Microservices to Monolith

Lessons from NBA TopShot

Quinn Hou
Senior Software Engineer @ Dapper Labs
October 2023

Hello

Quinn Hou
Senior Software Engineer
4 years @ Dapper Labs

NBA TopShot, NFL AllDay,
LaLiga Golazos, R&D, Cheeze Wizards,
Dapper Wallet

Agenda

01 TopShot microservices

02 Example and challenges

03 AllDay simplification

04 Distributed example

05 Adapting to needs

06 Key learnings


HARDEN
LEONARD
CURRY
YOUNG
NBA
TOP SHOT


NFL
ALL DAY
NFLPA

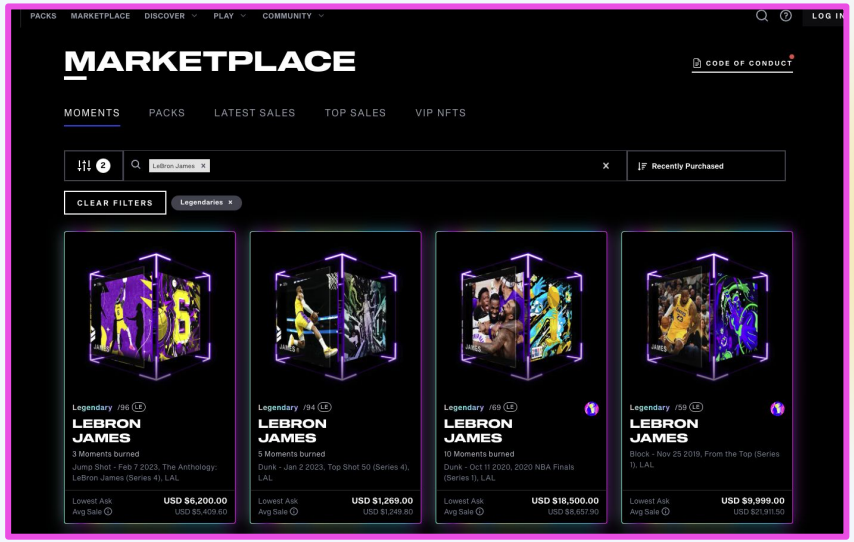
01

NBA TopShot

TopShot Introduction

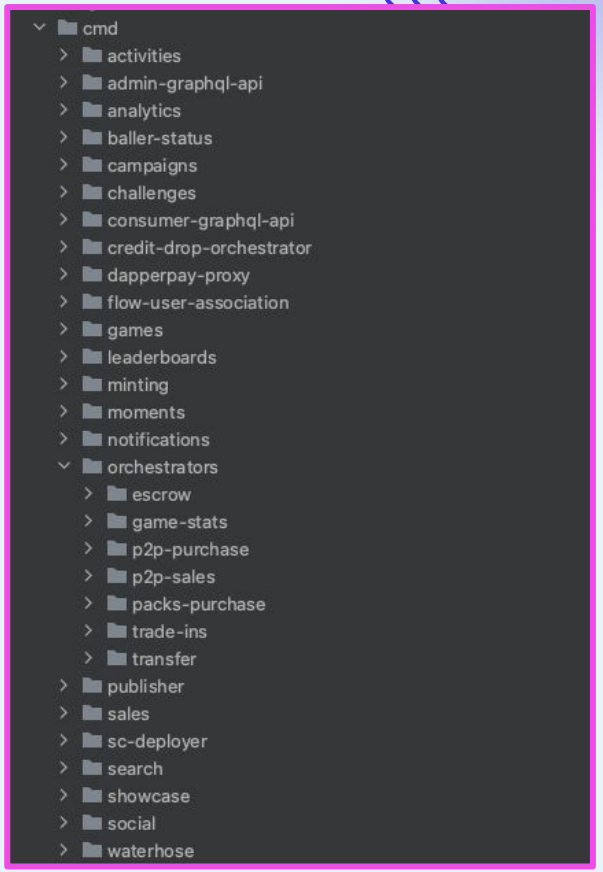
The product

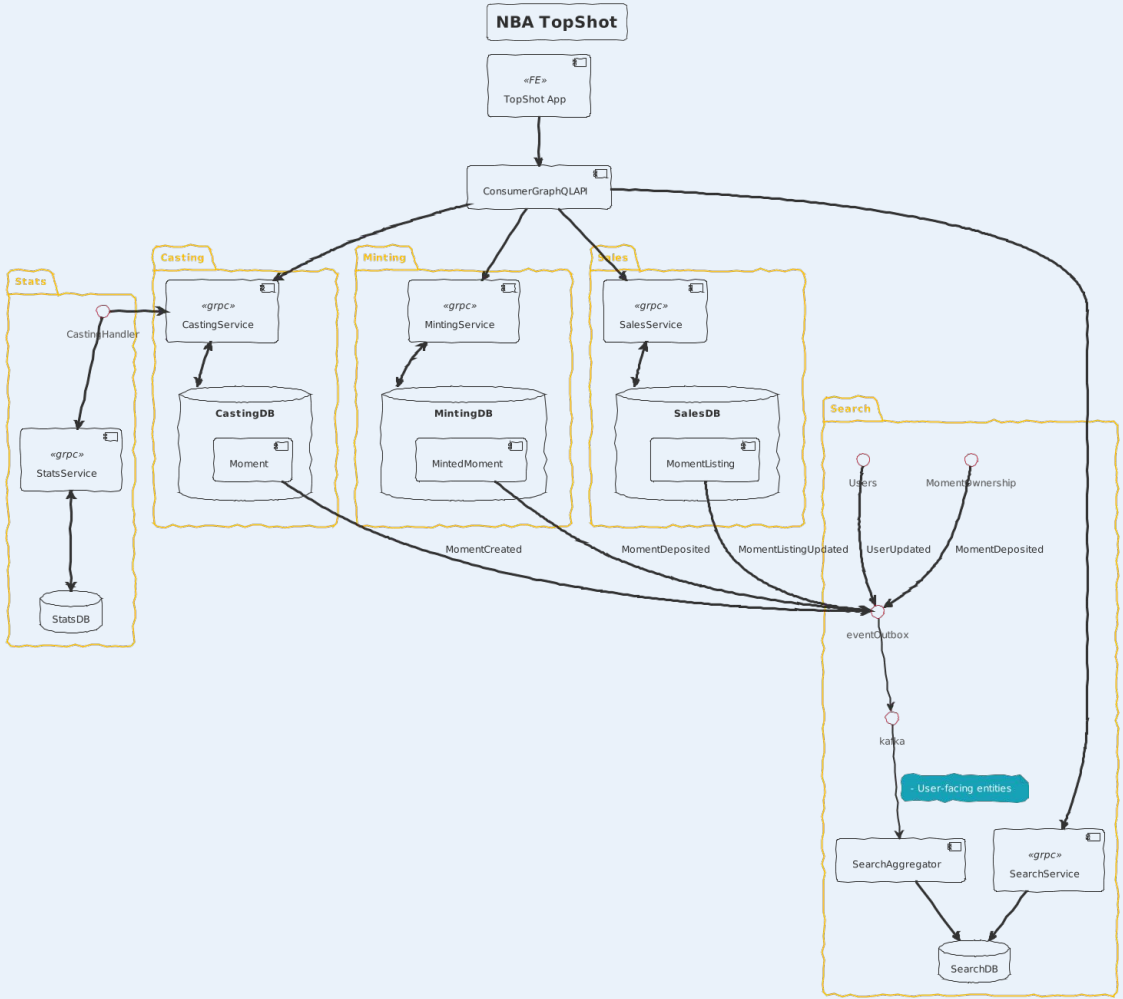
Sharing NBA fandom through NFT collectibles on the blockchain



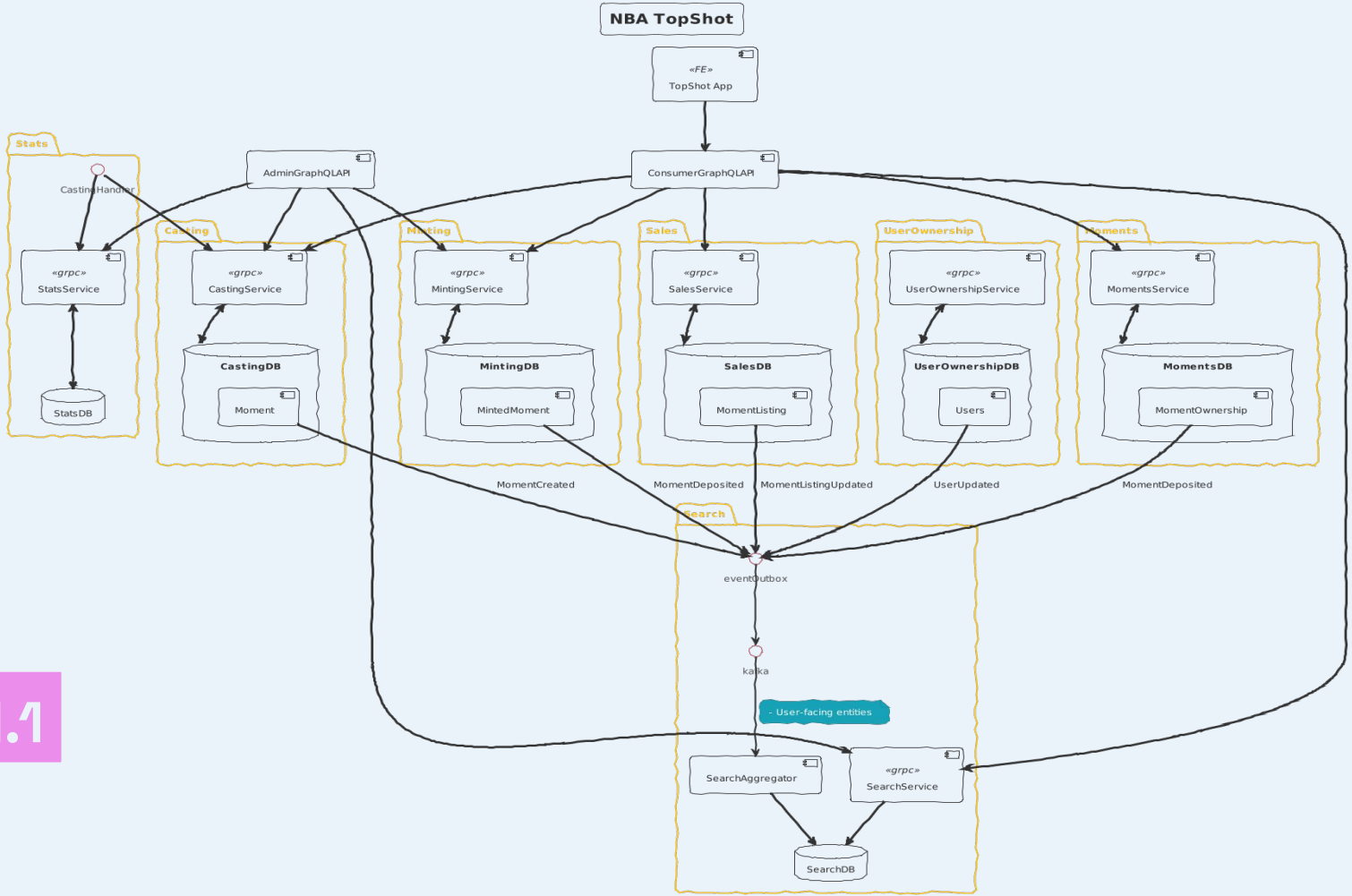
TopShot Microservices

28+ microservices
Corresponding databases
Separate read and writes
Event-driven




NBA TopShot
«FE»
TopShot App
ConsumerGraphQLAPI
Stats
Casting
Minting
Sales
Search
«grpc»
CastingService
«grpc»
MintingService
«grpc»
SalesService
CastingHandler
CastingDB
Moment
MintingDB
MintedMoment
SalesDB
MomentListing
«grpc»
StatsService
StatsDB
Users
MomentOwnership
MomentCreated
MomentDeposited
MomentListingUpdated
UserUpdated
MomentDeposited
eventOutbox
kafka
- User-facing entities
SearchAggregator
«grpc»
SearchService
SearchDB

v1


NBA TopShot
«FE»
TopShot App
AdminGraphQLAPI
ConsumerGraphQLAPI
Stats
Casting Handler
«grpc»
StatsService
StatsDB
Casting
«grpc»
CastingService
CastingDB
Moment
Minting
«grpc»
MintingService
MintingDB
MintedMoment
Sales
«grpc»
SalesService
SalesDB
MomentListing
UserOwnership
«grpc»
UserOwnershipService
UserOwnershipDB
Users
Moments
«grpc»
MomentsService
MomentsDB
MomentOwnership
MomentCreated
MomentDeposited
MomentListingUpdated
UserUpdated
MomentDeposited
Search
eventOutbox
kafka
User-facing entities
SearchAggregator
«grpc»
SearchService
SearchDB

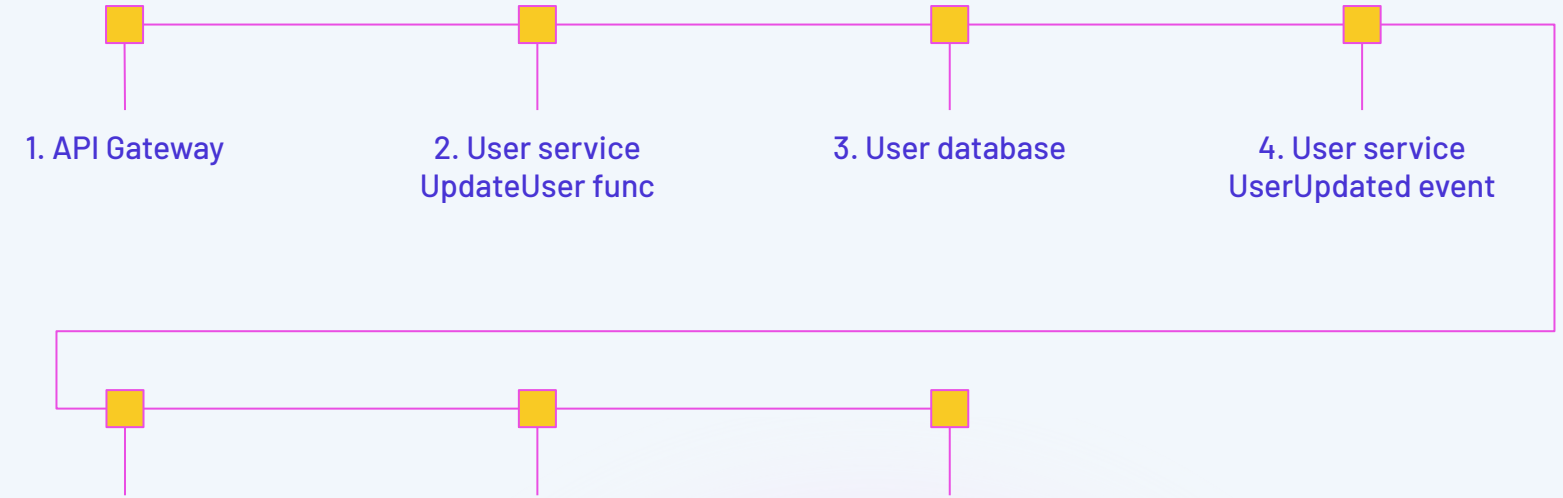
v1.1

02

Example and Challenges

TopShot Microservices

Example: updating a user



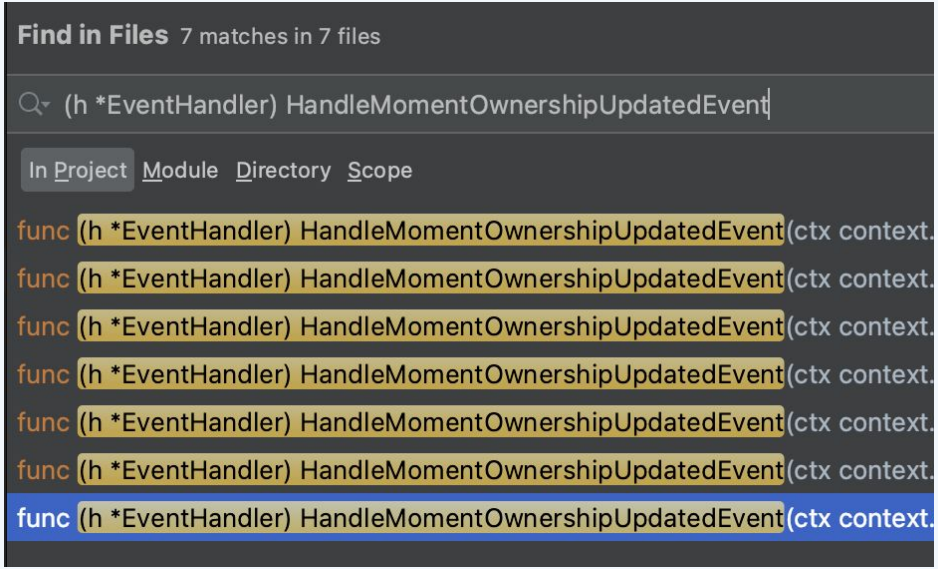
Microservices: the challenges

Interservice Communication

Redundant and complex code

Slow development speed

The Challenges: redundant code



SearchAggregator Handler Pseudocode

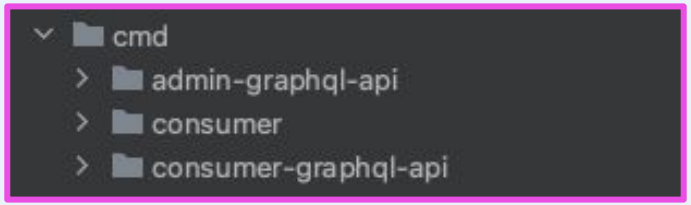
```
// HandleMomentOwnershipUpdatedEvent handles an incoming momentOwnershipUpdatedEvent from user-moment-service
// and updates the owner details if the ownership has changed
function HandleMomentOwnershipUpdatedEvent(msg):
      event = parseMessageToEvent(msg)
      momentOwnership, err = createMomentOwnershipFromEvent(event)
      if err: return err

      If !validateMomentOwnership(momentOwnership):
            return err

      if upsertMomentOwnership(ctx, momentOwnership) has error:
            return err

      if updateMomentMetadataCirculationData(momentId) has error:
            return  err
      if recalculateTopshotScoreForUser(ctx, event, momentOwnership) has error:
            return err
      return null
```

Sales-service Handler Pseudocode

```
// HandleMomentOwnershipUpdatedEvent updates the owner details if the ownership has changed
function HandleMomentOwnershipUpdatedEvent(h, ctx, msg):
      event = parseMessageToEvent(msg)

      newOwner = determineNewOwner(flowOwnerAddress)

      err = updateMomentOwnershipForPacks(momentFlowID, newOwner,
            func(ctx, packMoment):
                  outboxEvents publishPackMomentUpdatedEvent(packMoment)
      if err: return err

      return null
```

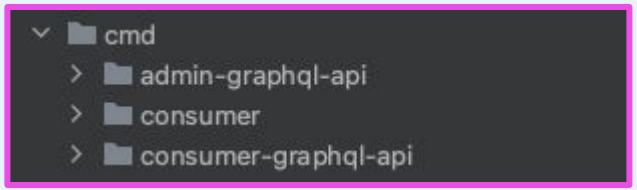
An Evolutionary Relationship

03

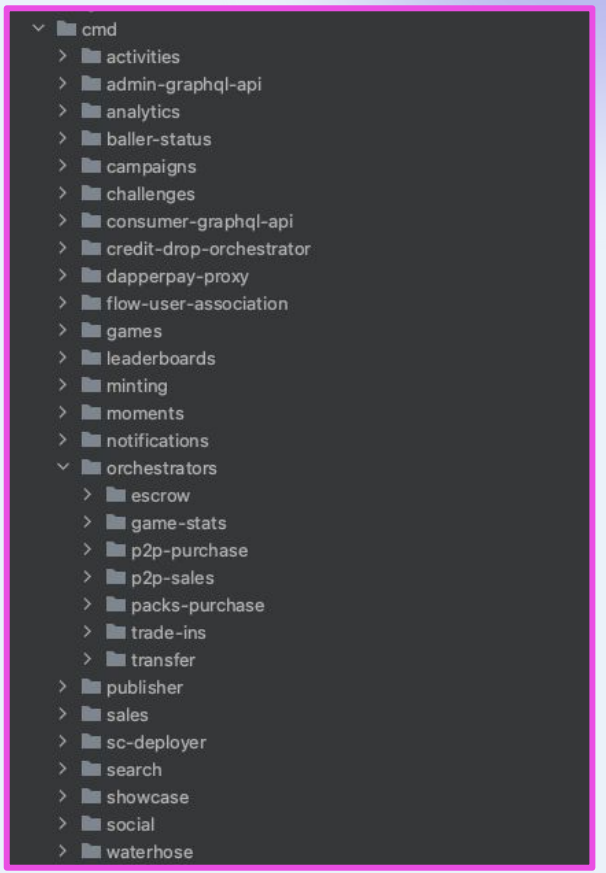


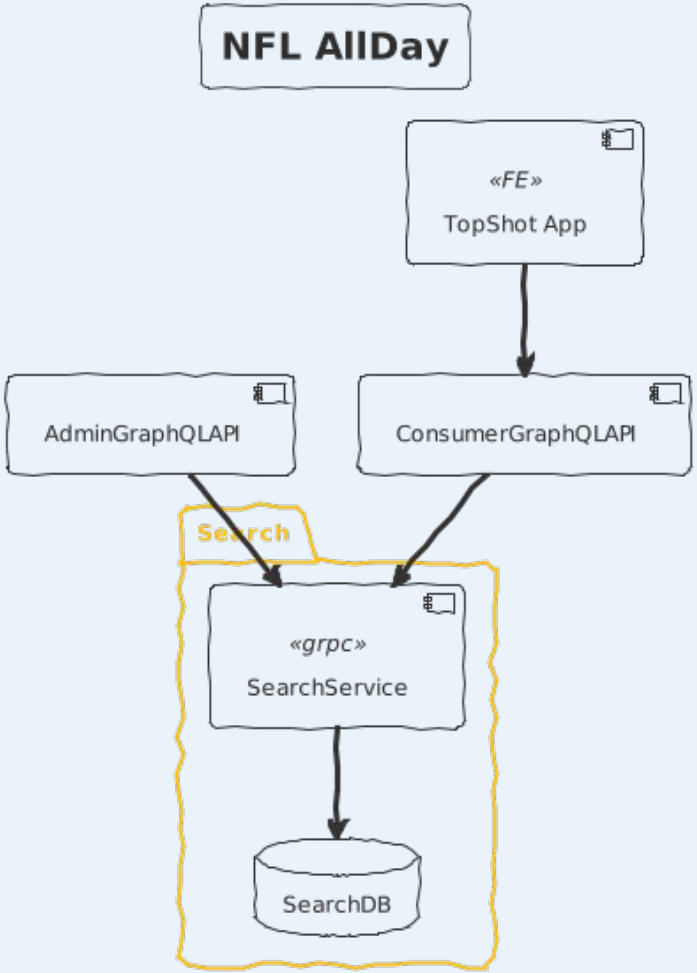
NFL AllDay

A monolithic starting point



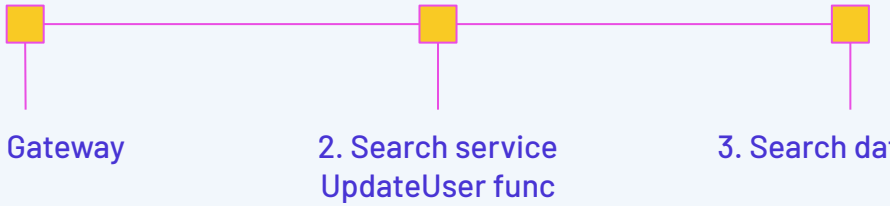
(We won't need a full page this time)




NFL AllDay
«FE»
TopShot App
AdminGraphQLAPI
ConsumerGraphQLAPI
Search
«grpc»
SearchService
SearchDB

v1

Feature Comparison: updating a user



Monolithic: the nice things

01 Straightforward to reason

02 Faster implementation

03 Happier Developers

04 Immediate consistency

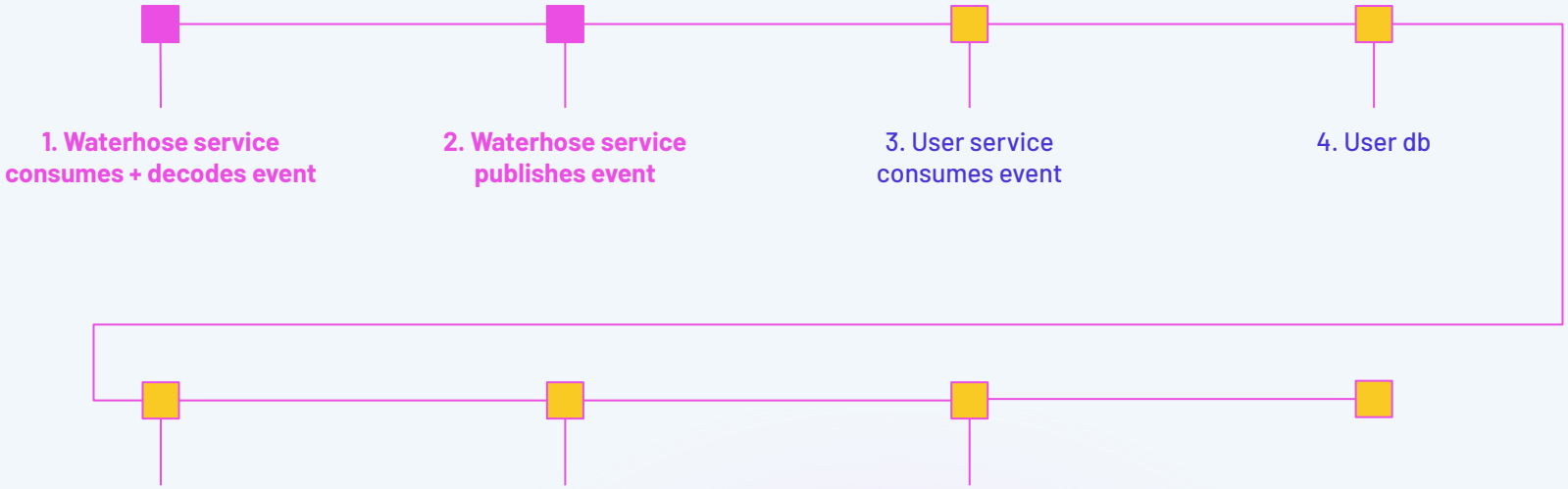
05 Easier expansion

04

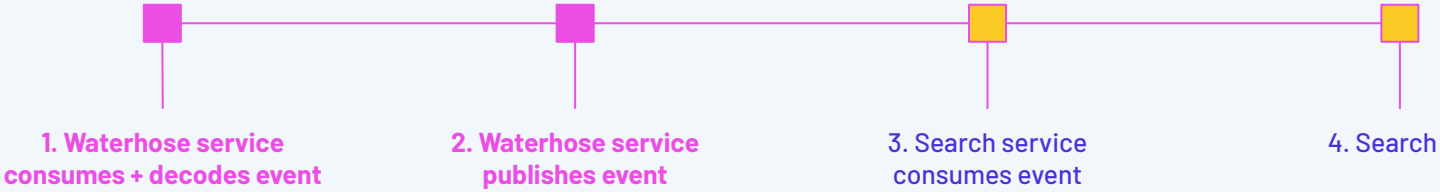
Comparison: Distributed Example

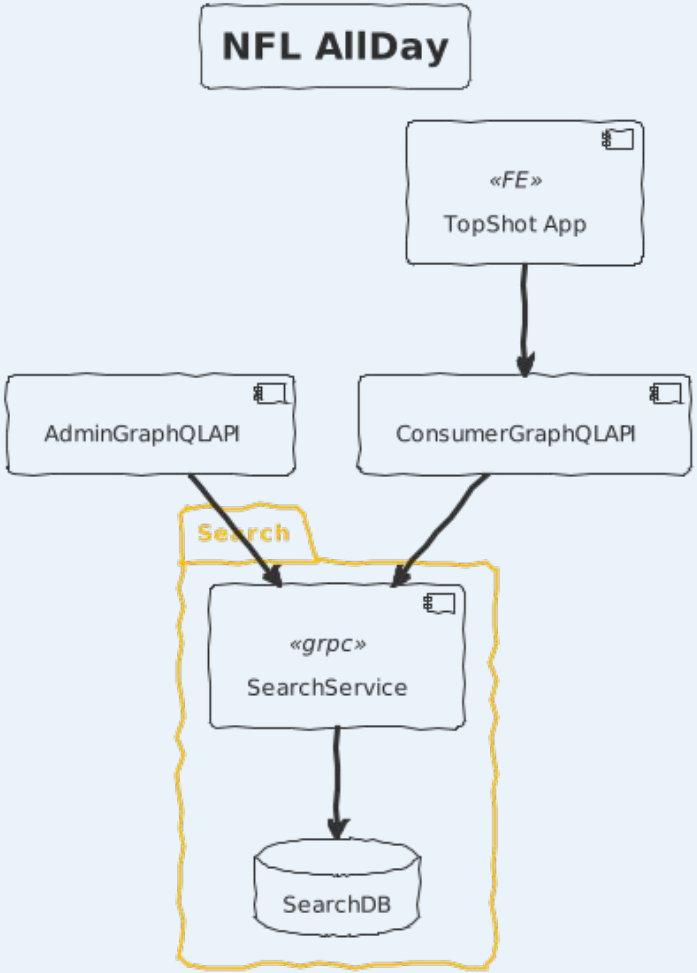
MomentDepositedEvent

MomentDeposited on TopShot

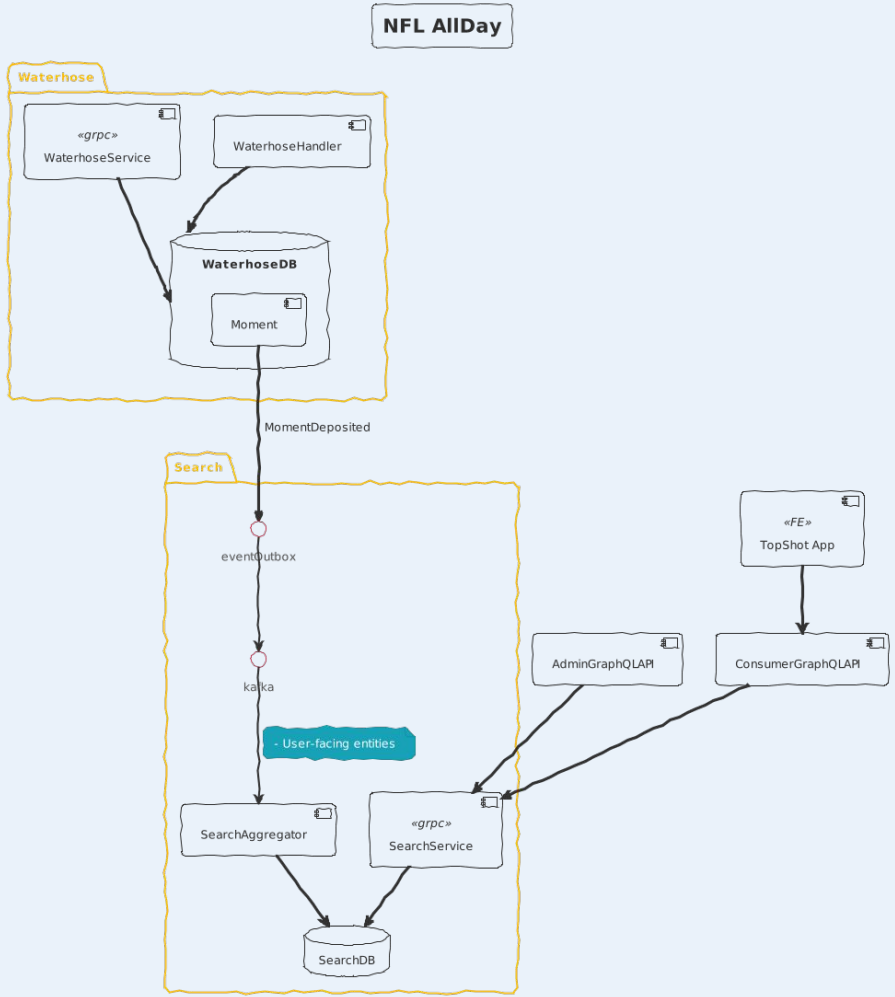


AllDay, In Contrast...




NFL AllDay
«FE»
TopShot App
AdminGraphQLAPI
ConsumerGraphQLAPI
Search
«grpc»
SearchService
SearchDB

v1


NFL AllDay
Waterhose
«grpc»
WaterhoseService
WaterhoseHandler
WaterhoseDB
Moment
MomentDeposited
Search
eventOutbox
kafka
- User-facing entities
SearchAggregator
«grpc»
SearchService
SearchDB
«FE»
TopShot App
AdminGraphQLAPI
ConsumerGraphQLAPI

v2

05

Adapting to Needs

Creating New Services

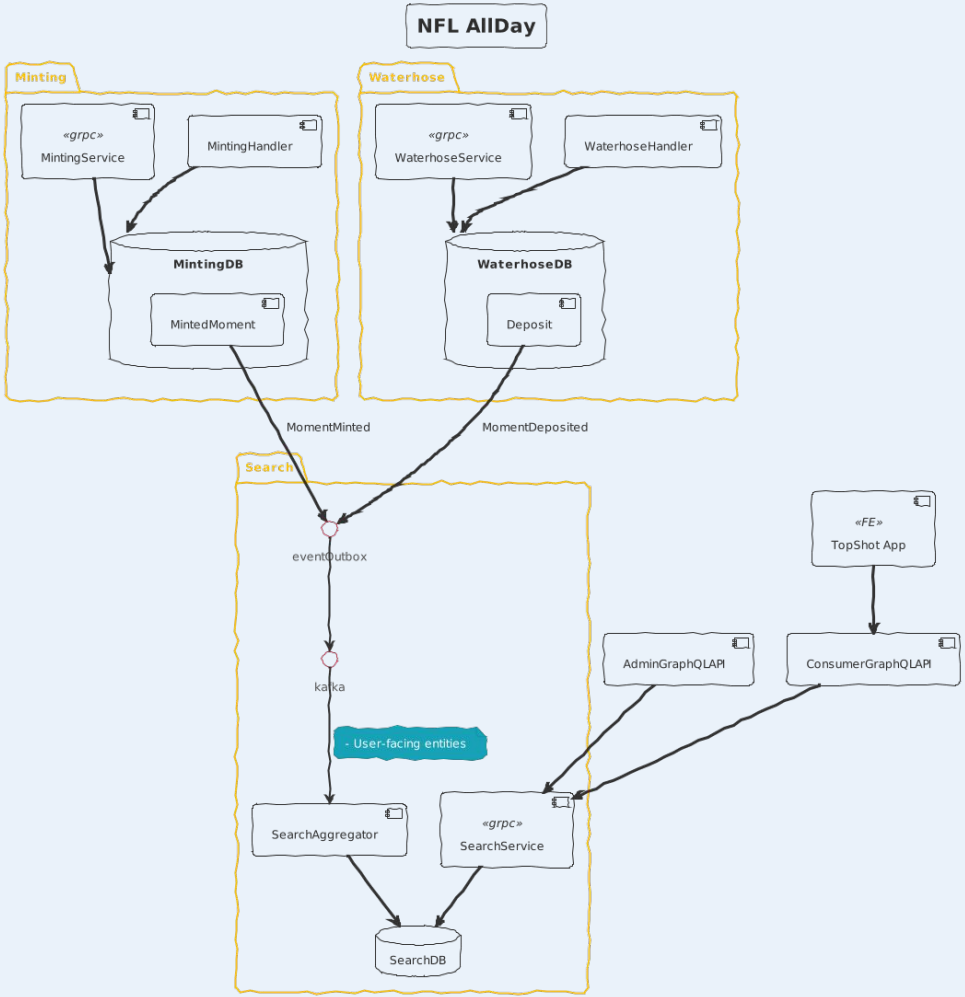
Knowing when

(to spin out new services)

Clear domain understanding

Stable boundaries

Independent scalability


NFL AllDay
Minting
«grpc»
MintingService
MintingHandler
MintingDB
MintedMoment
Waterhose
«grpc»
WaterhoseService
WaterhoseHandler
WaterhoseDB
Deposit
MomentMinted
MomentDeposited
Search
eventOutbox
kafka
- User-facing entities
SearchAggregator
«grpc»
SearchService
SearchDB
«FE»
TopShot App
AdminGraphQLAPI
ConsumerGraphQLAPI

v3

06

Our Takeaways

01

Know your domain

02

Identify architecture-team fit

03

Understand trade-offs

04

Build for architectural evolution

07

> How do we build for **now** that can make it easier for where we want to go?

Conclusion

Thank you

quinn.hou@dapperlabs.com
quinnhou

Q&A