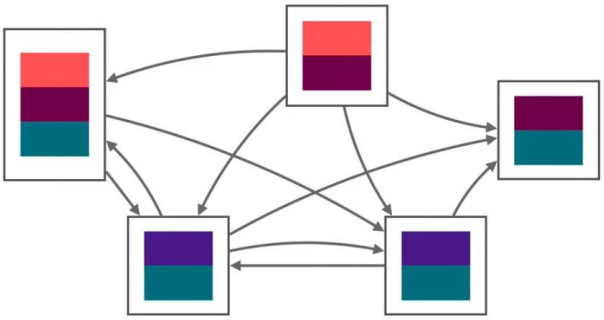


Migrate to new services across stack in parallel

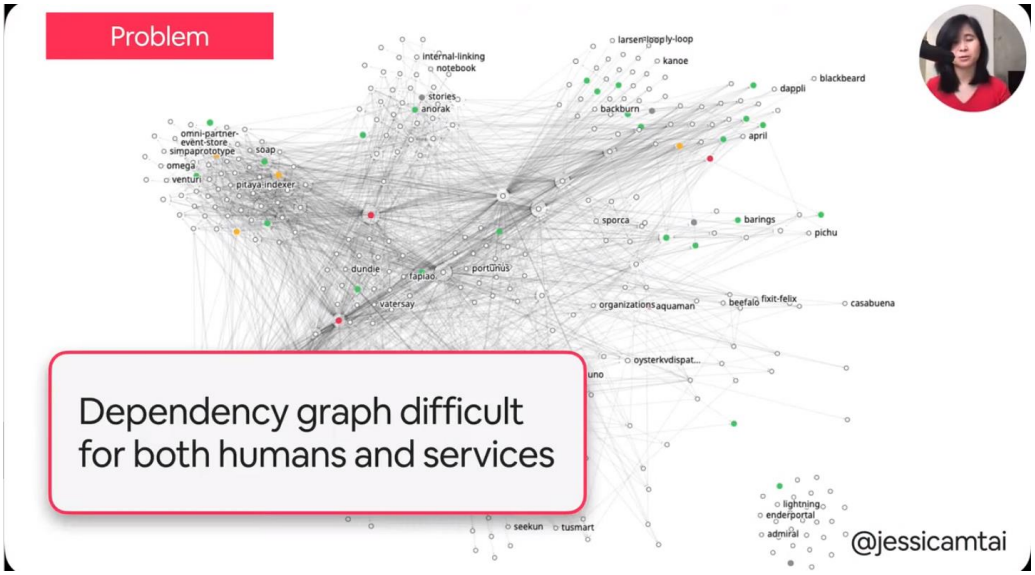


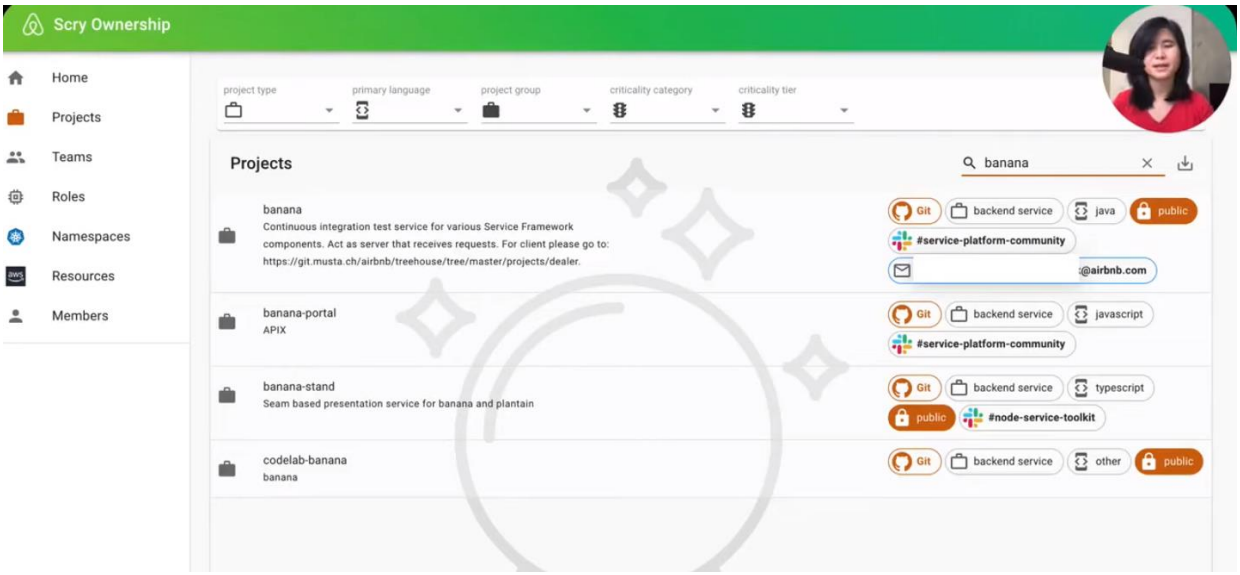


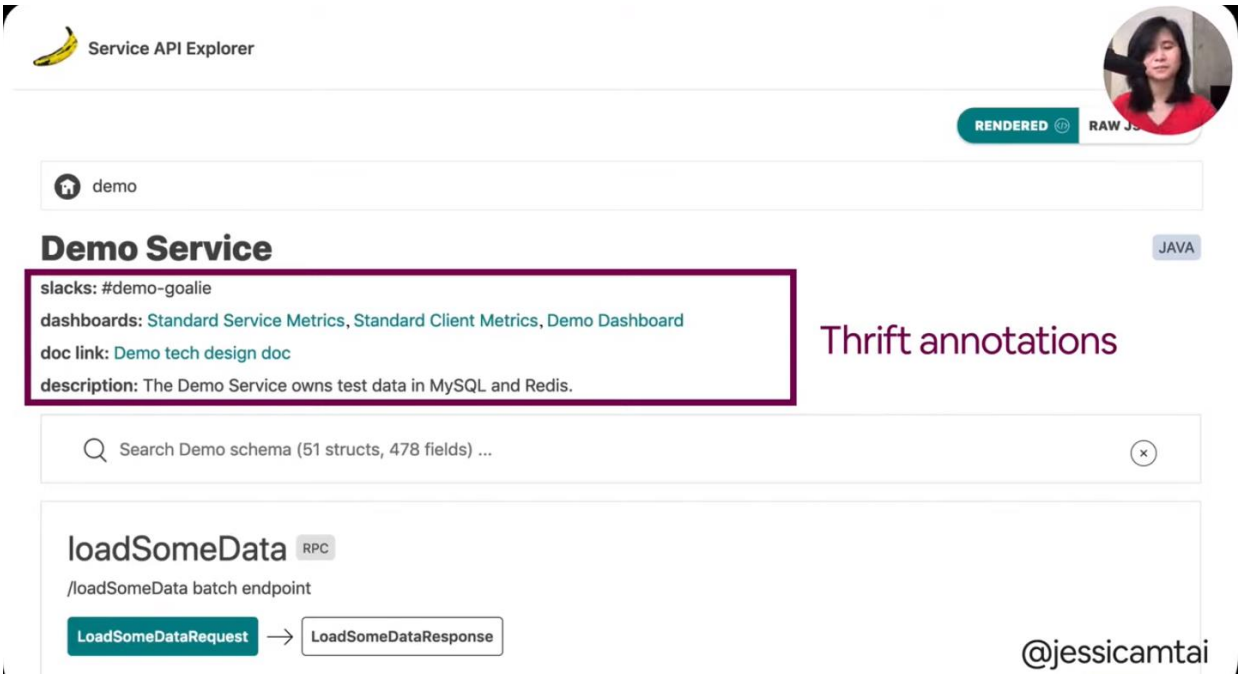
Quick dev velocity led to mixed functionality & APIs

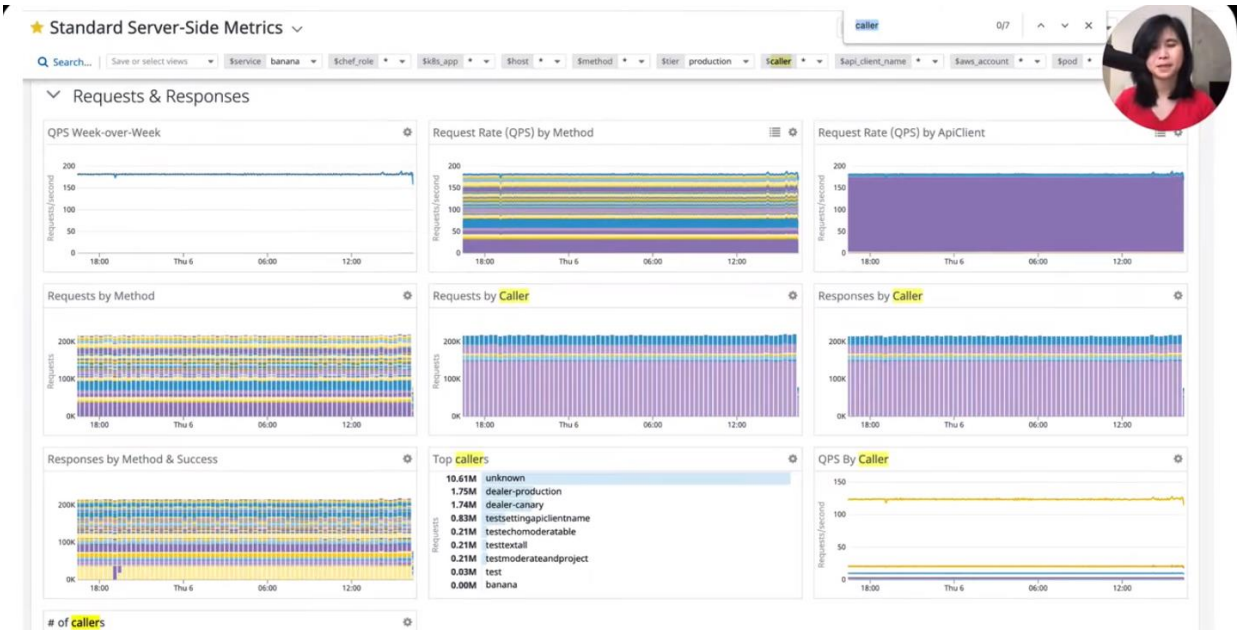


Services at Scale




Scry Ownership
Home
Projects
Teams
Roles
Namespaces
Resources
Members
project type
primary language
project group
criticality category
criticality tier
Projects
banana
banana
Continuous integration test service for various Service Framework components. Act as server that receives requests. For client please go to: https://git.musta.ch/airbnb/treehouse/tree/master/projects/dealer.
Git
backend service
java
public
#service-platform-community
@airbnb.com
banana-portal
APIX
Git
backend service
javascript
#service-platform-community
banana-stand
Seam based presentation service for banana and plantain
Git
backend service
typescript
public
#node-service-toolkit
codelab-banana
banana
Git
backend service
other
public

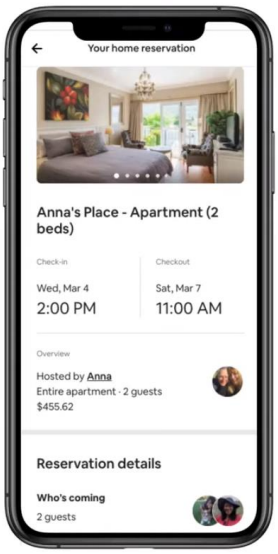

Service API Explorer
RENDERED
demo
Demo Service
JAVA
slacks: #demo-goalie
dashboards: Standard Service Metrics, Standard Client Metrics, Demo Dashboard
doc link: Demo tech design doc
description: The Demo Service owns test data in MySQL and Redis.
Thrift annotations
Search Demo schema (51 structs, 478 fields) ...
loadSomeData
RPC
/loadSomeData batch endpoint
LoadSomeDataRequest
LoadSomeDataResponse
@jessicamtai


Standard Server-Side Metrics
Requests & Responses
QPS Week-over-Week
Request Rate (QPS) by Method
Request Rate (QPS) by ApiClient
Requests by Method
Requests by Caller
Responses by Caller
Responses by Method & Success
Top callers
10.61M unknown
1.75M dealer-production
1.74M dealer-canary
0.83M testsettingapiclientname
0.21M testechomoderatable
0.21M testtextail
0.21M testmoderateandproject
0.03M test
0.00M banana
QPS By Caller
# of callers

Solution to make it better

Replace bespoke Thrift schemas with unified GraphQL

```
type Reservation {
  checkInDate: Date
  numberOfGuests: Int
  …
}
```



Get people to adopt solution

Improve velocity with annotations & codegen

```
type Reservation {
  @owners(lists: “team-123”)
  @serviceBackedNode(
      service: “ExampleService”
      methodName: “/example/endpoint”
      …
    )
  checkInDate: Date @dataClassification(level: “other_data”)
  numberOfGuests: Int @dataClassification(level: “other_data”)
    @privacy(…)
}
```

Get people to adopt solution

Improve velocity with annotations & codegen

```
type Reservation {
  @owners(lists: “team-123”)
  @serviceBackedNode(
      service: “ExampleService”
      methodName: “/example/endpoint”
      …
    )
  checkInDate: Date @dataClassification(level: “other_data”)
  numberOfGuests: Int @dataClassification(level: “other_data”)
    @privacy(…)
}
```

Alerts, mandatory code reviewers

Get people to adopt solution

Improve velocity with annotations & codegen

```
type Reservation {
  @owners(lists: “team-123”)
  @serviceBackedNode(
      service: “ExampleService”
      methodName: “/example/endpoint”
      …
    )
  checkInDate: Date @dataClassification(level: “other_data”)
  numberOfGuests: Int @dataClassification(level: “other_data”)
    @privacy(…)
}
```

Resolver to call internal microservice

Get people to adopt solution

Improve velocity with annotations & codegen

```
type Reservation {
  @owners(lists: "team-123")
  @serviceBackedNode(
      service: "ExampleService"
      methodName: "/example/endpoint"
      …
    )
  checkInDate: Date @dataClassification(level: "other_data")
  numberOfGuests: Int @dataClassification(level: "other_data")
    @privacy(…)
}
```

Permission checks to internal service

Lots of people use solution

Migrated hundreds of data entities to unified GraphQL schema

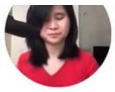
Scaling challenge

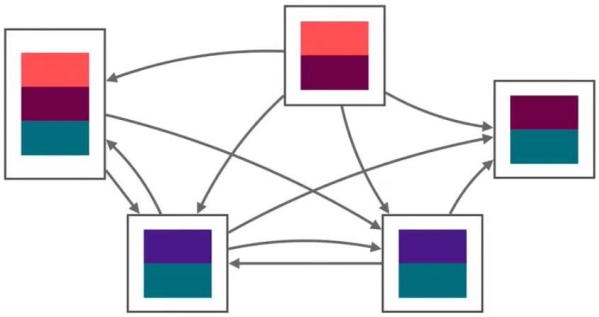
Slower build and deploy times for central data aggregator

What's next?

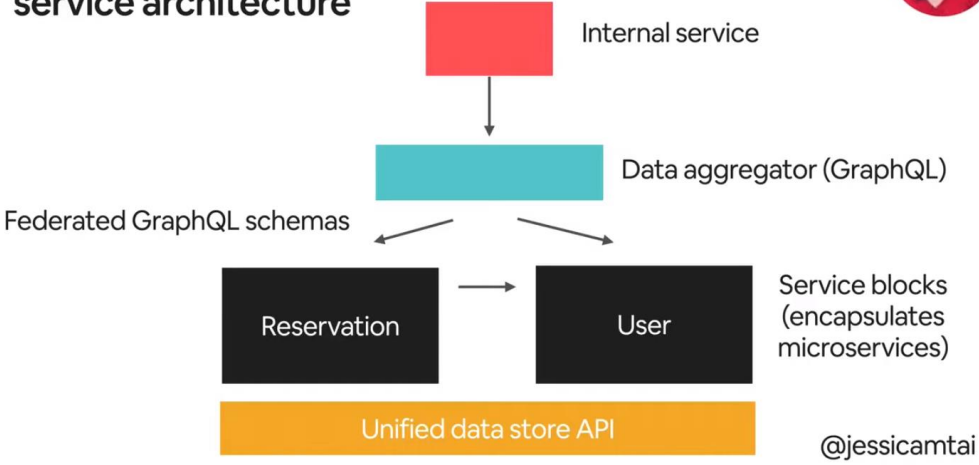
Airbnb is moving towards a hybrid mode - aggregators abstracting microservices.

Migrate away from hundreds of free-range services



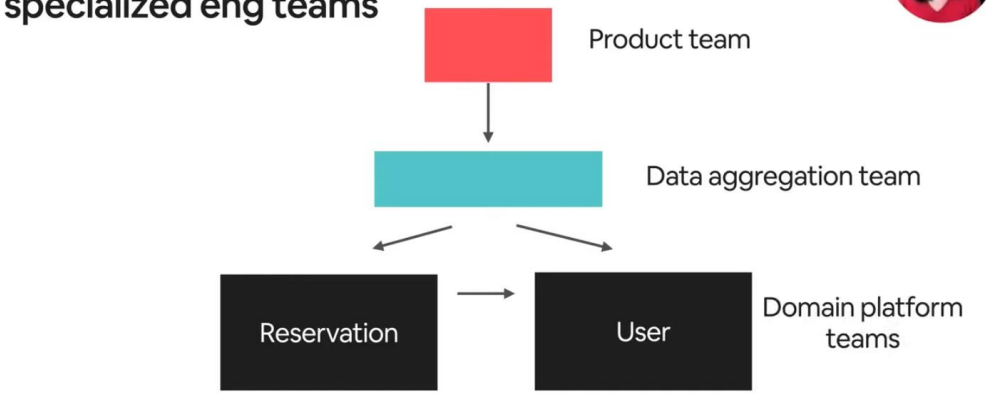


Enforce paved path for micro+macro service architecture

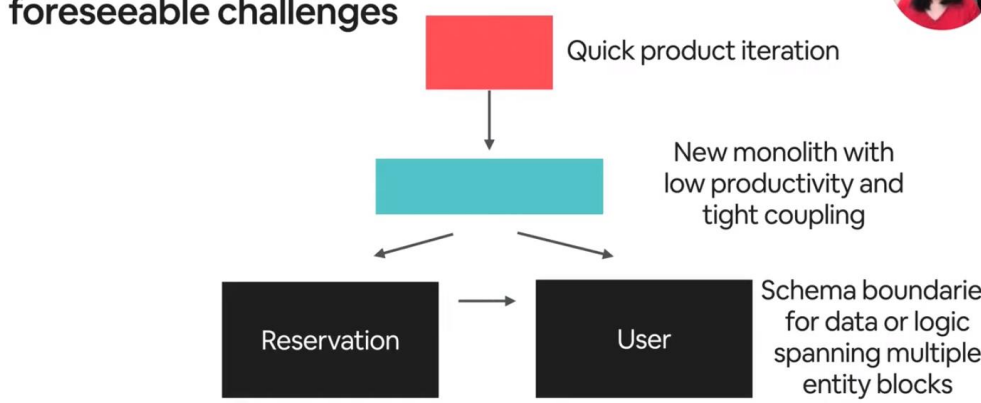


@jessicamtai

Clarify responsibilities with specialized eng teams



Get ahead of foreseeable challenges



Deprecation of the monolith takes a very long time.

Deprecation working group

Mobile app deprecation >12 months

Elevate & track monolith tech debt

Sunset low usage endpoints

Long-term ownership

Recognize deprecation as impactful

Restructure teams to collaborat
new architecture

Takeaways

Improve human productivity

Invest in developer productivity through tools, automation

Increase focus with central domains and platform teams

Expect new challenges to appear

Decentralizing into microservices created both new technical and human challenges...

But many of the growing pains were necessary parts of our evolving journey.