AWS re:Invent

NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

ARC213

Create a scalable mobile service for enterprise success

Hyungil Kim
Sr. Solutions Architect
AWS

Piljoong Kim
Sr. Developer Specialist SA
AWS

Session goal

- Learn what scalability is in mobile service and how to factor it into service creation
- Learn proven solutions as patterns based on API, data, event, streams

Have more opportunities to ask your questions at any time!

Customer journey

Pivoting
(Outsourcing to in-house development)

Performance optimization
(Effectiveness)

Scale
(Growing user base)

Service integration
(Extensibility)

Capabilities of a scalable mobile service

Secure

Resilient

Elastic

Modular

Automated

Interoperable

Patterns

# Cloud-Native patterns, design patterns, system patterns, etc.

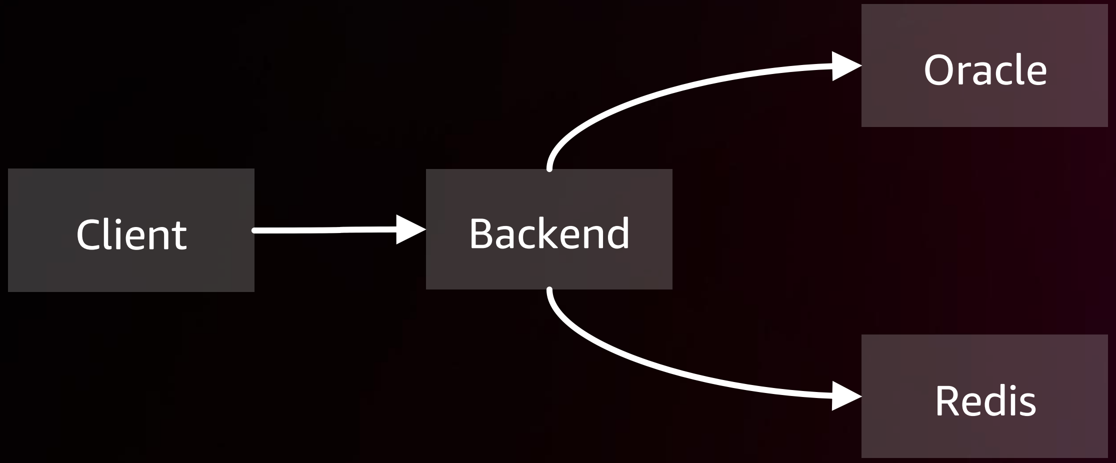
Strangle Monolith Application
A/B Testing
Automated Infrastructure
Automated Testing
Blamless Inquiry
Orchestration Pattern
Communicate Through APIs
Communicate Through Tribes
Containerized Apps
Continuous Delivery
Continuous Deployment
Continuous Integration
Dynamic Scheduling
Pub-Sub
Event-based Asynchronous

Circuit Breaker
Client-side Discovery
Self-registration
Server-side Discovery
Pre-Calculation
Polling Publisher
Function chaining
Transaction Log Tailing
Transactional Outbox
Saga
Aggregate
Domain Event
Domain Model
Event Sourcing
API Composition

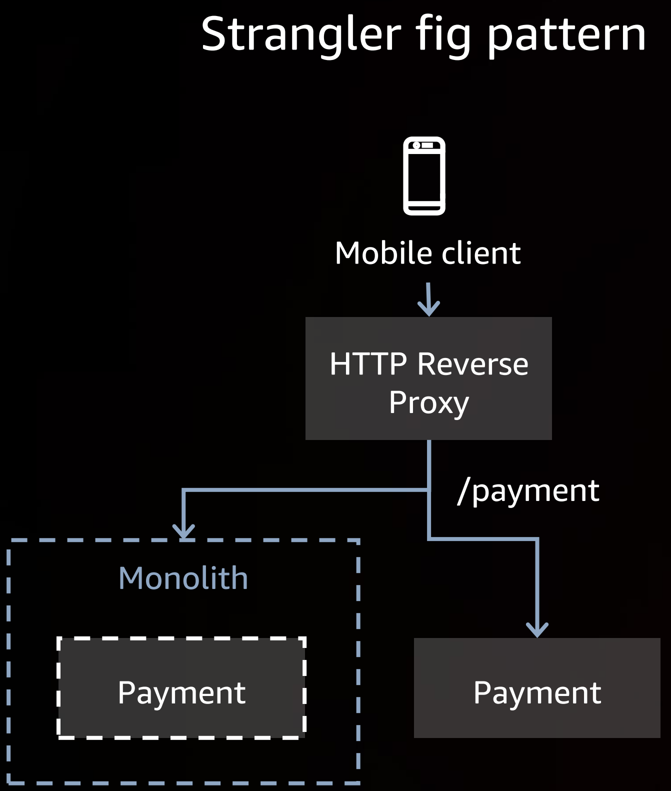
CQRS
API Gateway
Access Token
Externalized Configuration
Centralized Configuration
Application Metrics
Single-Receiver
Multiple-Receiver
Distributed Tracing
Log Aggregation
Service Mesh
Sidecar
Deploy Service as Container
Serverless Deployment
ACL

Whiteboarding

Started from here

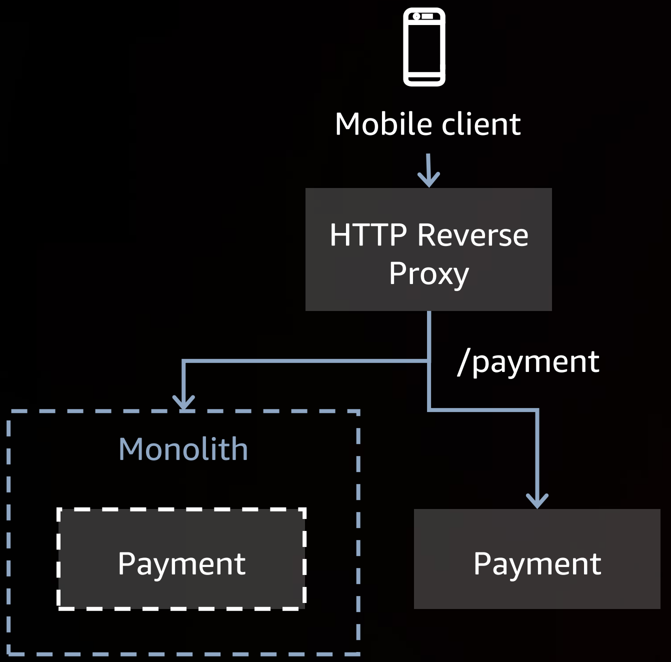


Pivoting: Split monolith

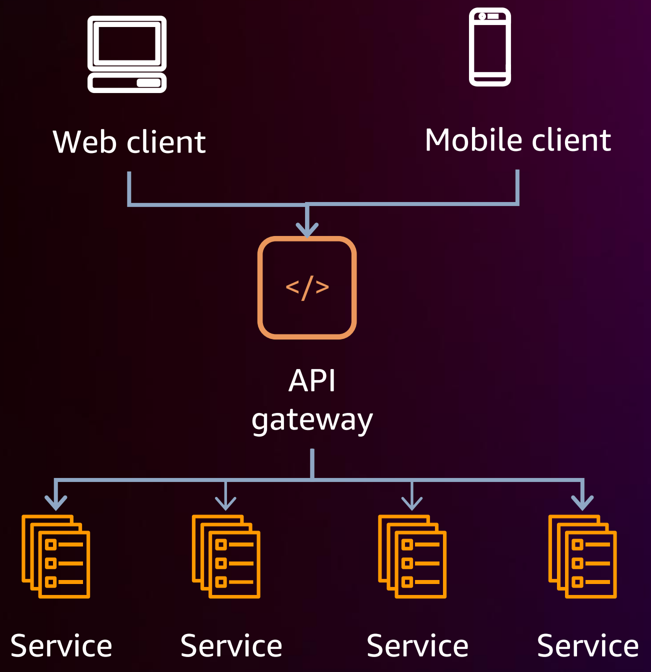


Pivoting: Split monolith

Strangler fig pattern



API gateway pattern



Pivoting: whiteboarding

Add more features on the existing application and migrate from a monolith to a series of microservices

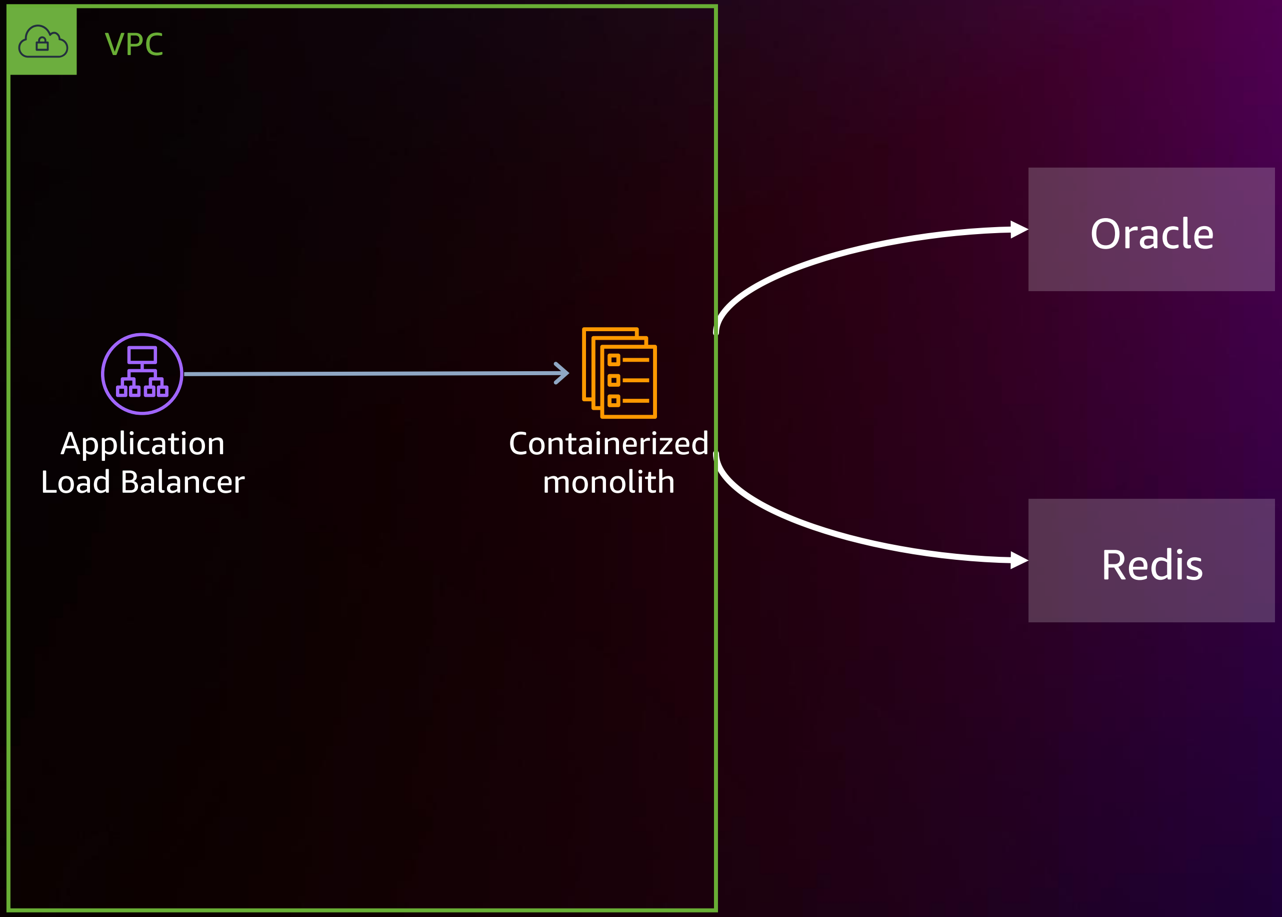
- Minimal impact on the existing monolith
- Expose functionality to internal and external parties
- Control access policies and have visibilities of API
- Roll out new services incrementally

Pivoting: Whiteboarding

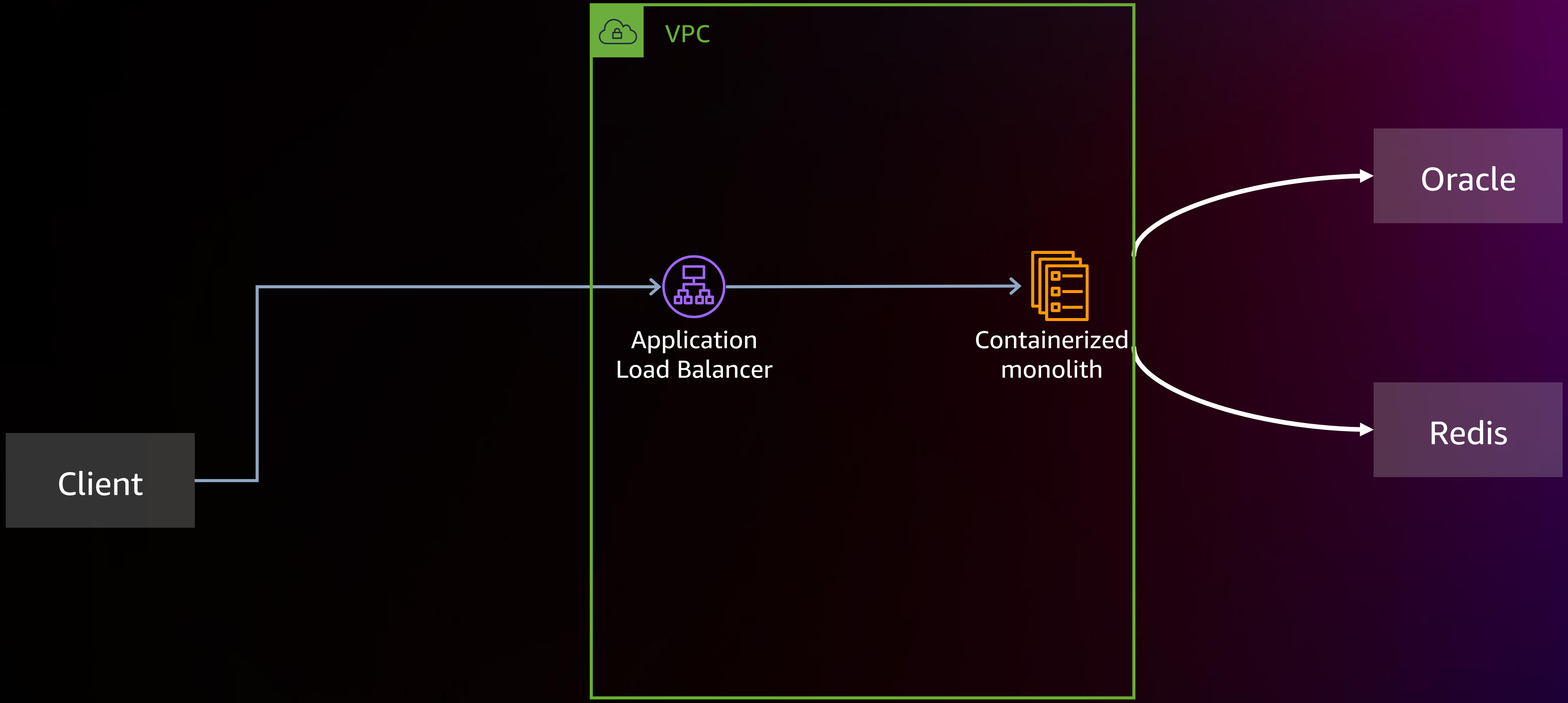
Oracle
Client
Redis

Pivoting | Scale | Optimization | Integration

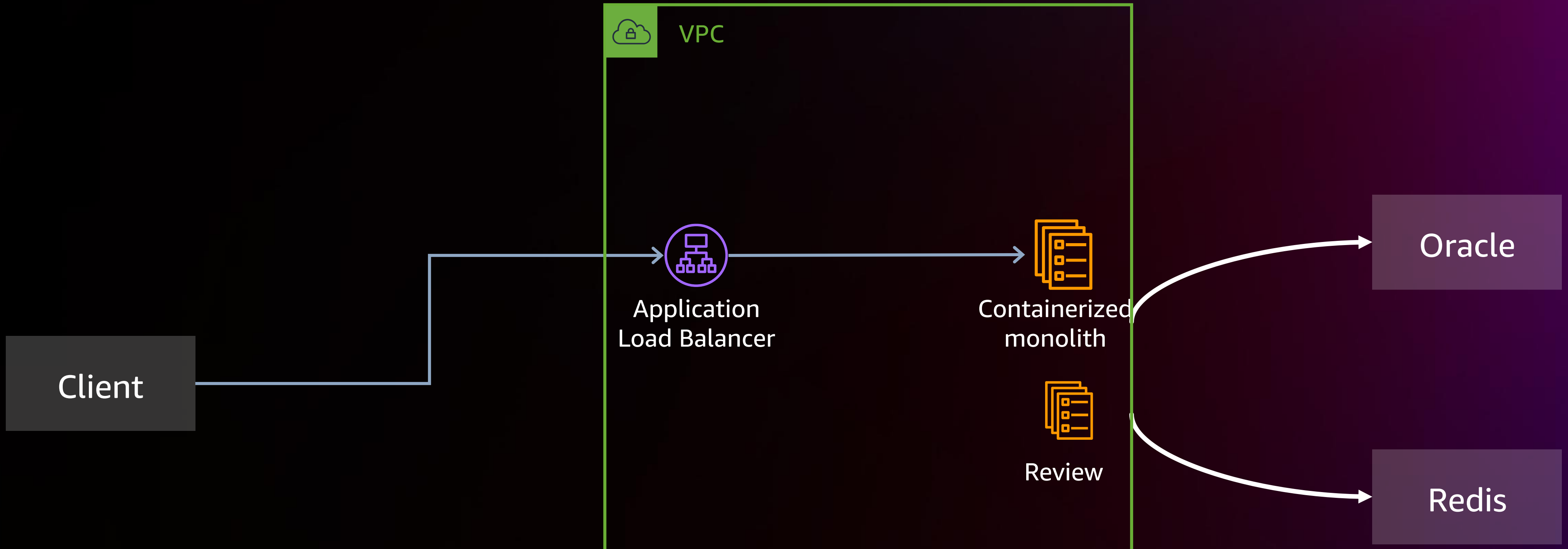
Pivoting: Whiteboarding



Pivoting: Whiteboarding

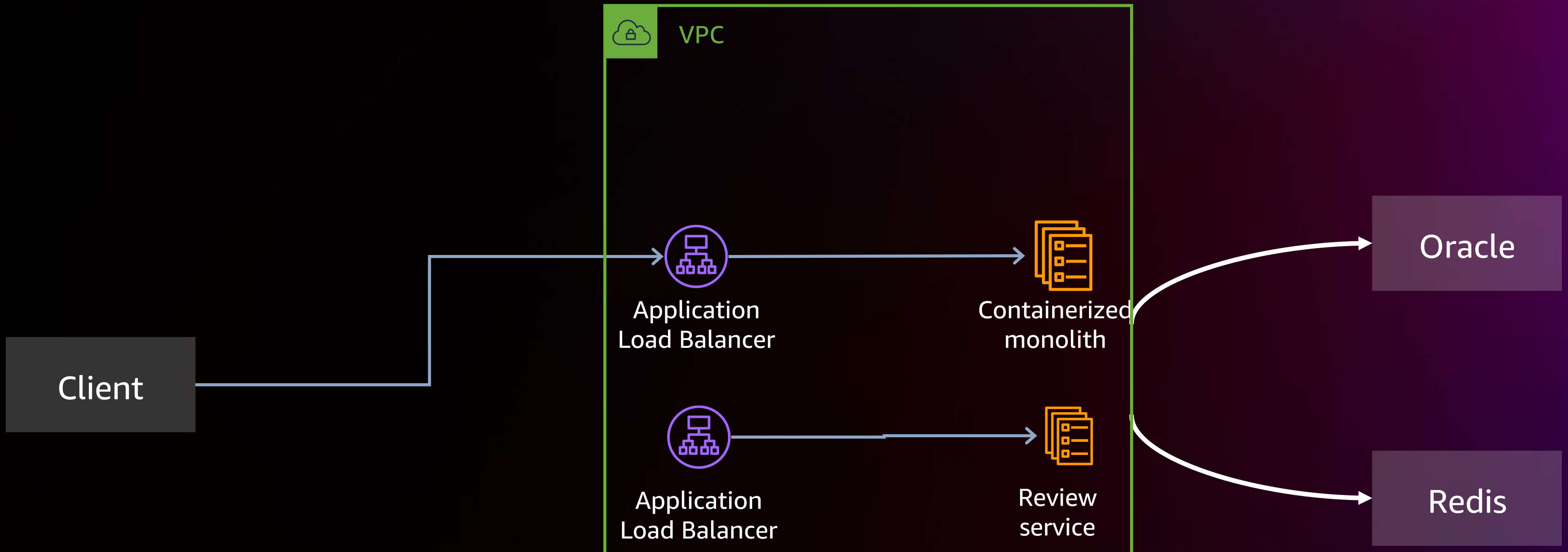


Pivoting: Whiteboarding



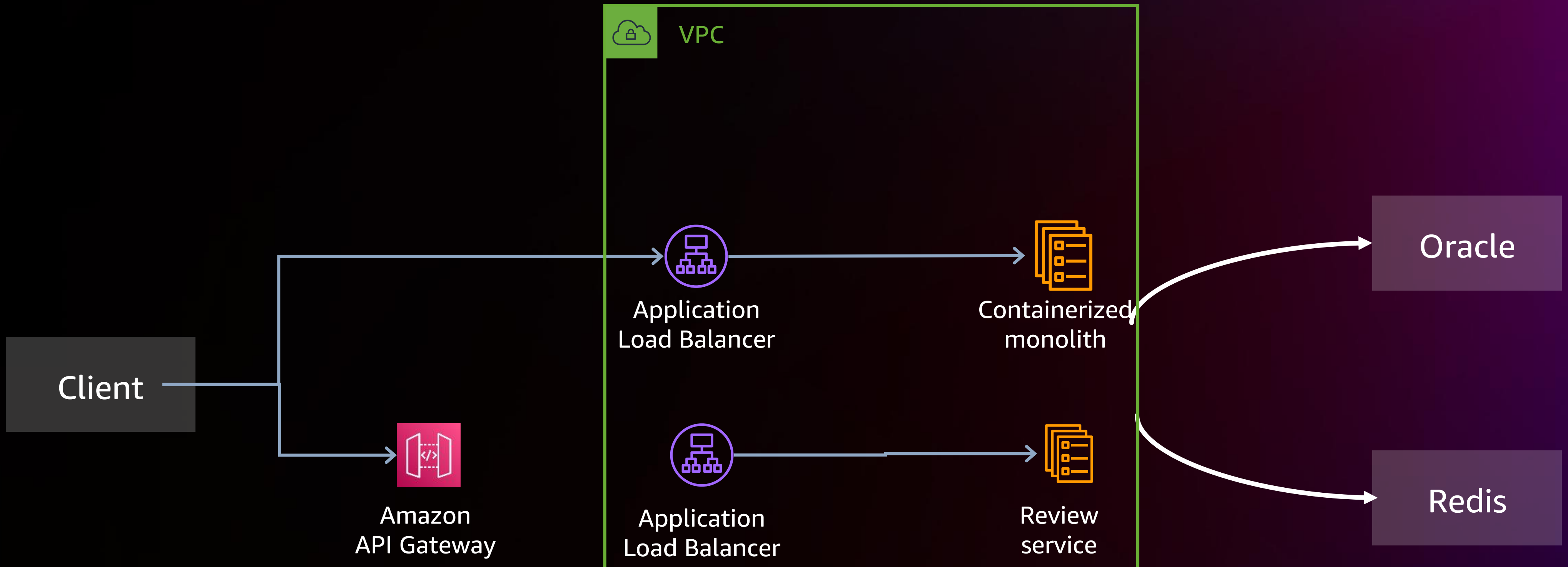
Pivoting | Scale | Optimization | Integration

Pivoting: Whiteboarding

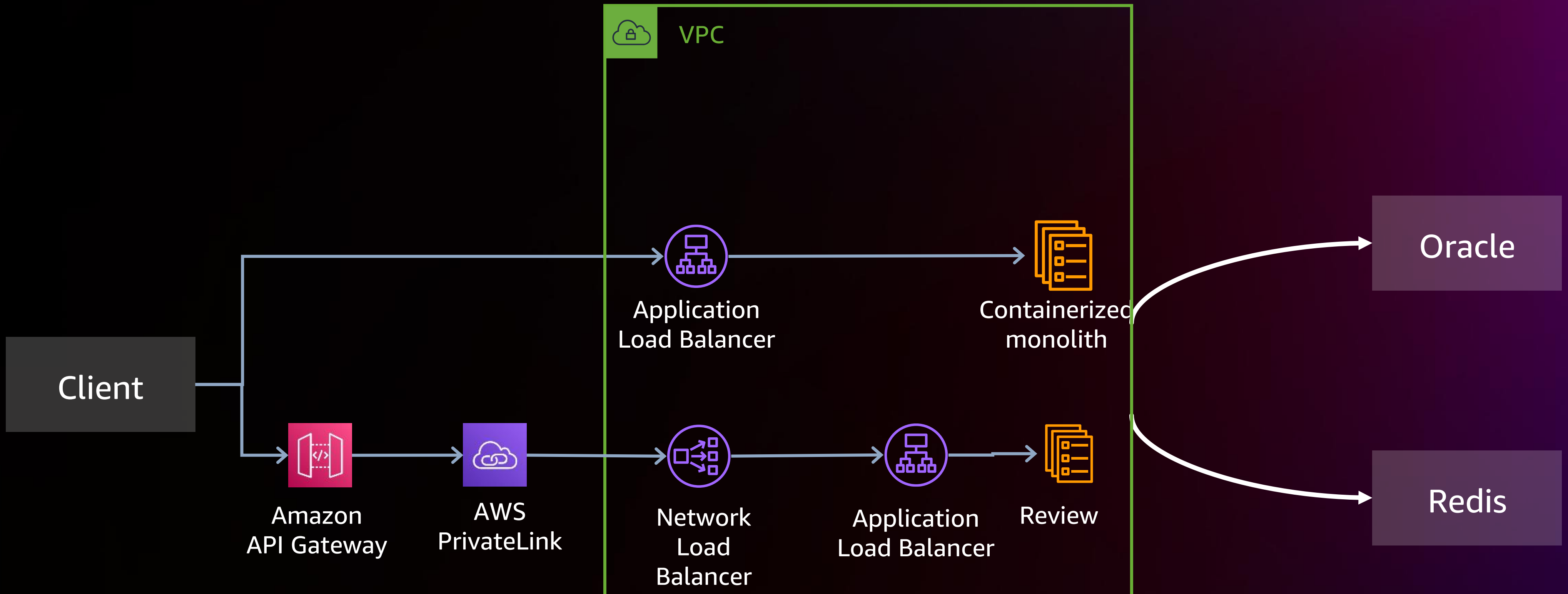


Pivoting | Scale | Optimization | Integration

Pivoting: Whiteboarding

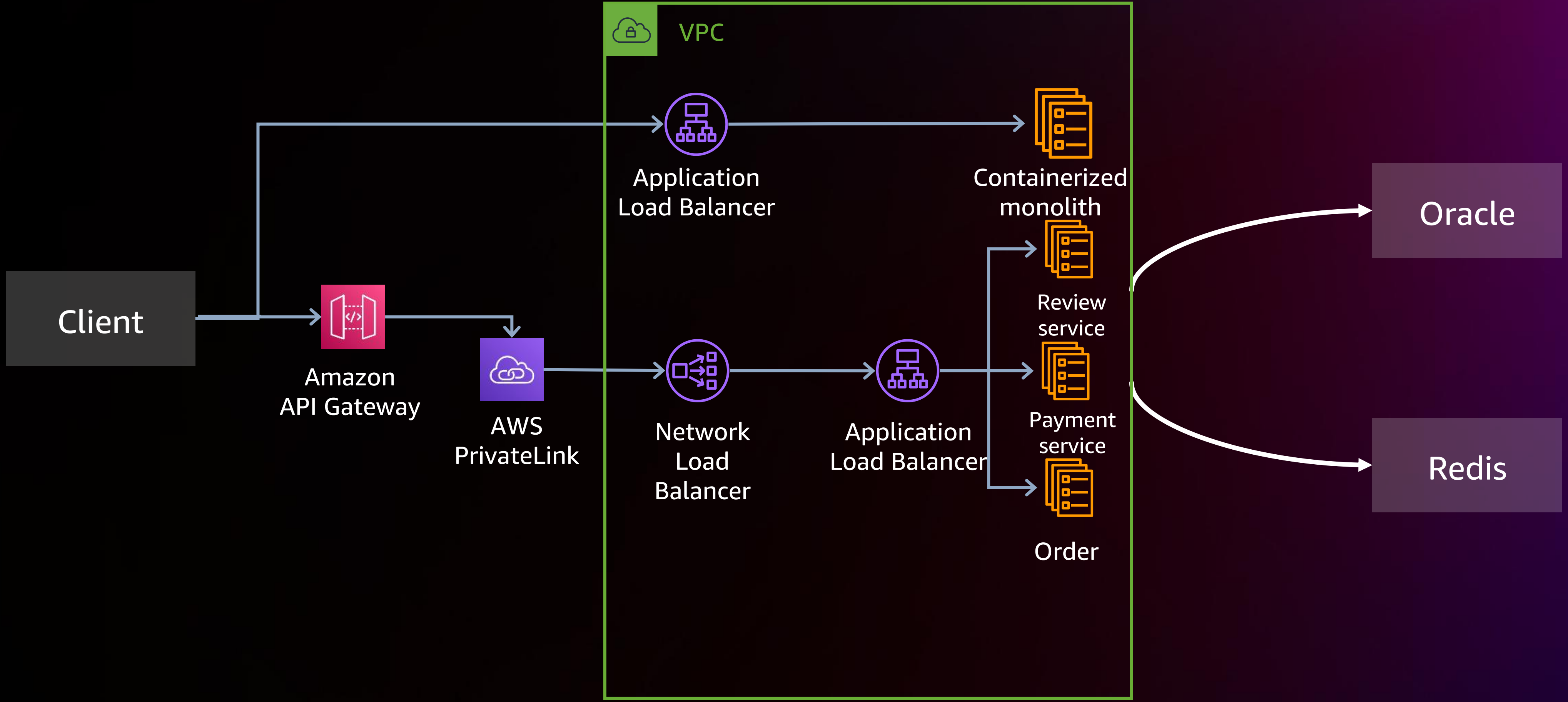


Pivoting: Whiteboarding



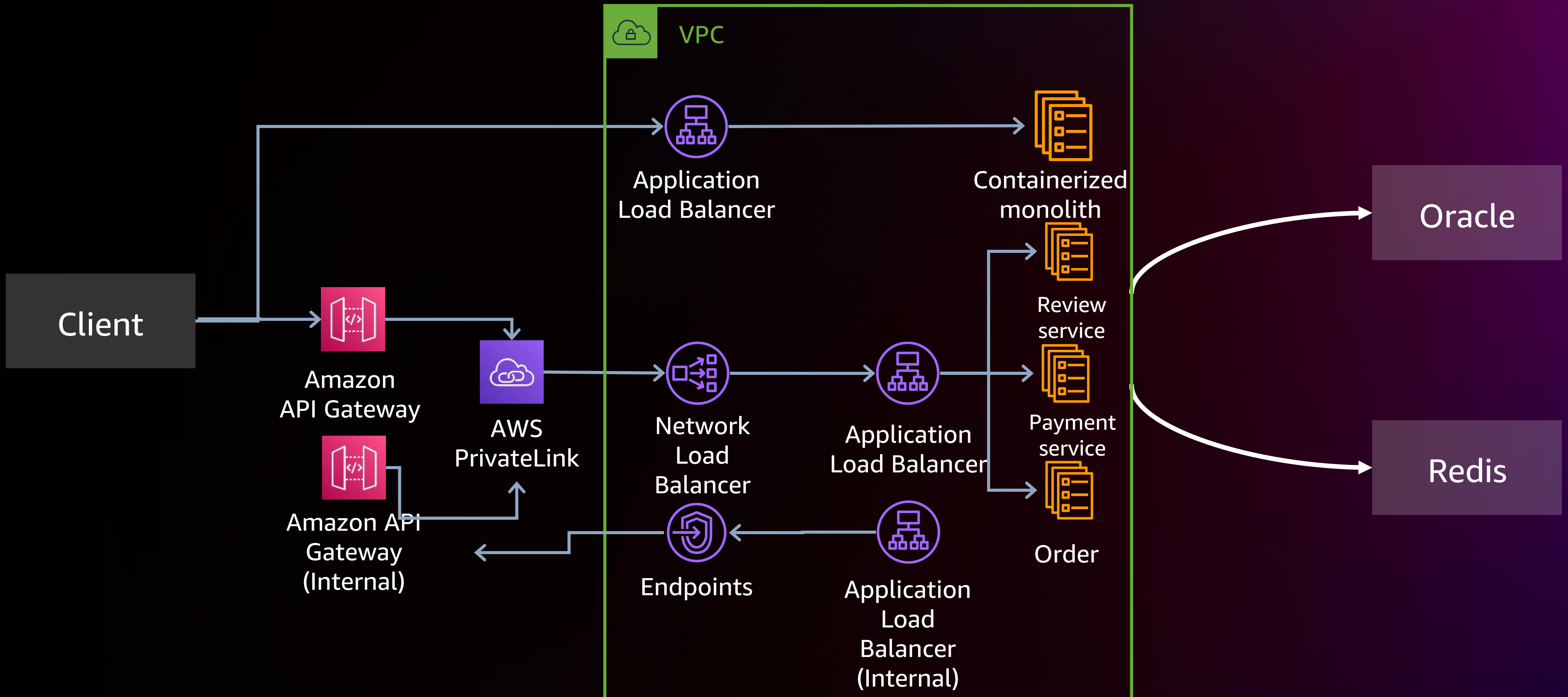
Pivoting | Scale | Optimization | Integration

Pivoting: Whiteboarding



Pivoting | Scale | Optimization | Integration

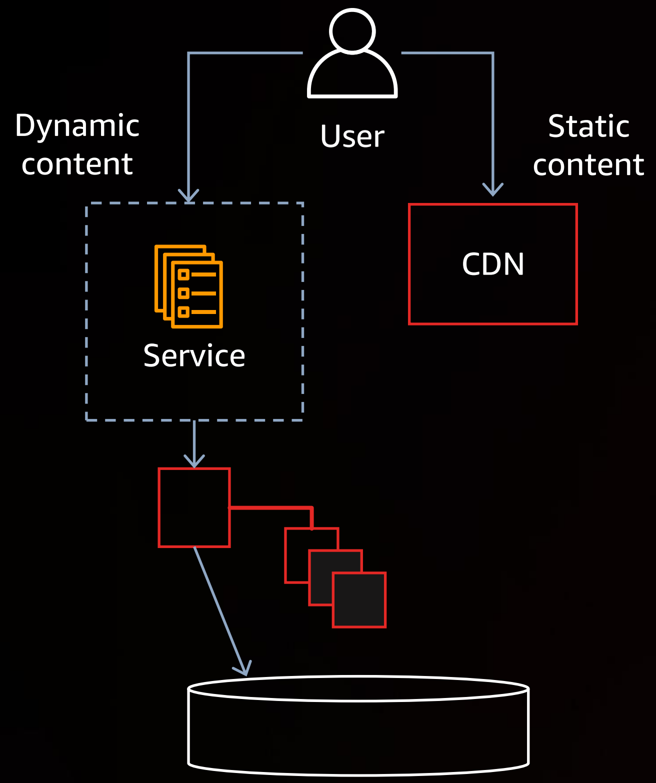
Pivoting: Whiteboarding



Pivoting | Scale | Optimization | Integration

Scalability: Growing user base

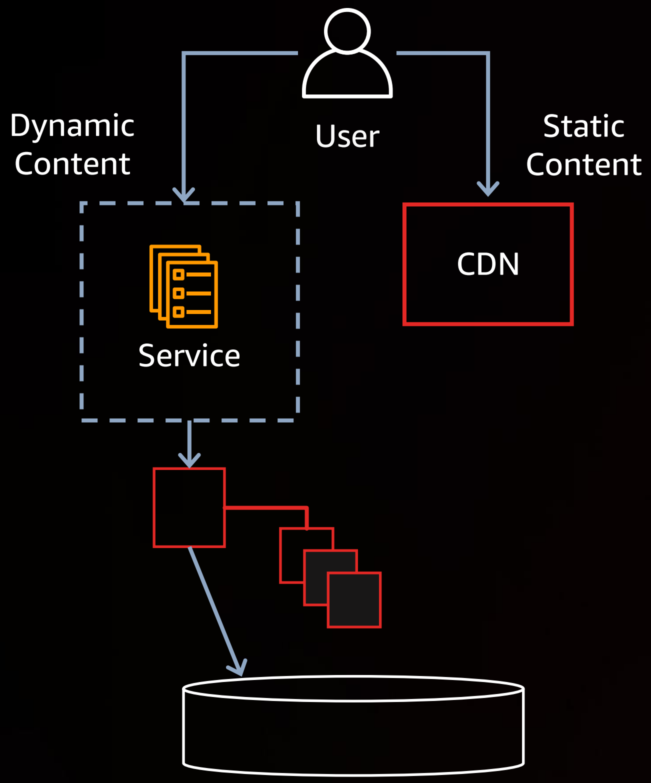
Static content hosting pattern



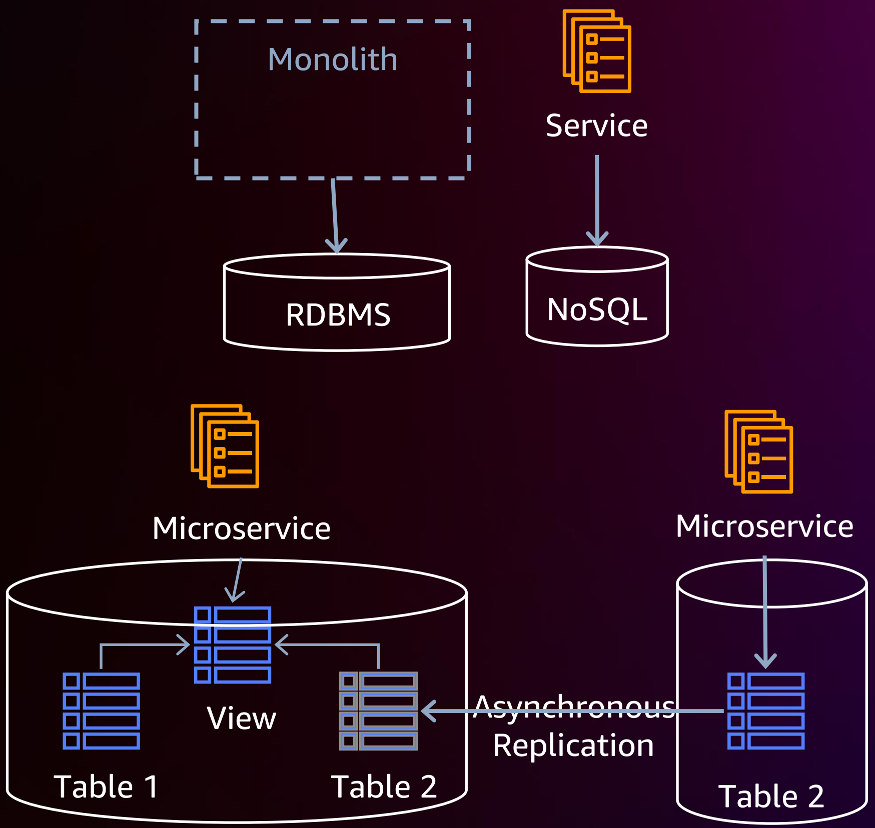
Pivoting | Scale | Optimization | Integration

Scalability: Growing user base

Static content hosting pattern



Data per service/materialized view



Pivoting | Scale | Optimization | Integration

Scalability: Whiteboarding

Cater to a growing user base with flawless performance and user experience

- Handle a larger number of user requests
- Keep it highly available and reliable as it grows
- Use caching wherever possible
- Require several sortings on write-intensive database

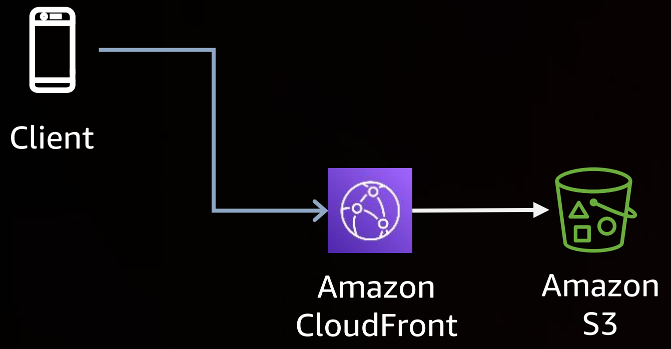
Scalability: Whiteboarding

Client

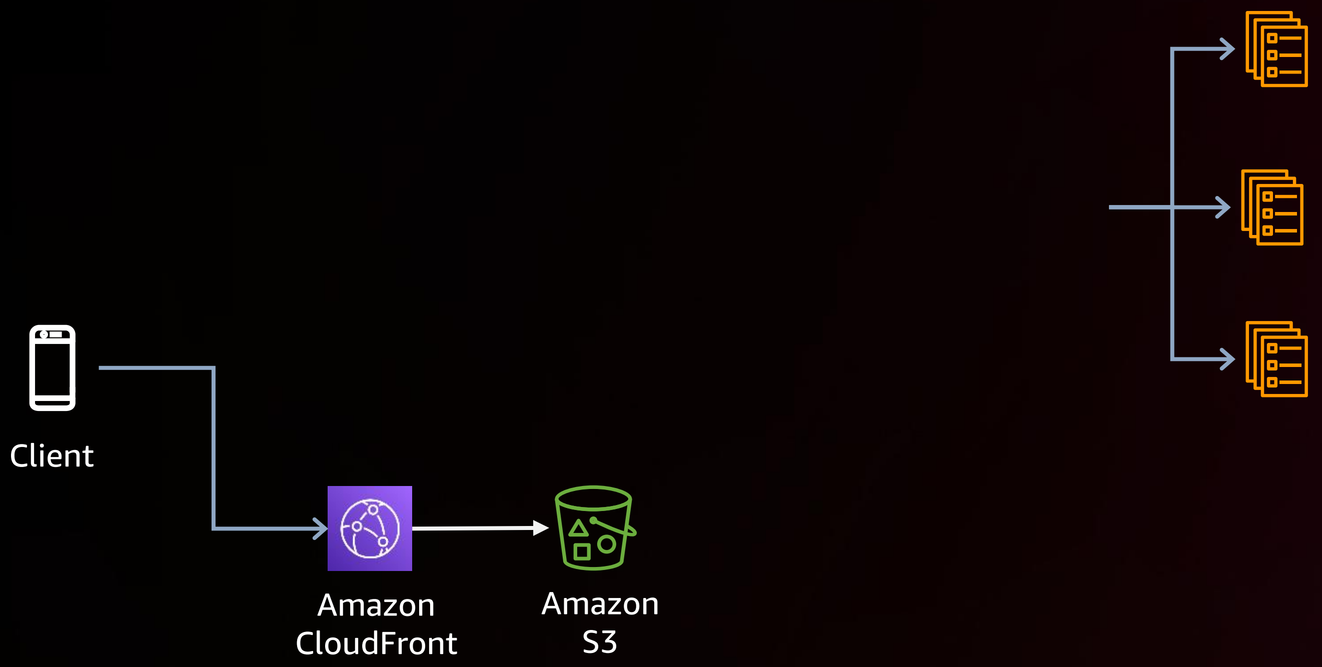
Scalability: Whiteboarding

Client
Amazon S3

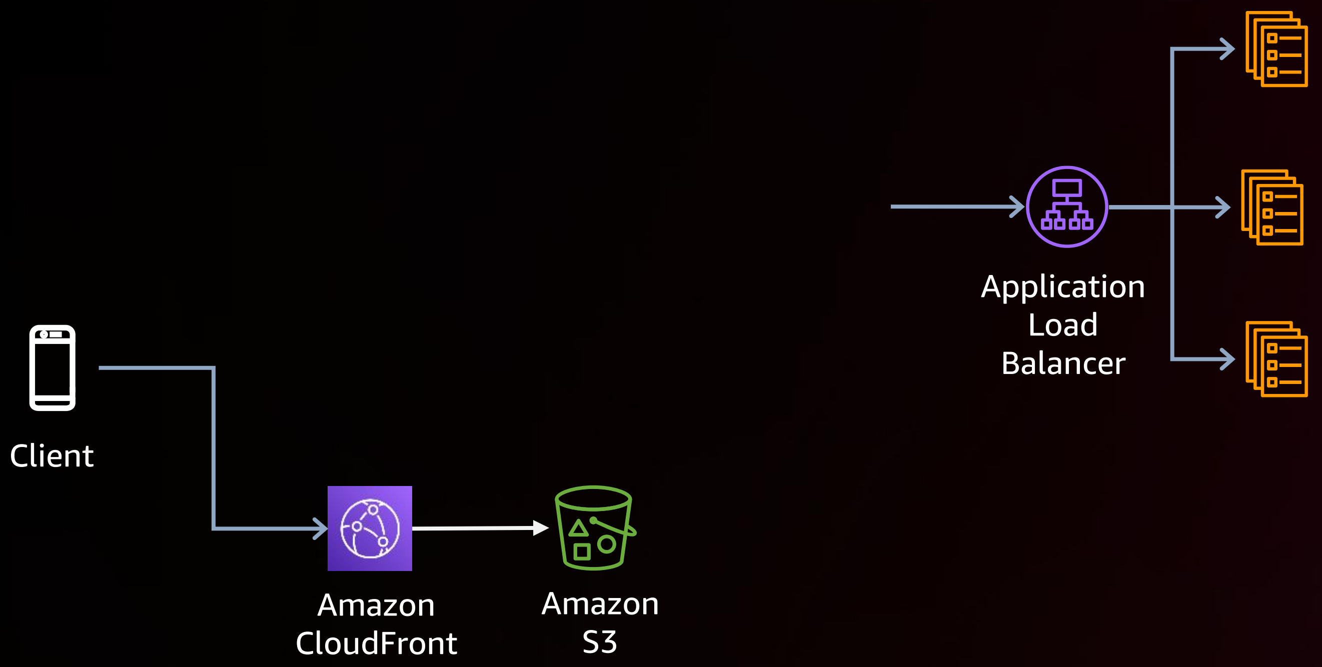
Scalability: Whiteboarding



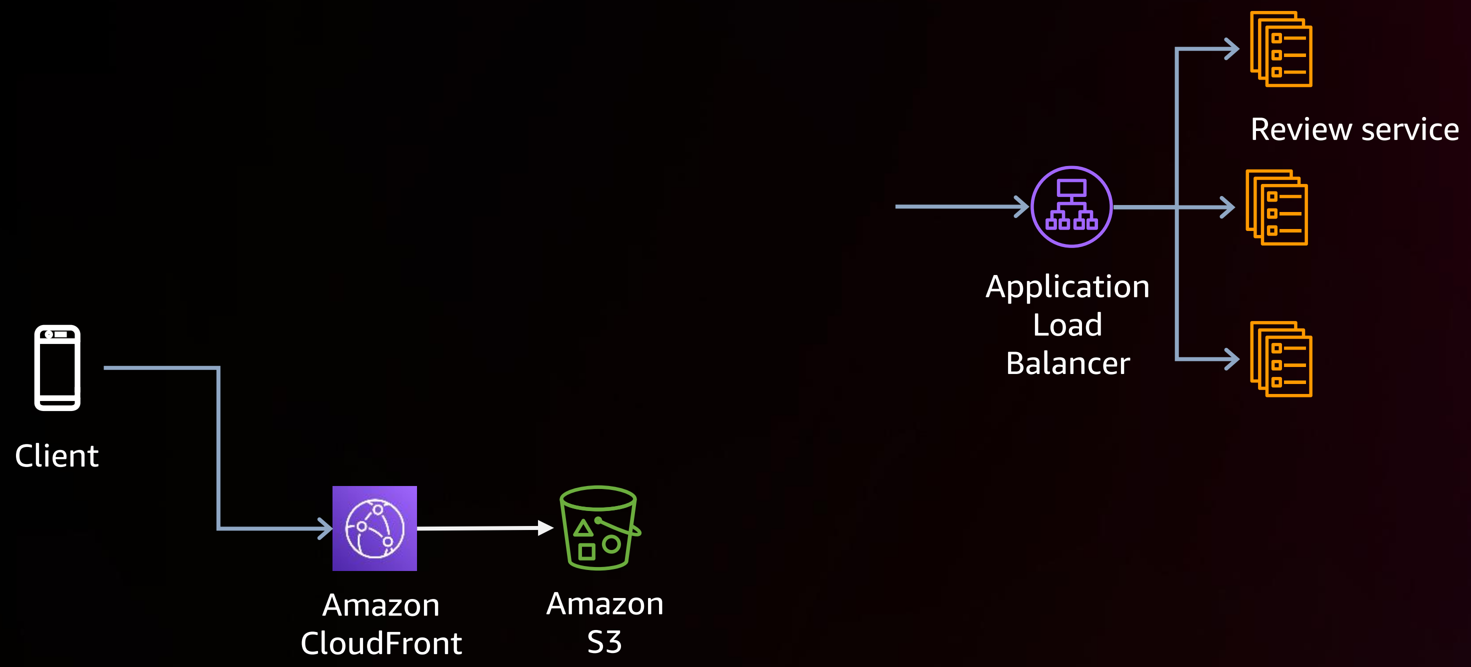
Scalability: Whiteboarding



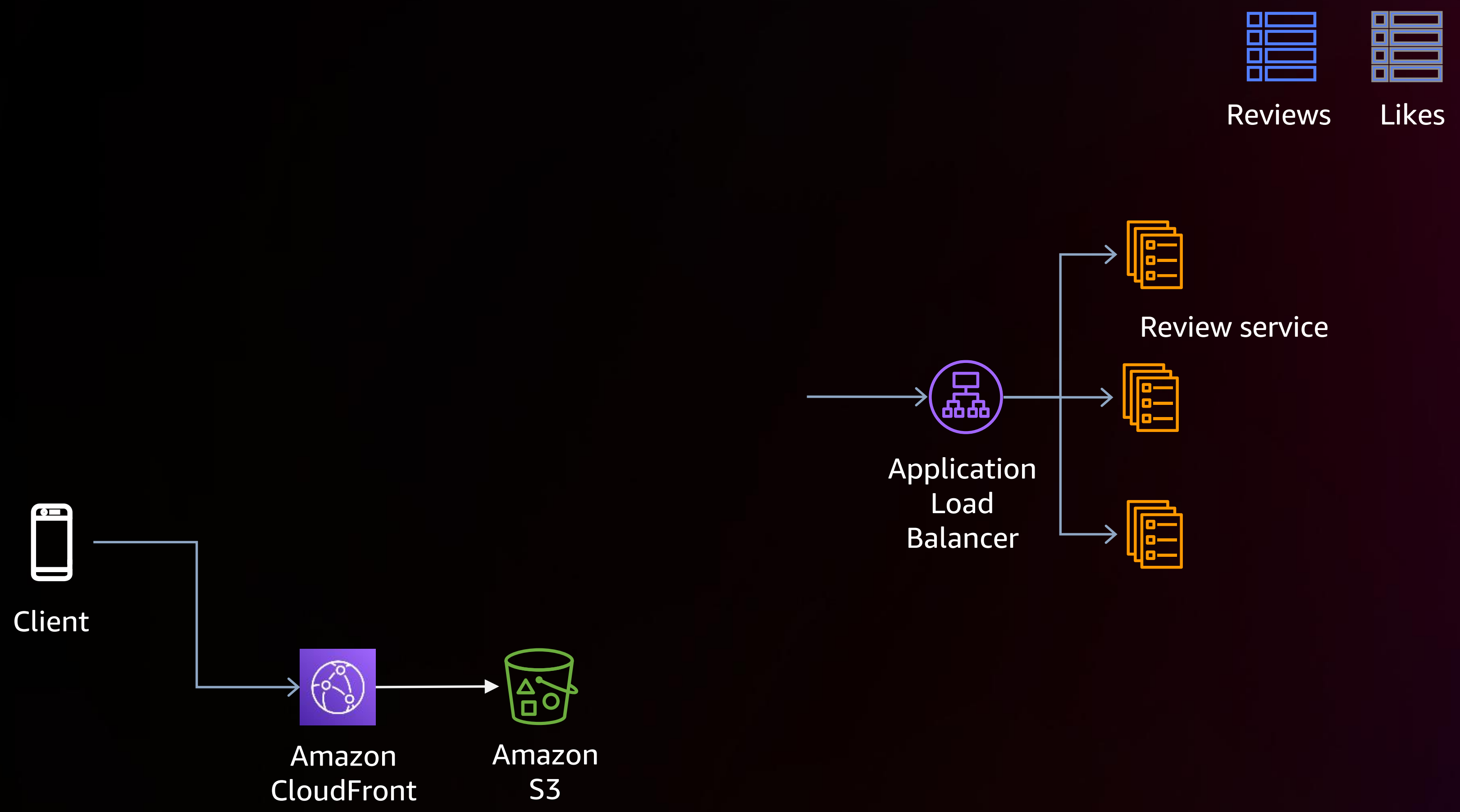
Scalability: Whiteboarding



Scalability: Whiteboarding



Scalability: Whiteboarding



AWS purpose-built databases

Relational	Key-value	Document	In-memory	Graph	Time-series	Ledger	Wide column
Aurora, Amazon RDS	DynamoDB	Amazon DocumentDB	ElastiCache	Neptune	Timestream	Amazon QLDB	Amazon Keyspaces

Scalability: Whiteboarding

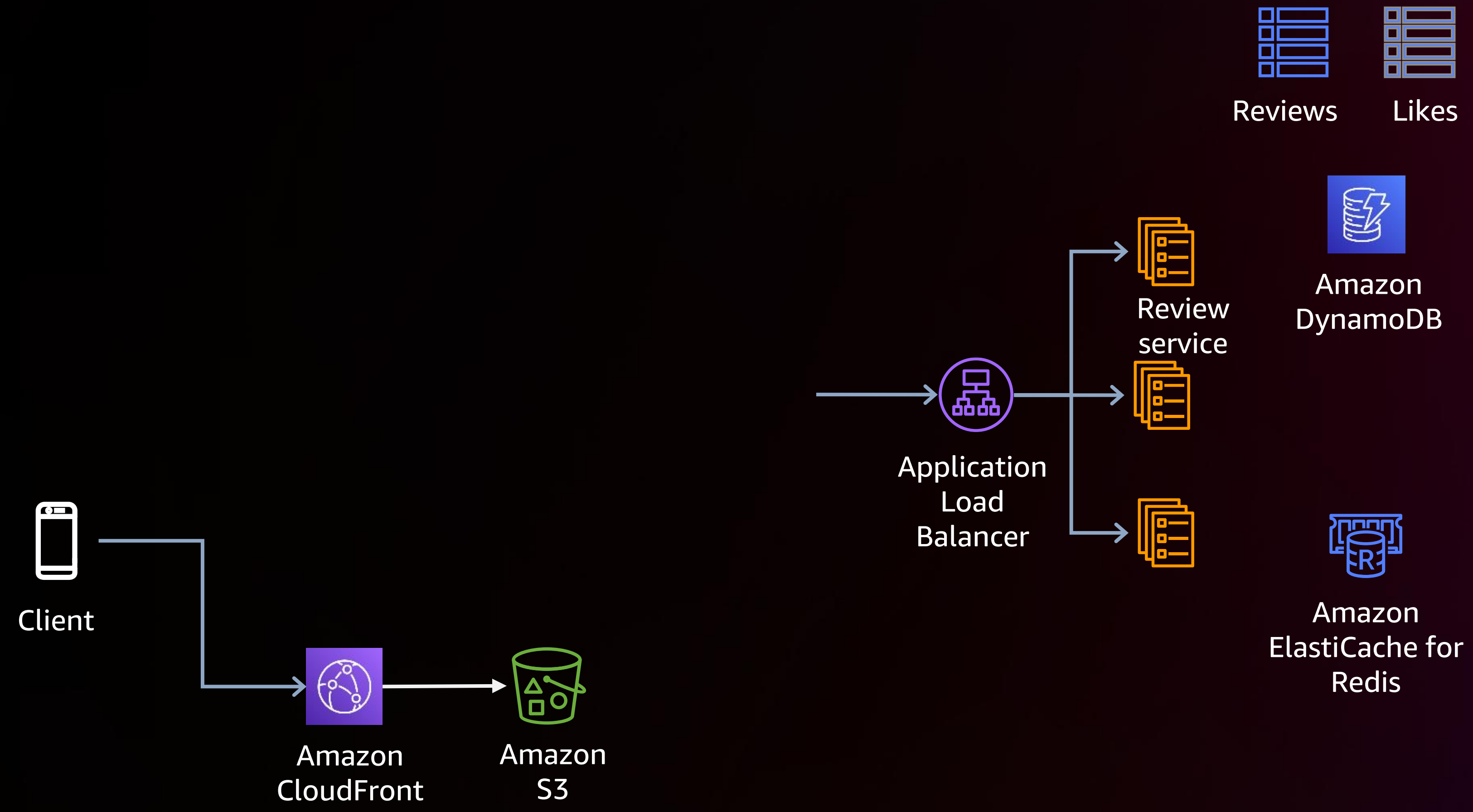
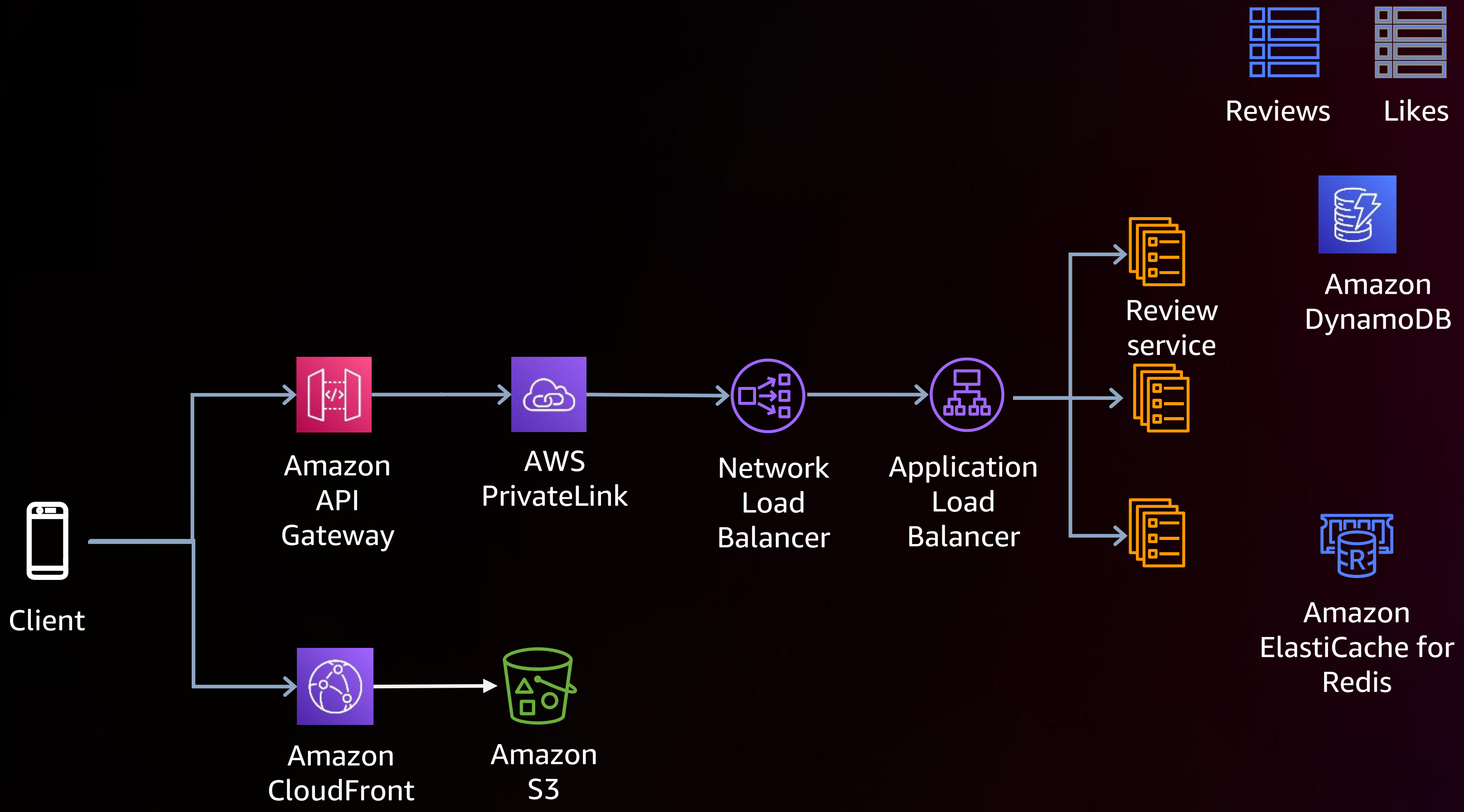


Table	Partition Key
Review	timestamp
Like	timestamp

GSI	Partition Key	Sort Key
creation_day-review_id-index	creation_day	review_id
sku_code-best_order-index	sku_code	best_order
sku_code-latest_order-index	sku_code	latest_order

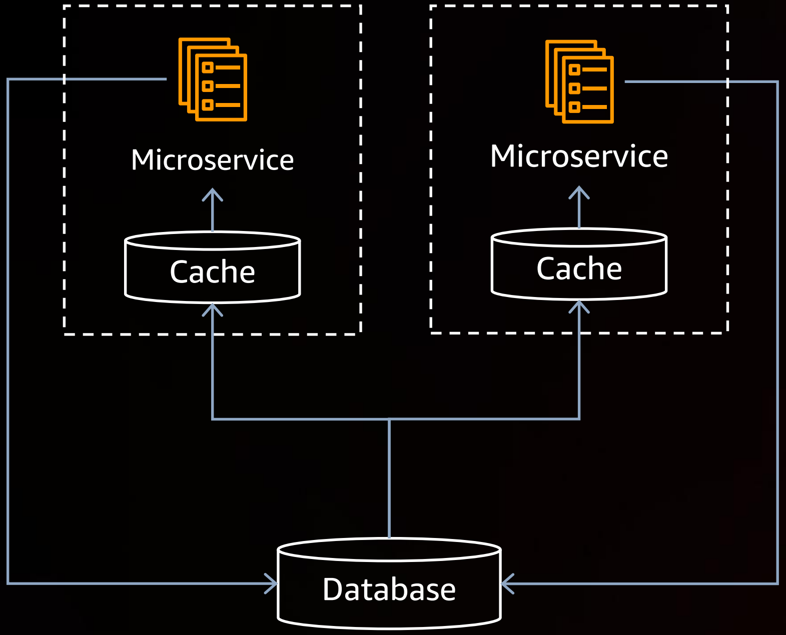
Attribute	Format
best_order	number_of_likes#image#creation_date
latest_order	date

Scalability: Whiteboarding

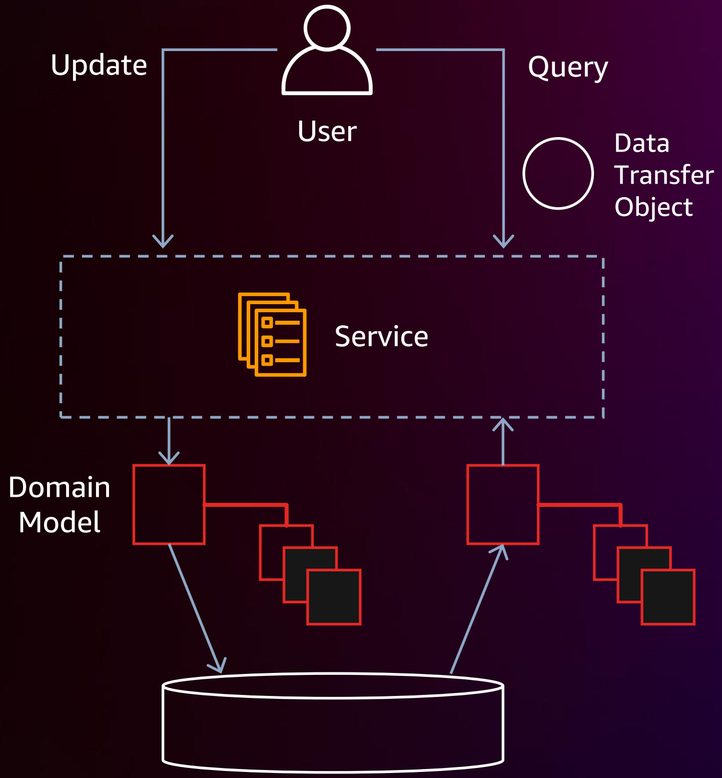


Performance optimization: Effectiveness

Caching Data
(Spatial-Locality)

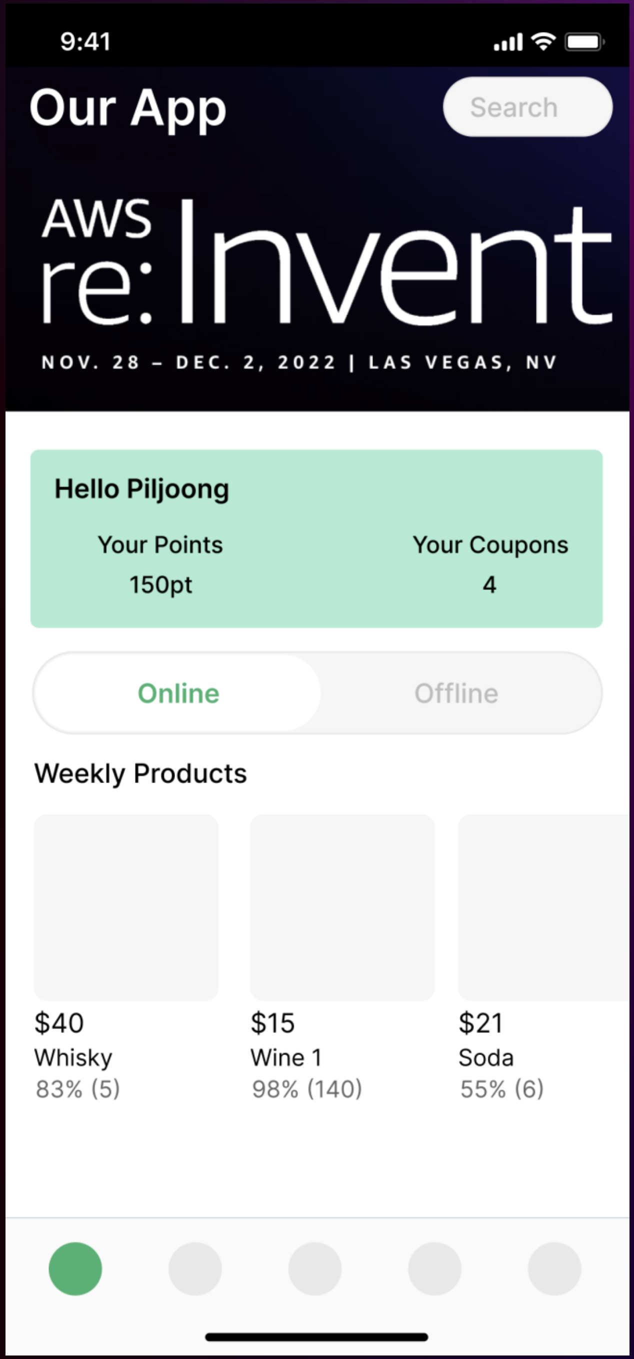


CQRS



Pivoting | Scale | Optimization | Integration

Main page


9:41

Our App

Search

AWS re:Invent

NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

Hello Piljoong

Your Points	Your Coupons
150pt	4

Online

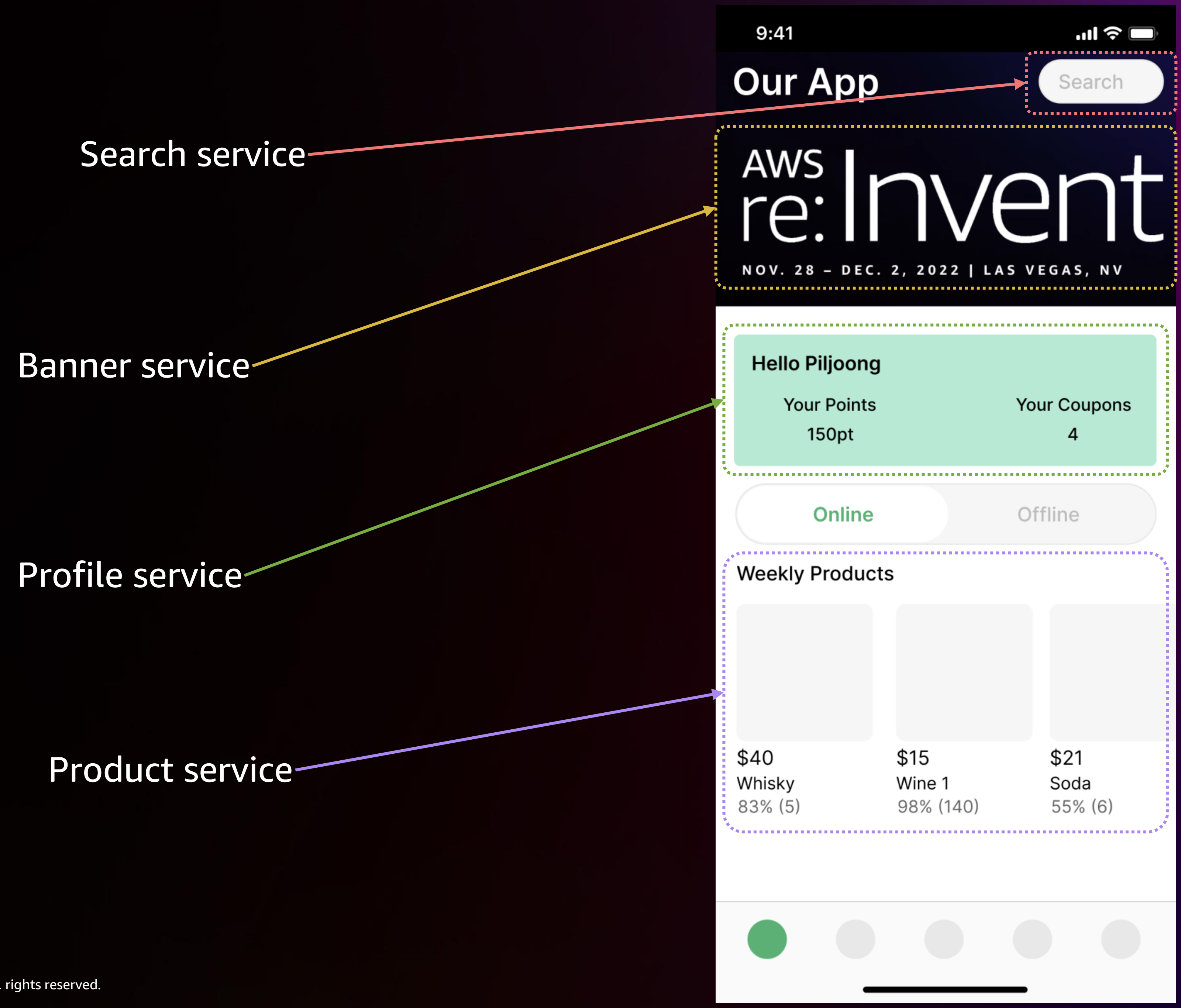
Offline

Weekly Products

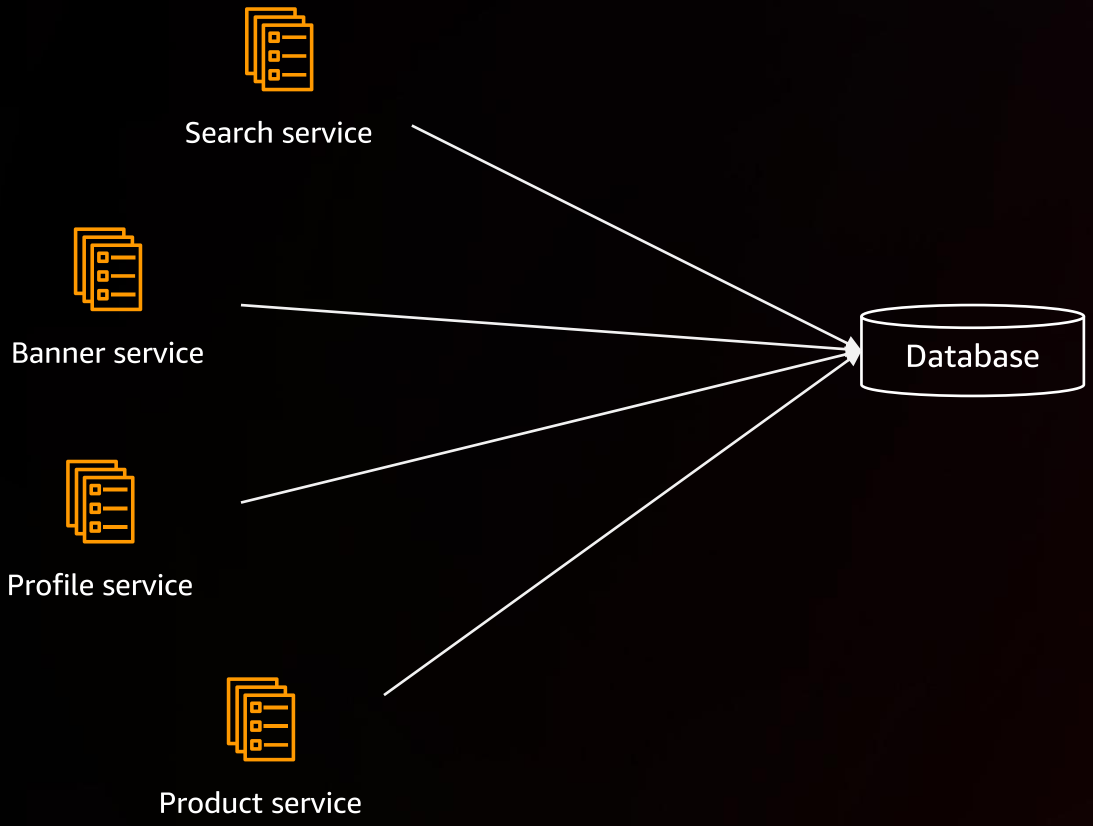
Price	Product	Rating
$40	Whisky	83% (5)
$15	Wine 1	98% (140)
$21	Soda	55% (6)

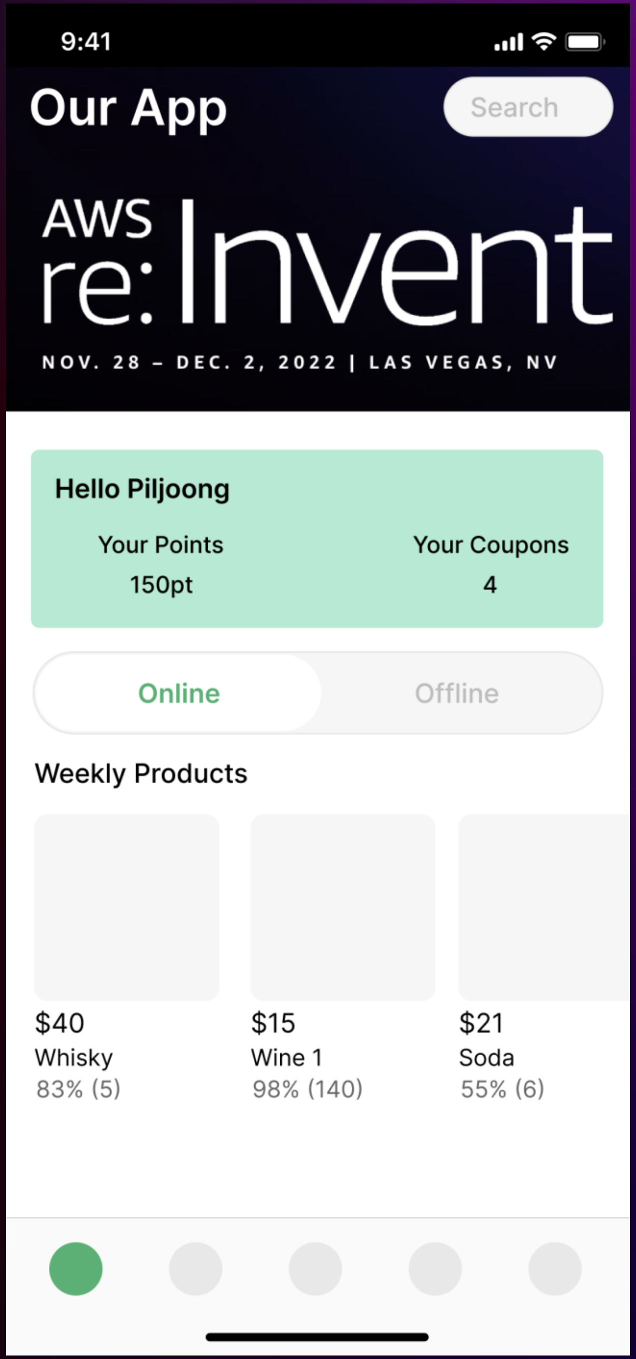
> Mobile ecommerce service

Small independent services



Initial Version





Hello Piljoong

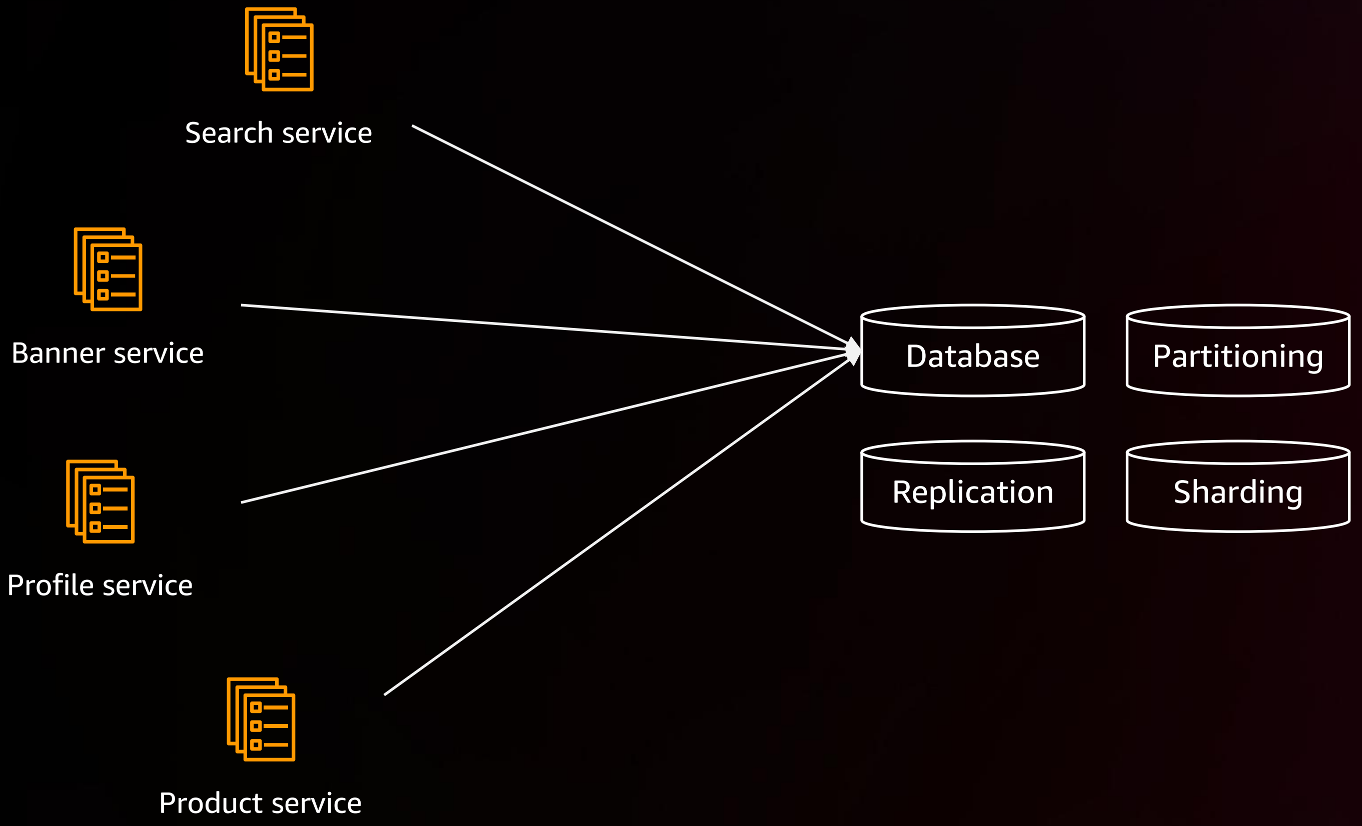
Your Points	Your Coupons
150pt	4

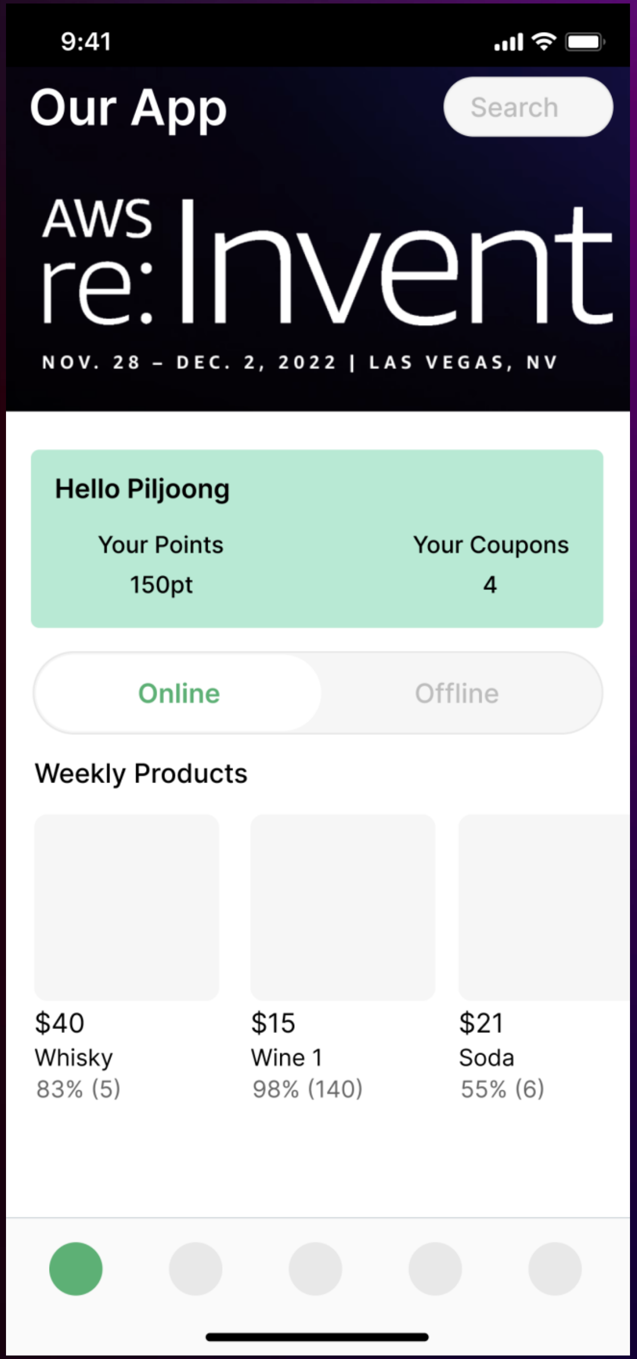
Weekly Products

$40	$15	$21
Whisky	Wine 1	Soda
83% (5)	98% (140)	55% (6)

> Architecture

Traditional Optimization





> Mobile ecommerce service

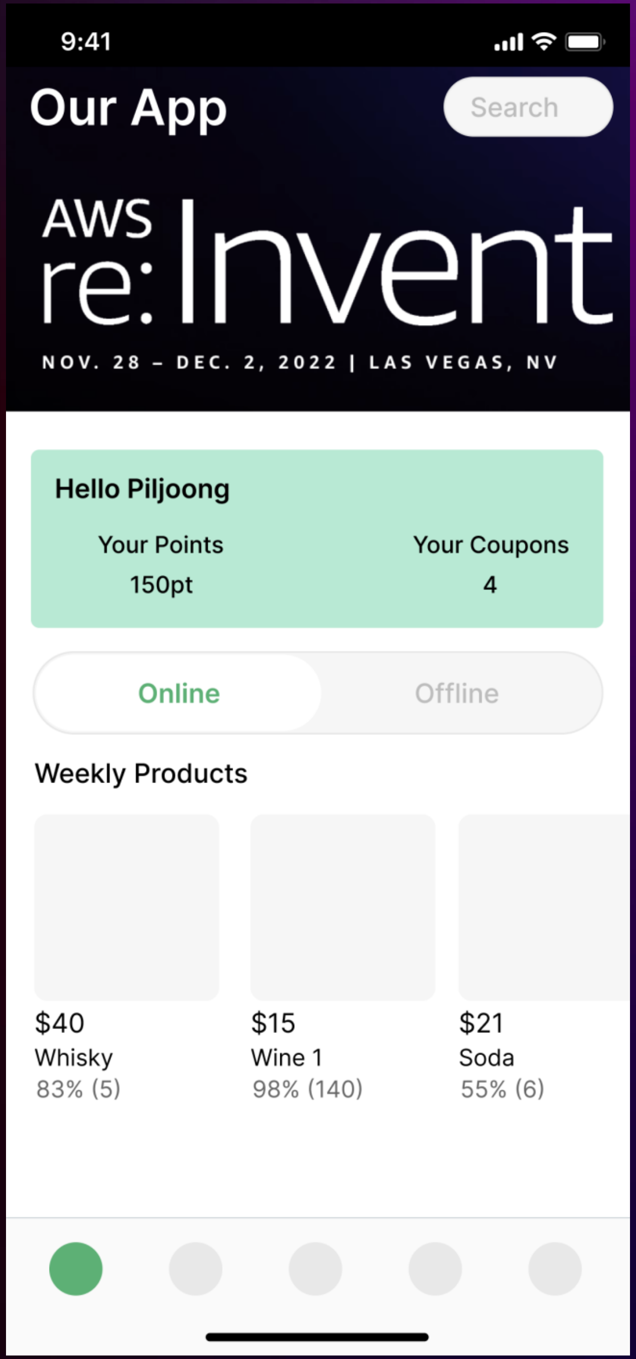
Functions

Search service — Returns products that match a keyword

Banner service — Displays notices and promotions

Profile service — A user's information

Product service — Limited-time offer with # of reviews & likes



Functions
- Product service

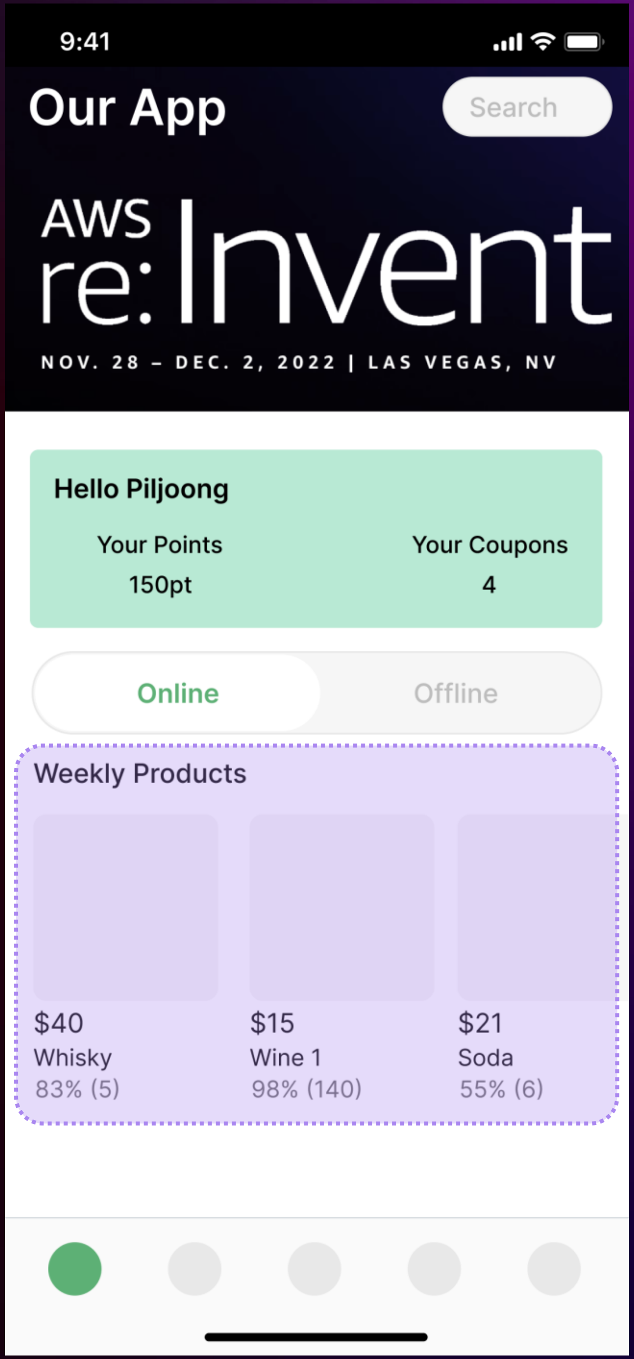
Search service — Returns products that match a keyword

Banner service — Displays notices and promotions

Profile service — A user's information

Product service — Limited-time offer with # of reviews & likes

- Display up to 20 products on sale this week
- Each product has # of reviews & likes
- Reviews and likes data are fed from a separate Review service
- Call API every time loading the page
- Run an aggregation query on multiple entities and services


9:41

Our App

Search

AWS re:Invent

NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

Hello Piljoong

Your Points	Your Coupons
150pt	4

Online | Offline

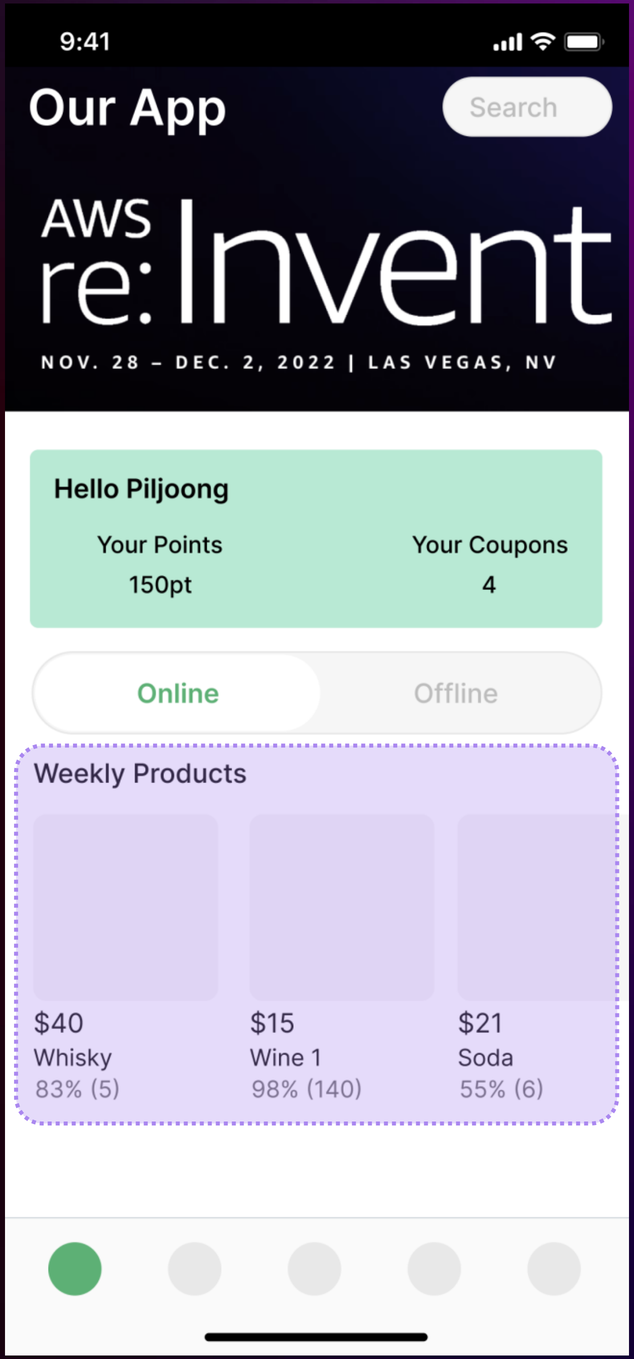
Weekly Products

$40	$15	$21
Whisky	Wine 1	Soda
83% (5)	98% (140)	55% (6)

> Data access pattern

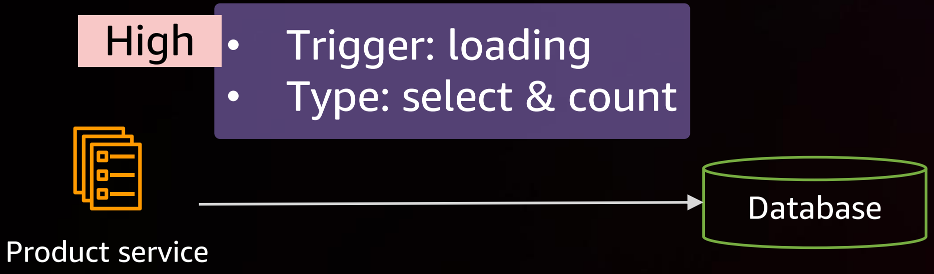
Trigger & Operation Type

Search service	Returns products that match a keyword	• Trigger: search • Type: search	Low
Banner service	Displays notices and promotions	• Trigger: loading • Type: select	Low
Profile service	A user's information	• Trigger: loading • Type: select & count	Mid
Product service	Limited-time offer with # of reviews & likes	• Trigger: loading • Type: select & count	High


9:41
Our App
Search
AWS re:Invent
NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV
Hello Piljoong
Your Points 150pt
Your Coupons 4
Online Offline
Weekly Products
$40 Whisky 83% (5)
$15 Wine 1 98% (140)
$21 Soda 55% (6)

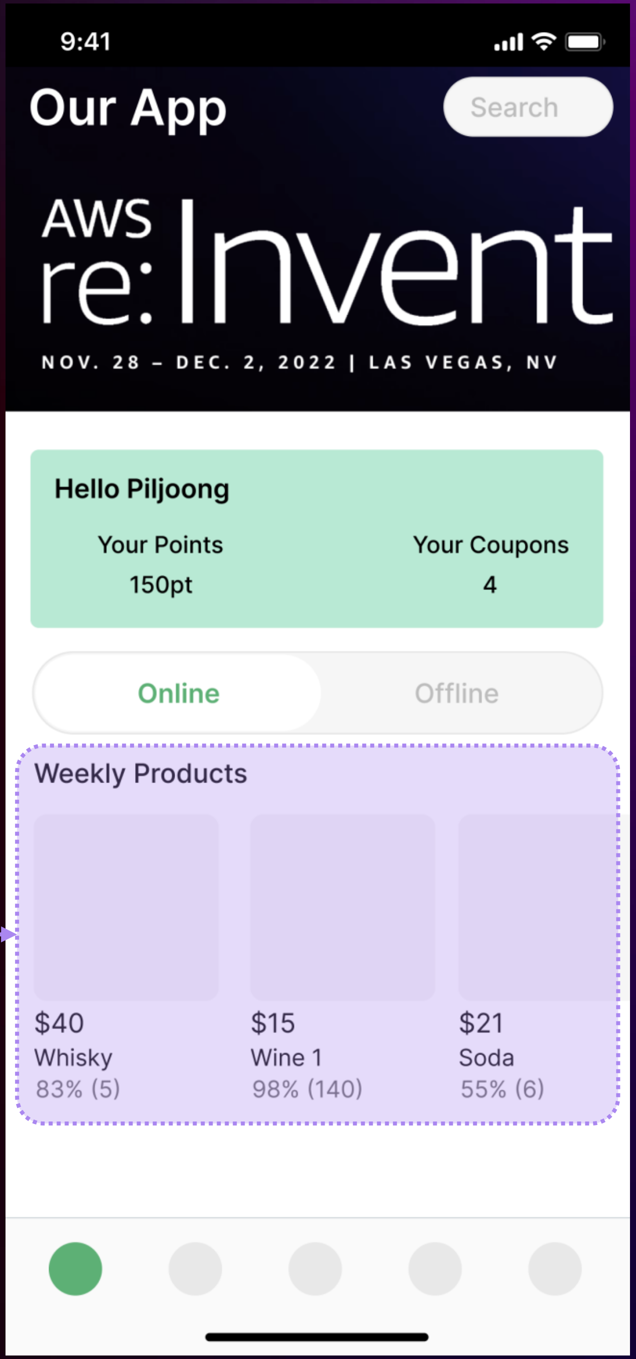
> Product service

The Target



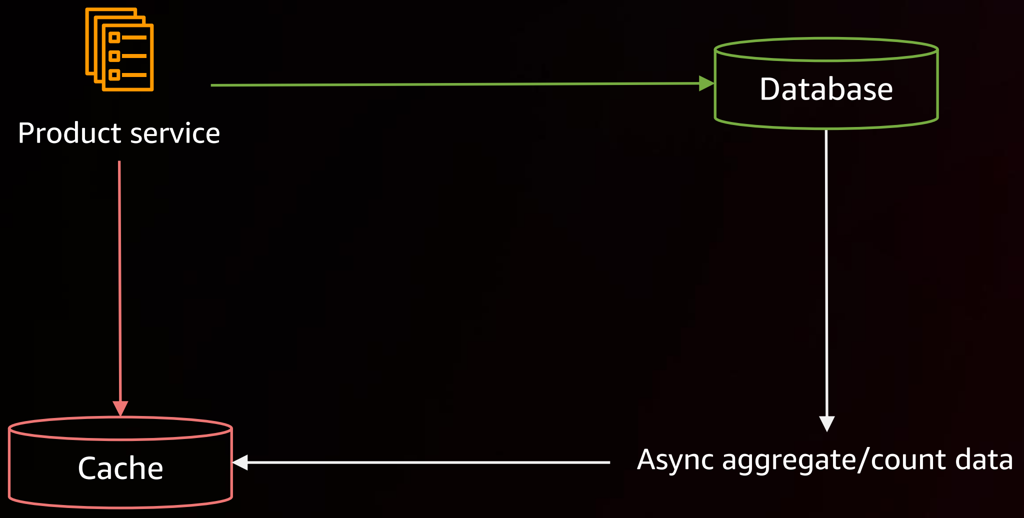
Weekly products

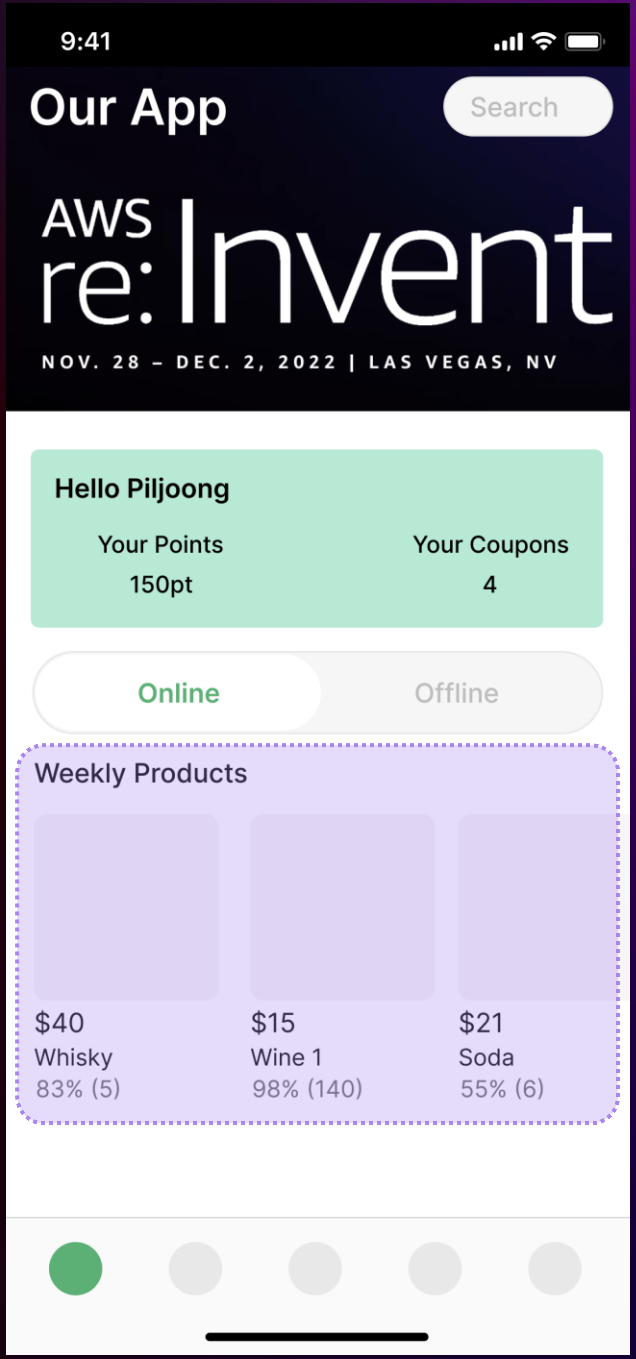
- Display up to 20 products on sale this week
- Each product has **# of reviews & likes**
- Reviews and likes **data are fed from a separate Review service**
- Call API every time loading the page
- Run **an aggregation query on multiple entities and services**



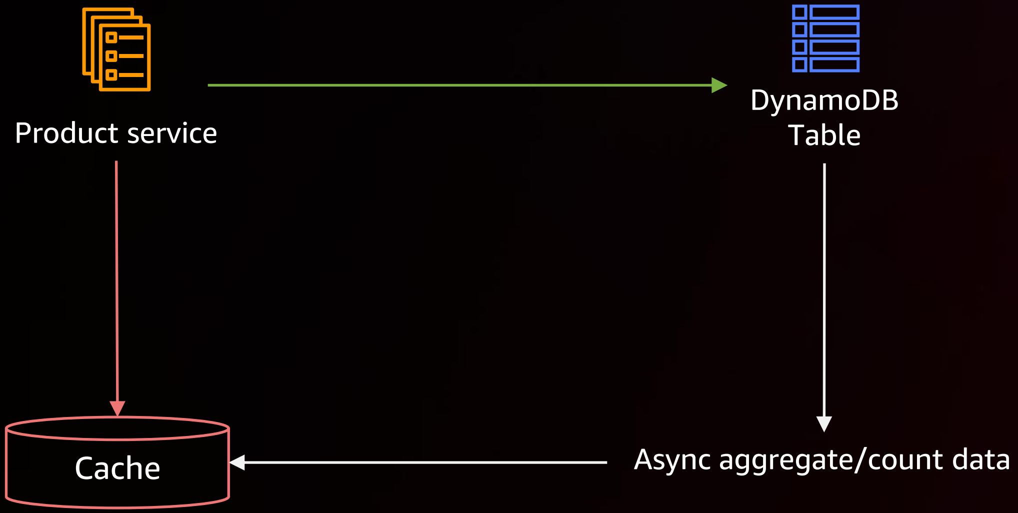
> Product service

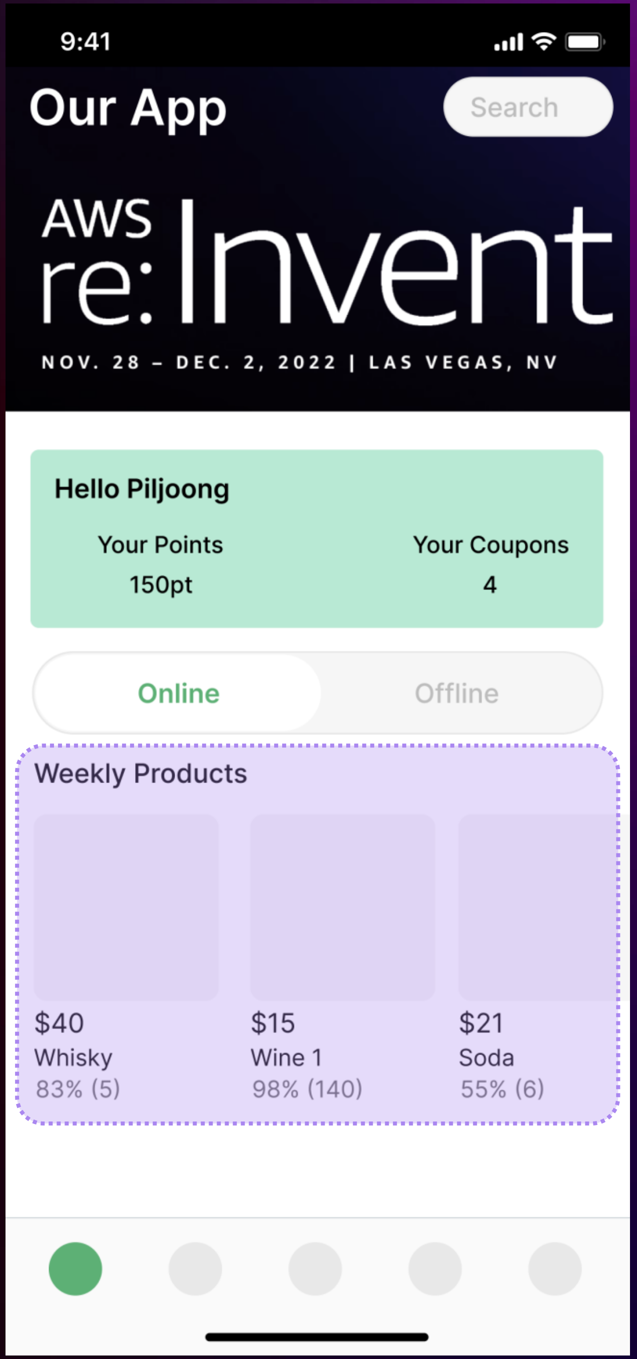
Optimization



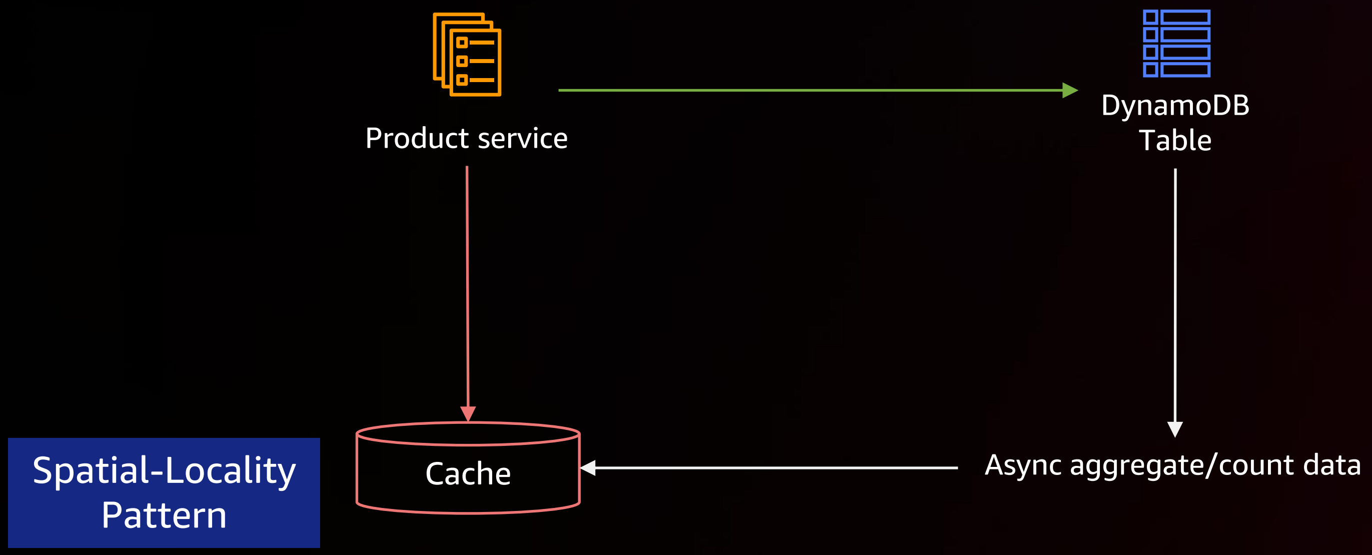


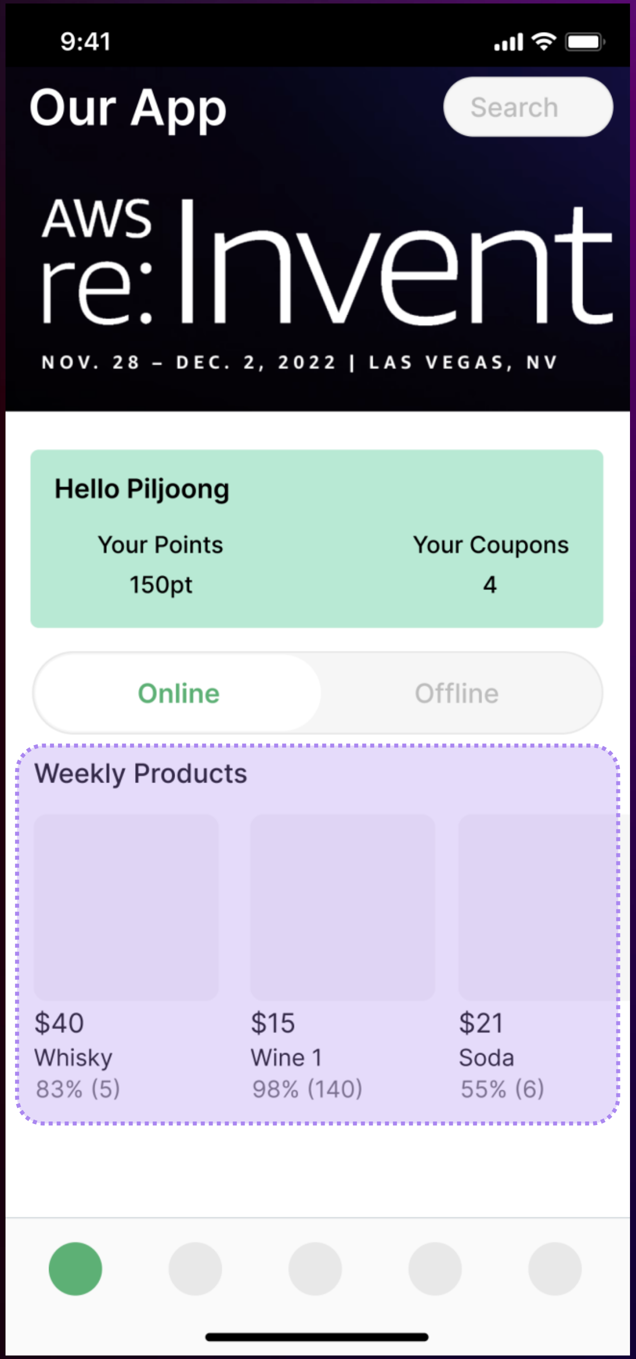
Optimization



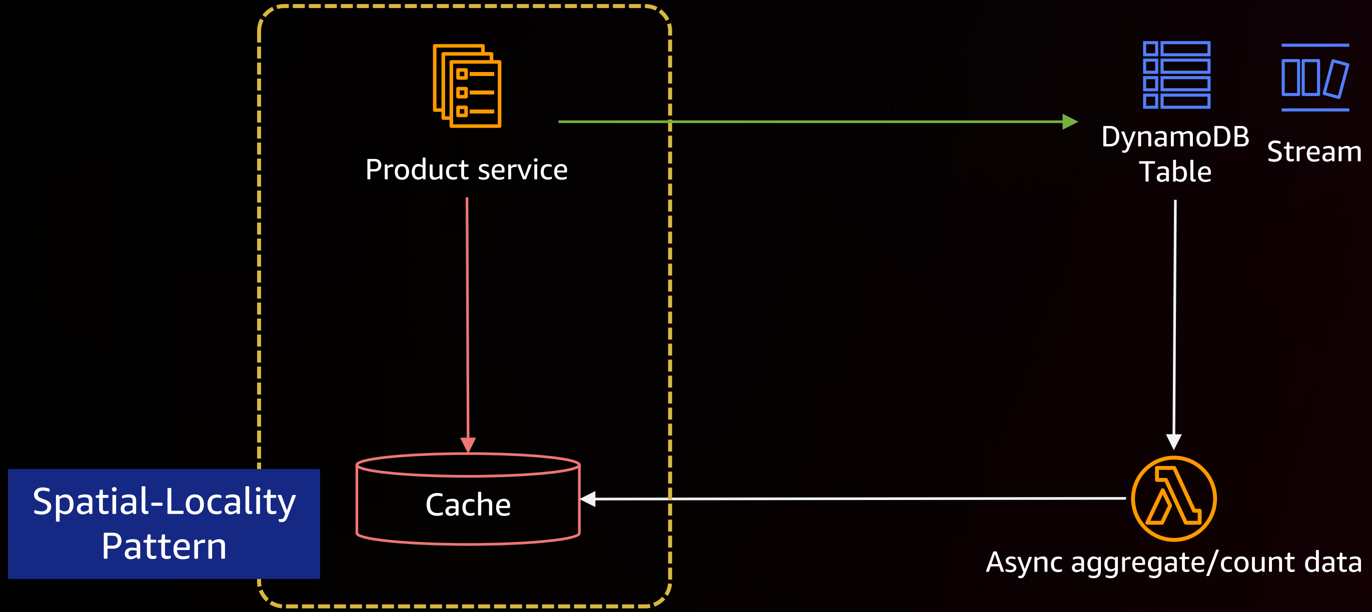


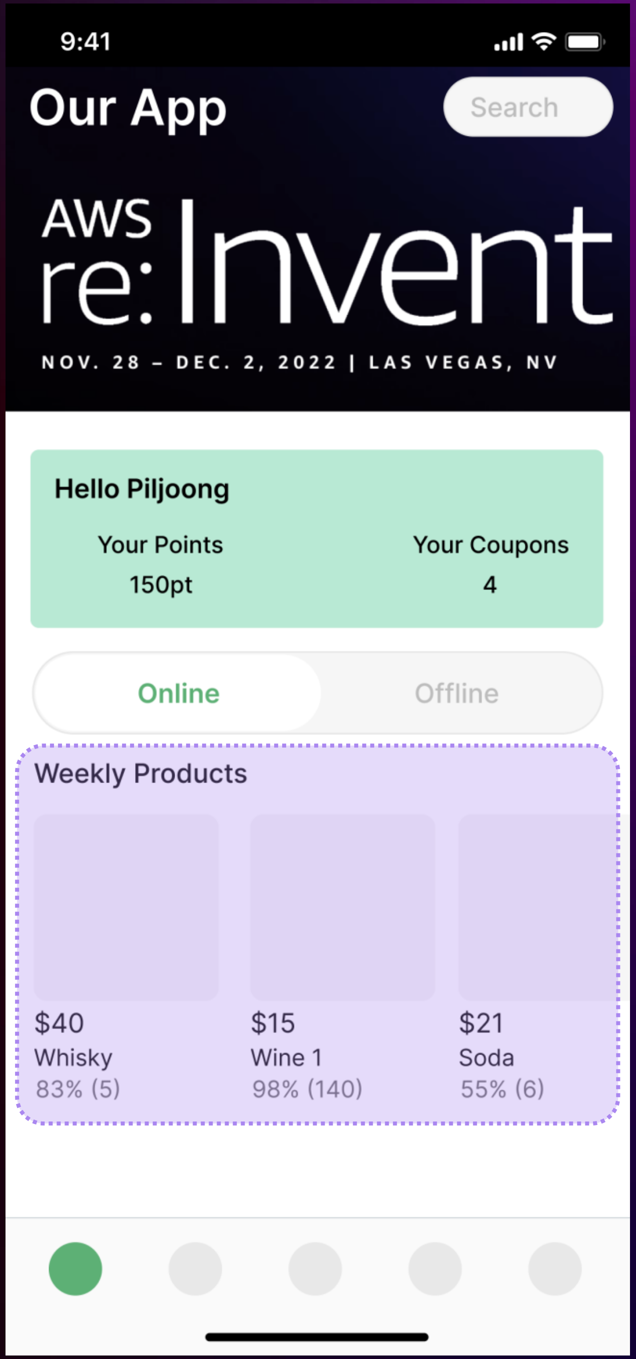
Optimization





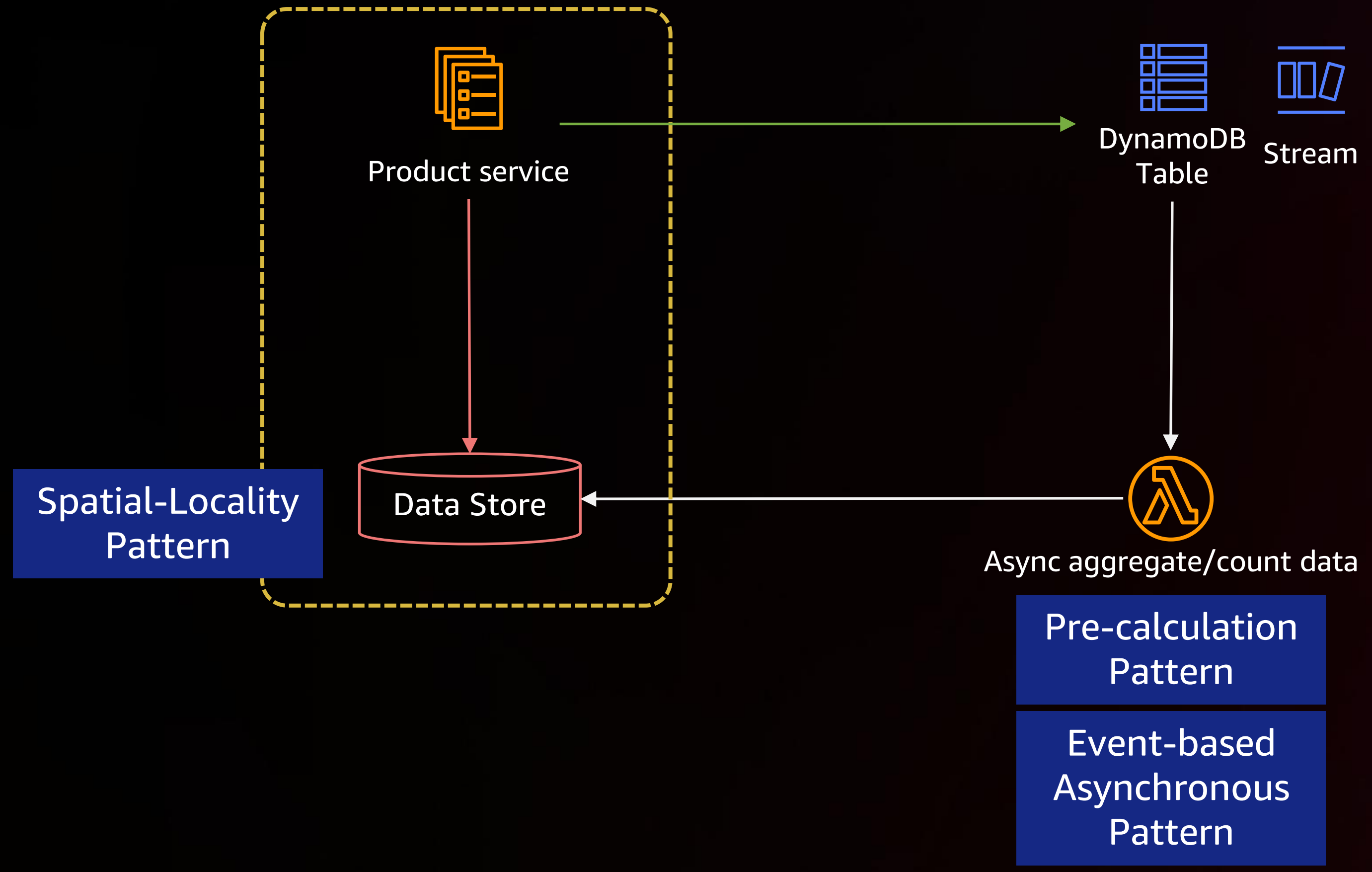
Optimization

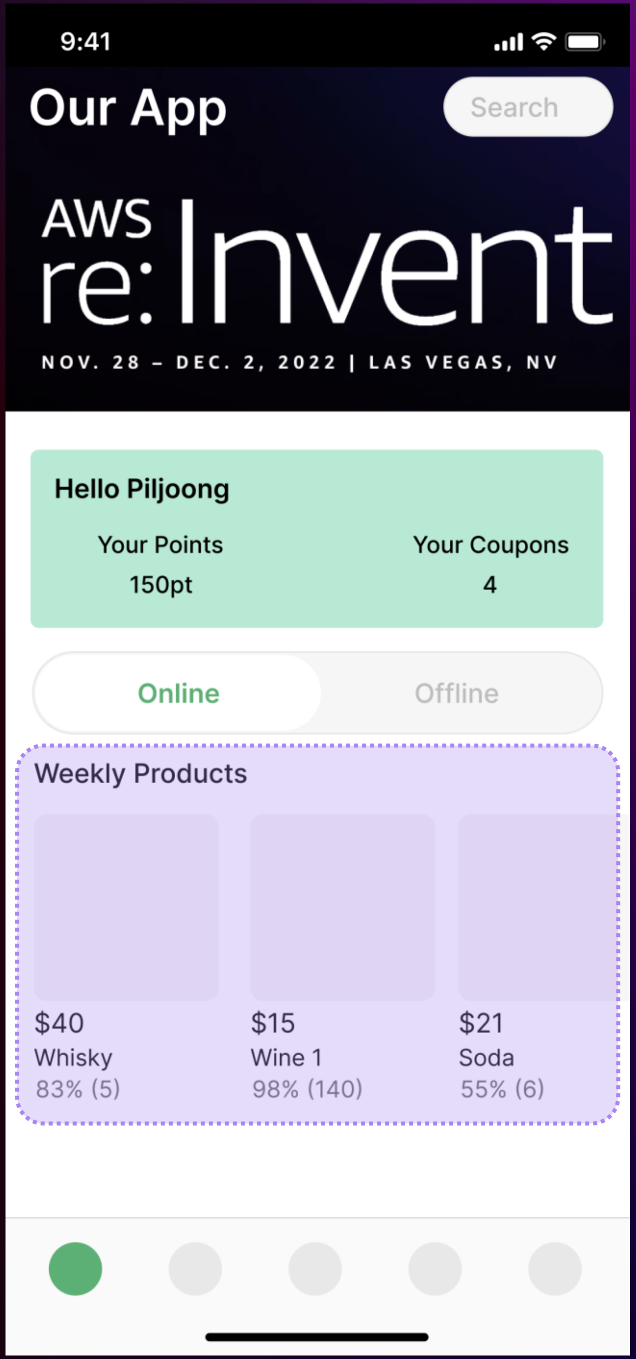




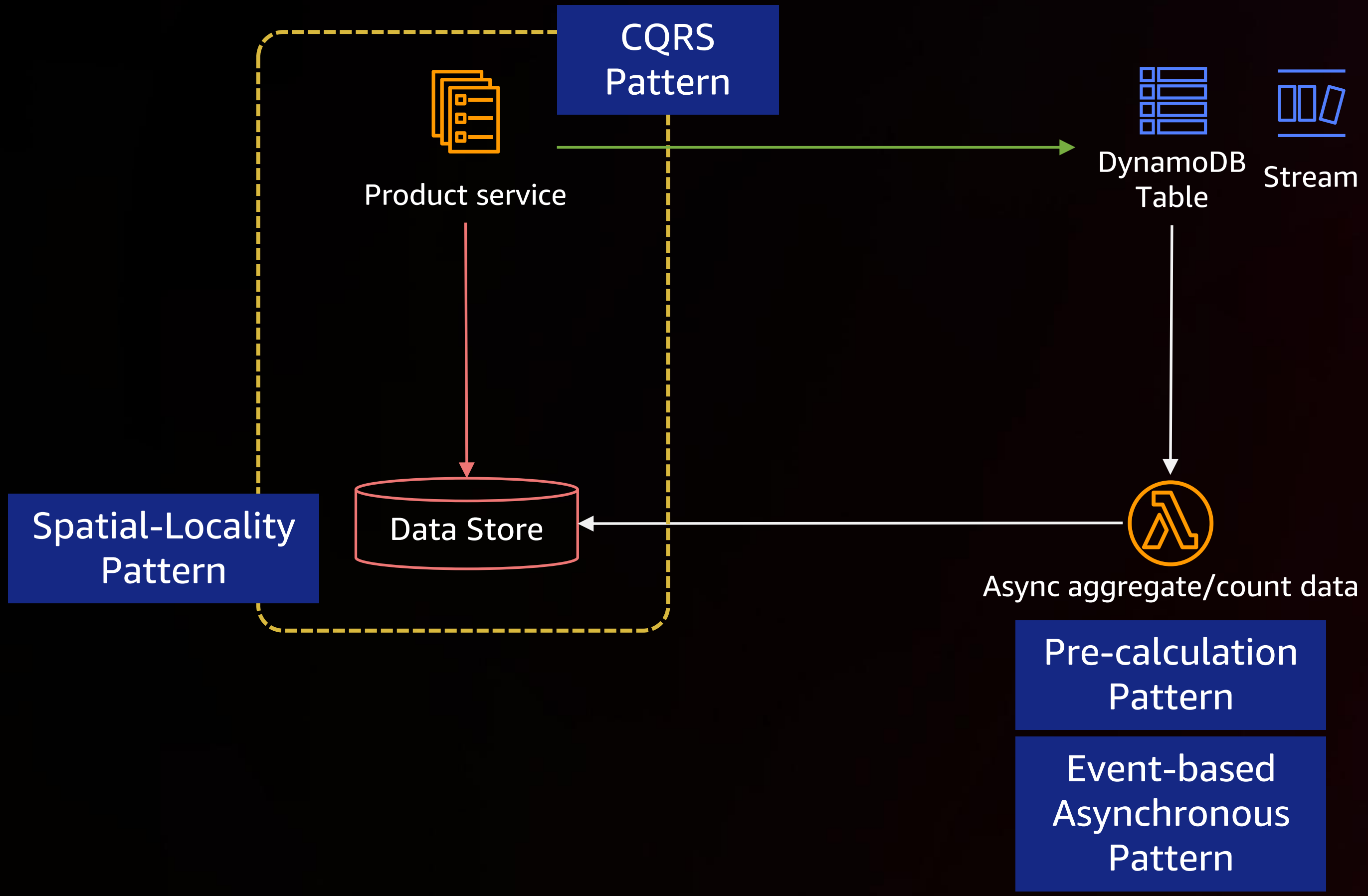
> Product service

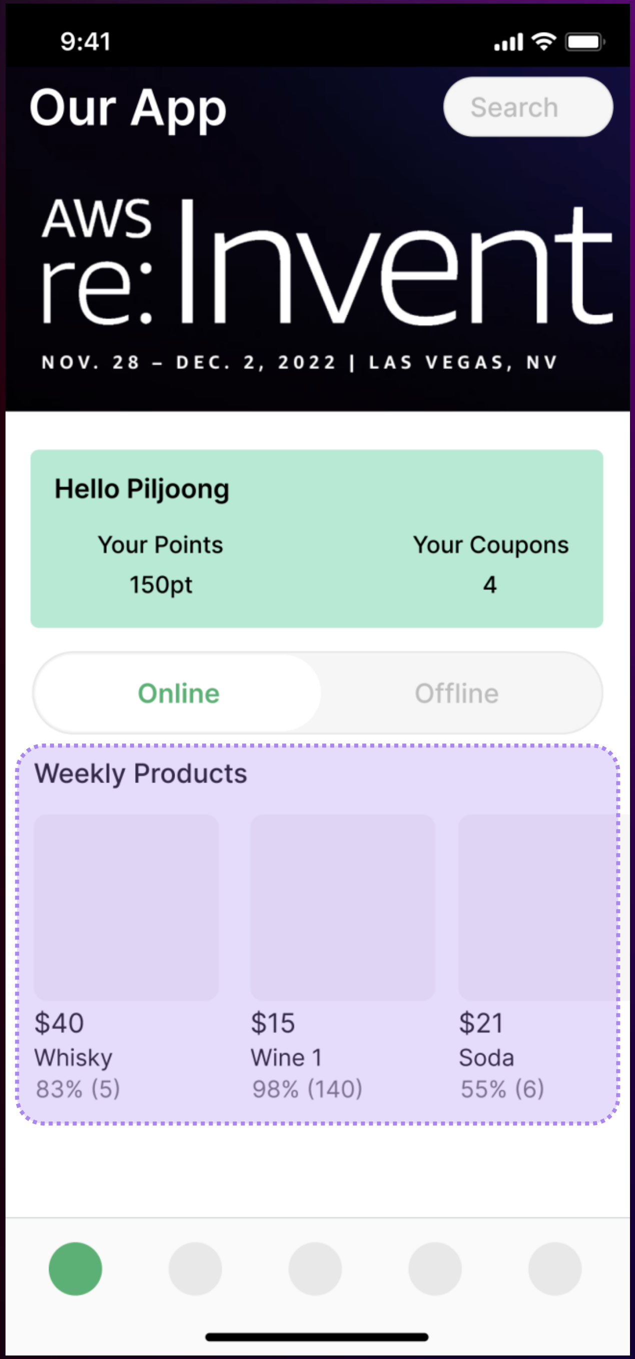
Optimization





Optimization





Hello Piljoong

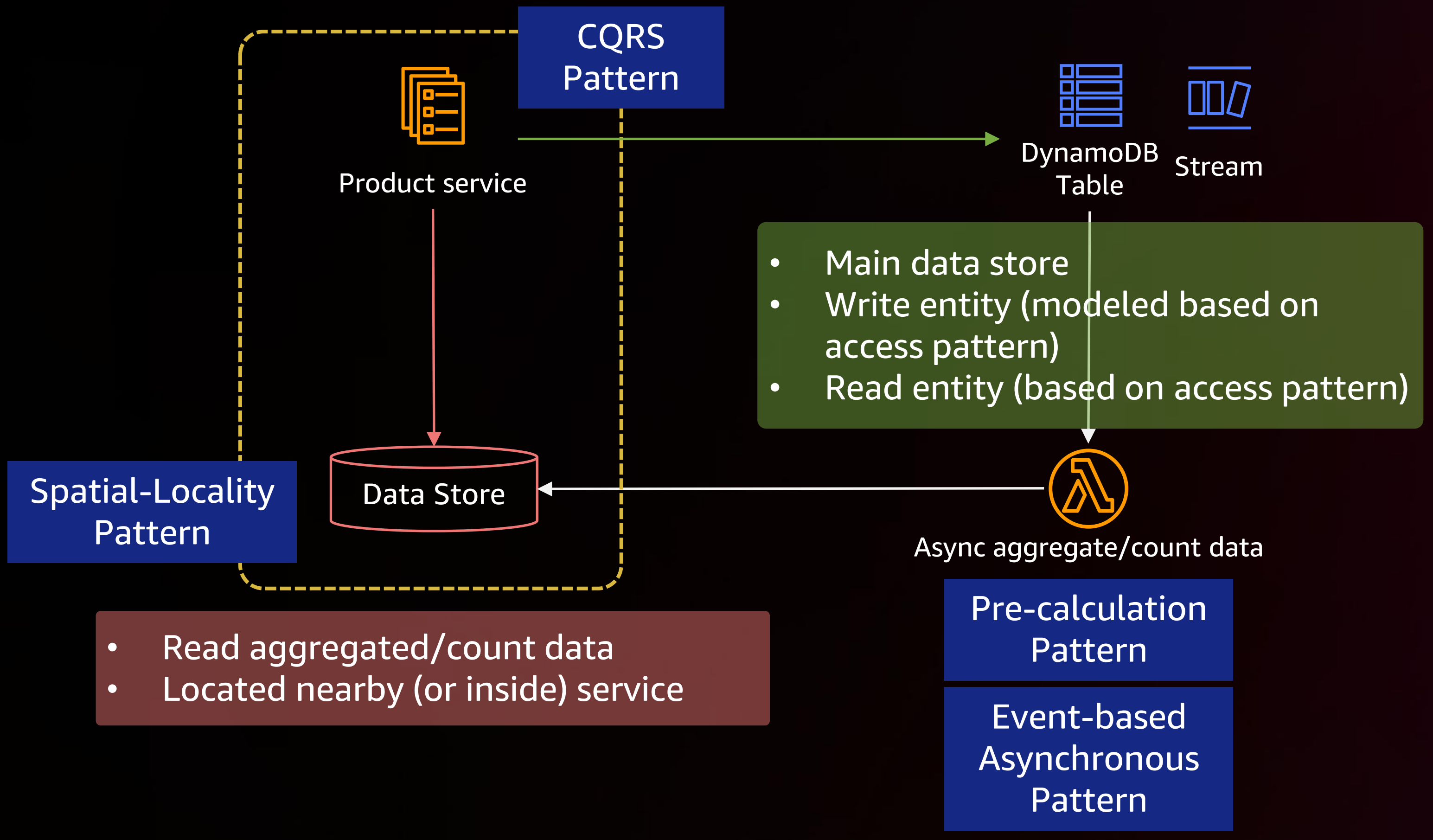
Your Points	Your Coupons
150pt	4

Weekly Products

$40	$15	$21
Whisky	Wine 1	Soda
83% (5)	98% (140)	55% (6)

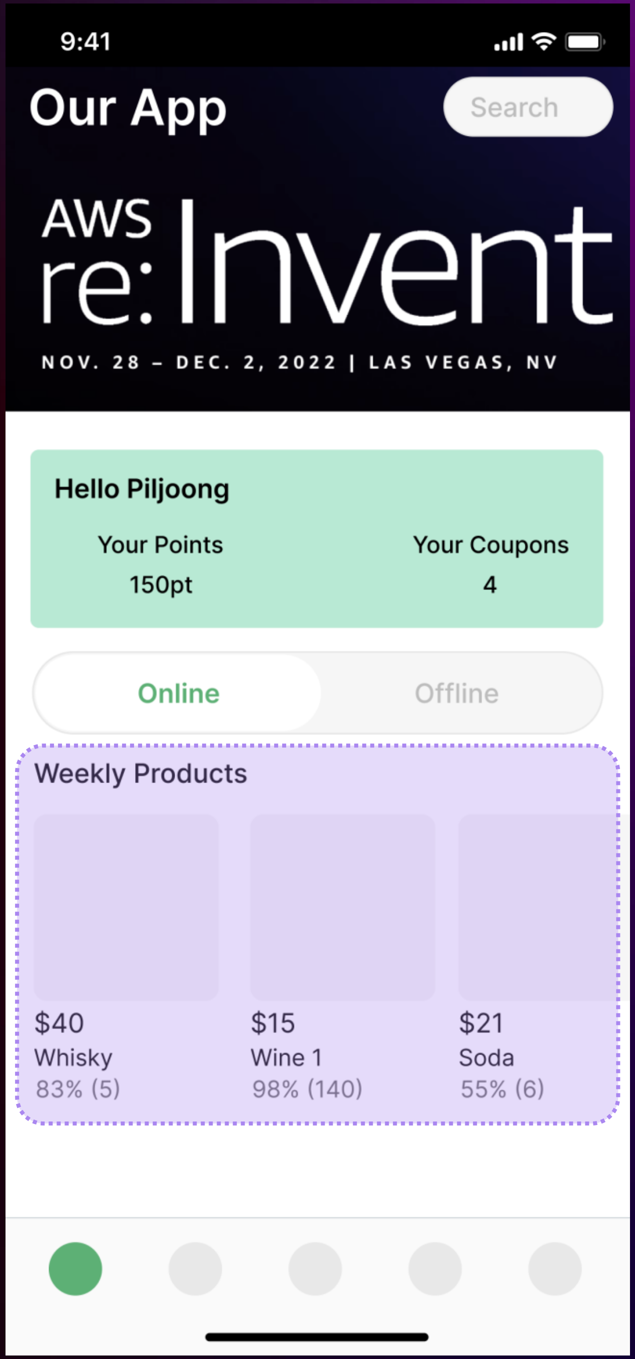
> Product service

Final Architecture



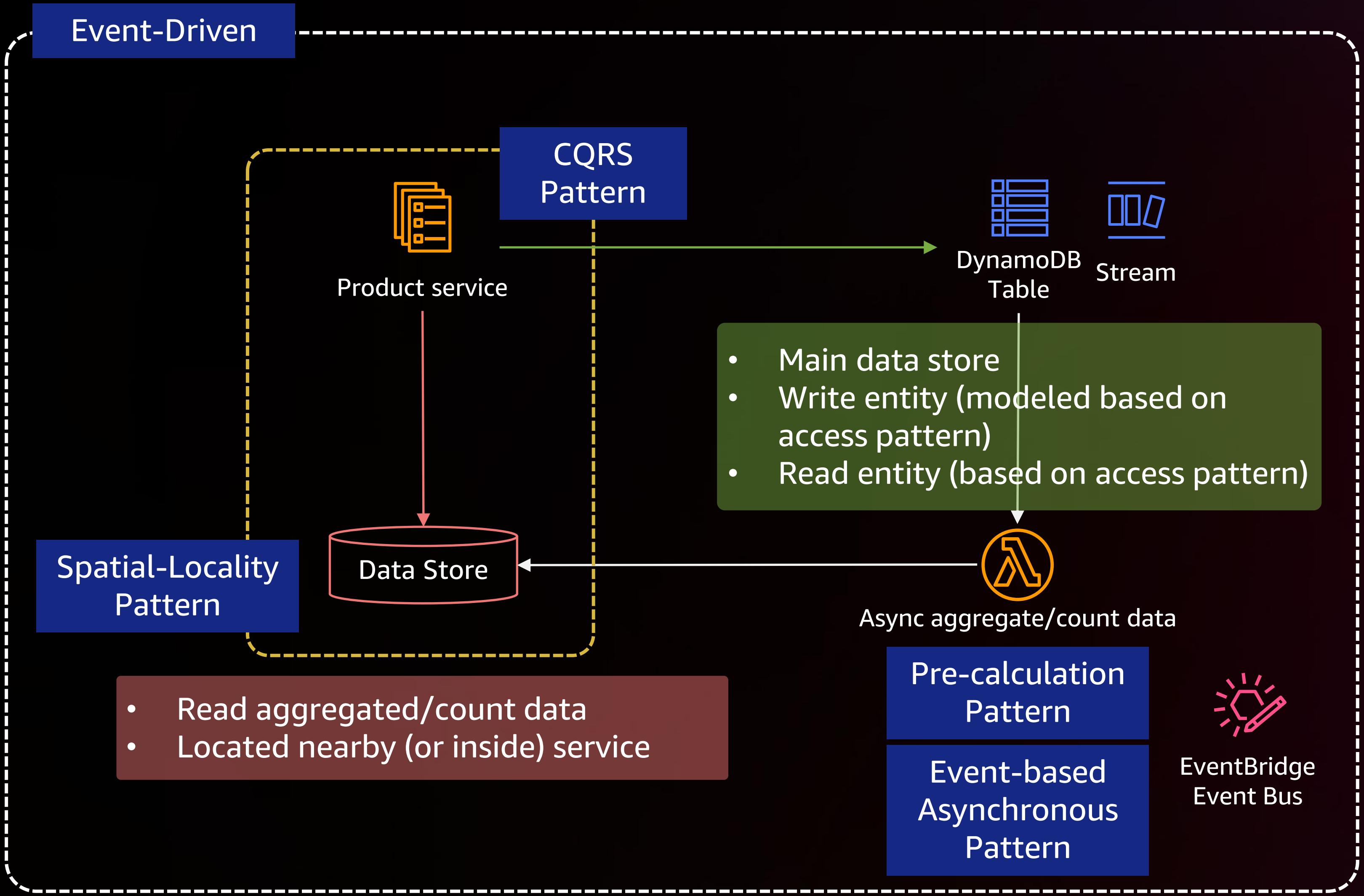
- Main data store
- Write entity (modeled based on access pattern)
- Read entity (based on access pattern)

- Read aggregated/count data
- Located nearby (or inside) service



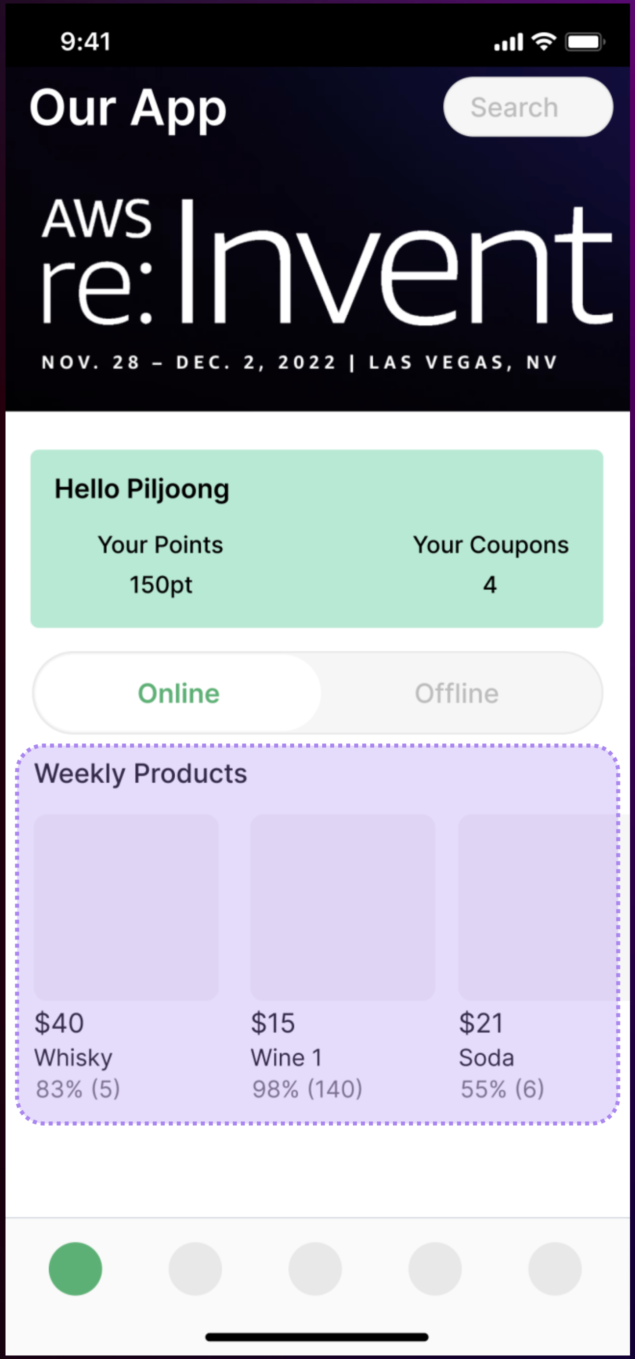
> Product service

Final Architecture



- Main data store
- Write entity (modeled based on access pattern)
- Read entity (based on access pattern)

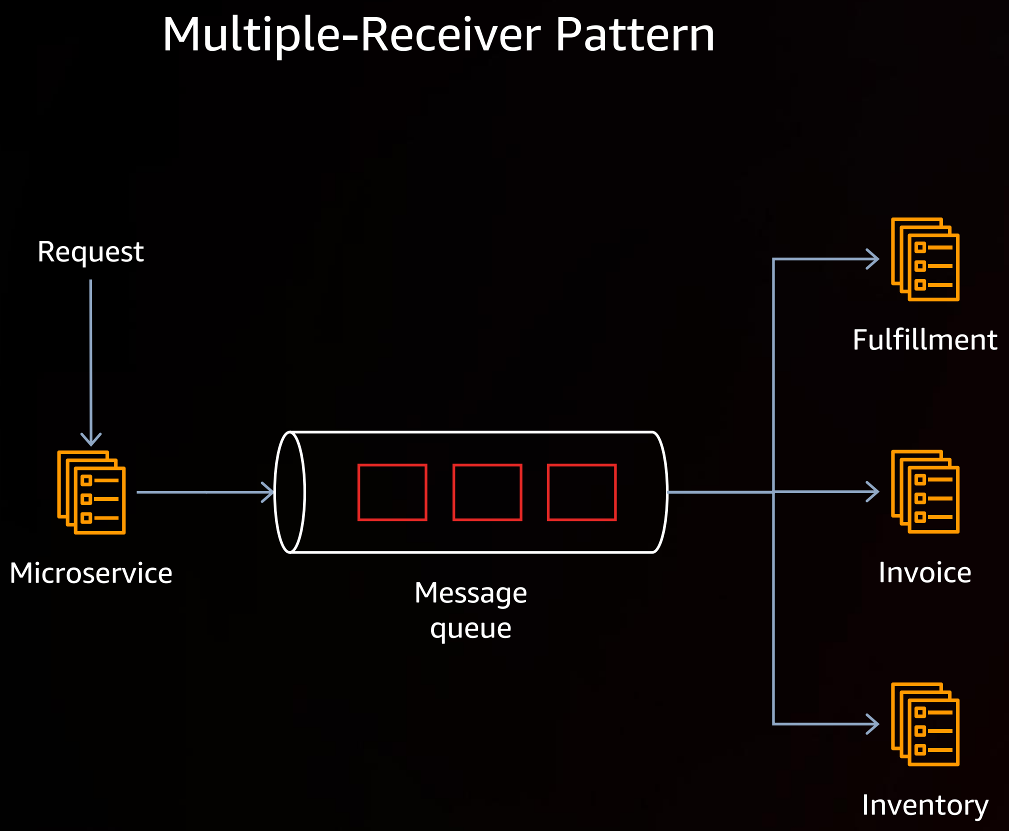
- Read aggregated/count data
- Located nearby (or inside) service

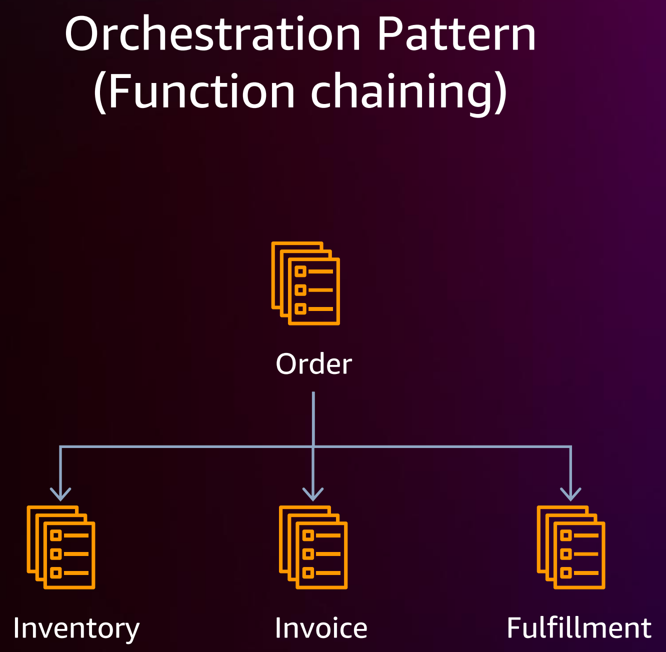


Optimization
- Rule of thumbs

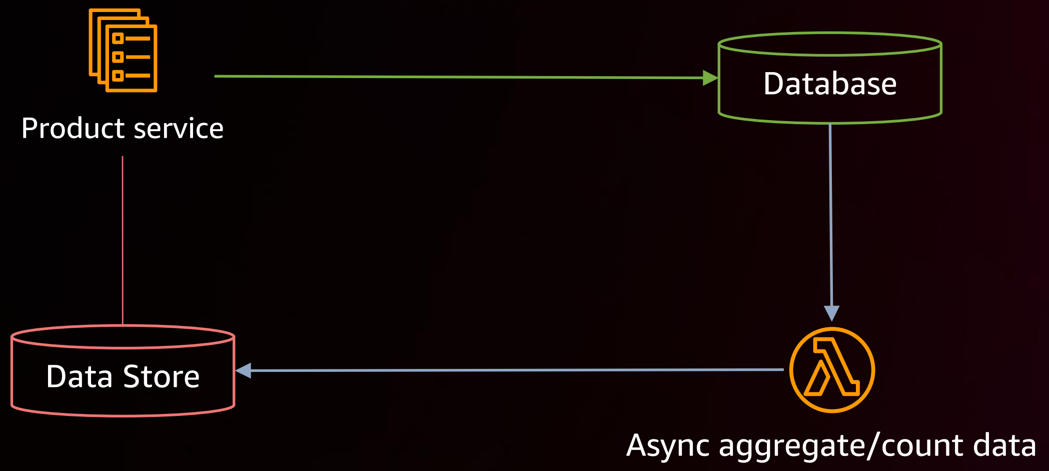
- Separate read and write stores
- Store pre-calculated data nearby it will be accessed to
- Redundant data when necessary to minimize resource consumption

Service Integration: Extensibility



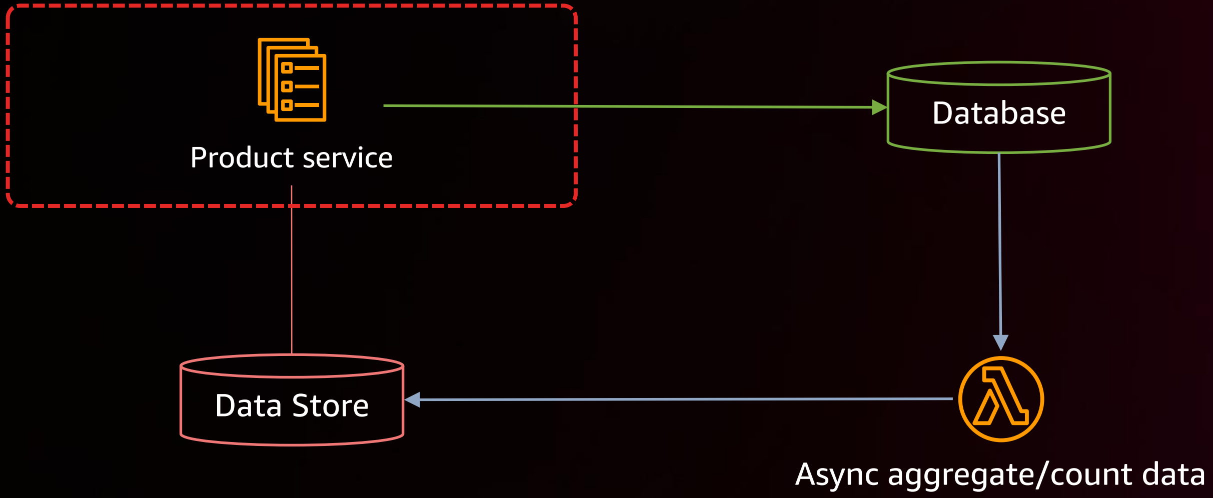


Spot

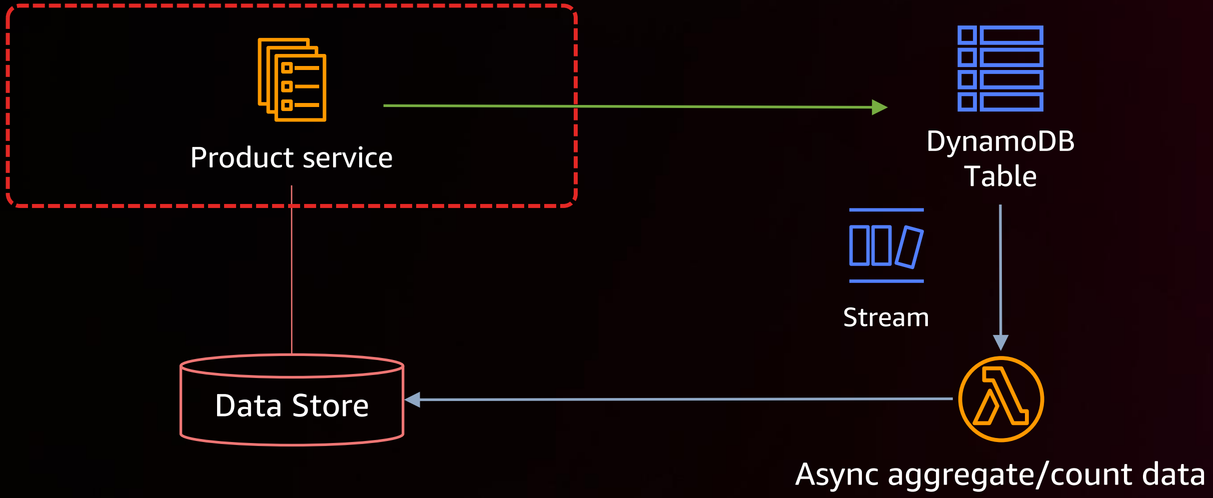

Product service
Database
Data Store
Async aggregate/count data

> Extensibility

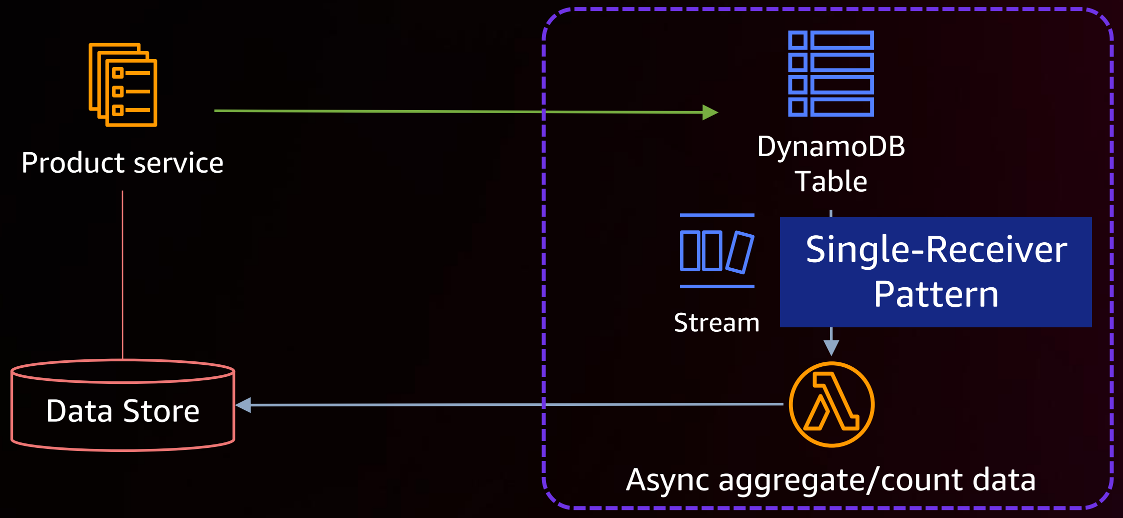
Spot



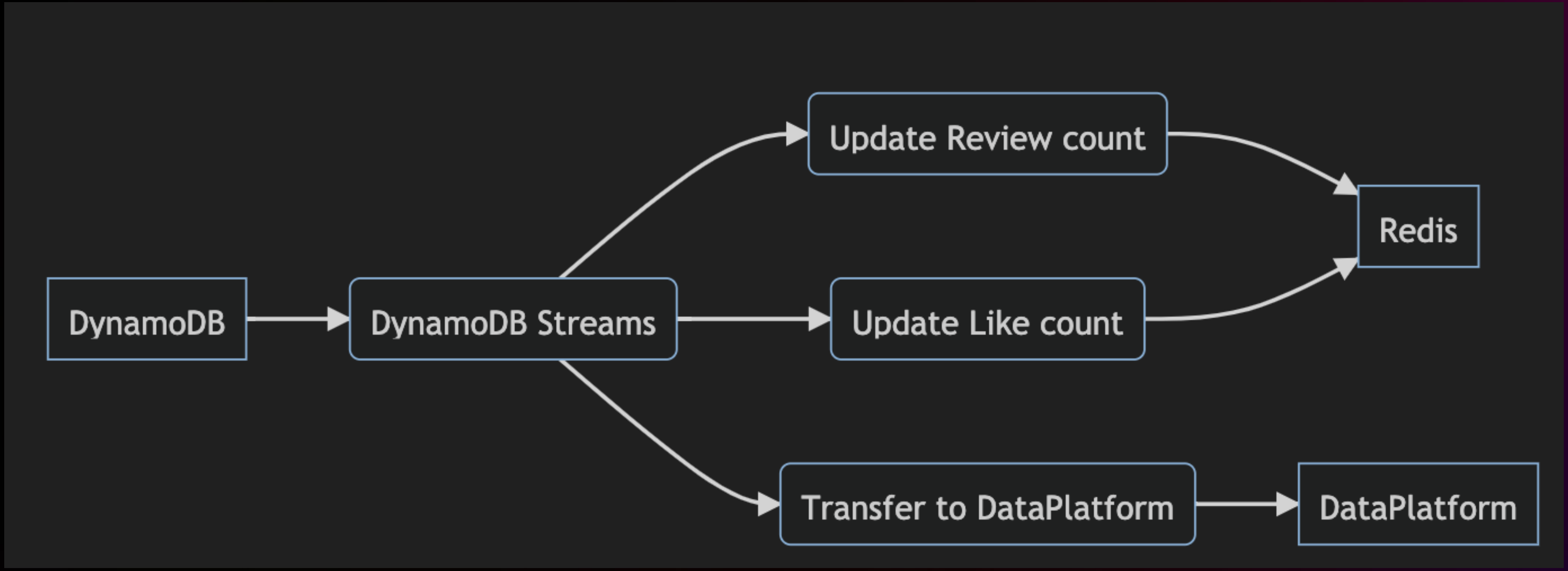
Spot


Product service
DynamoDB Table
Stream
Data Store
Async aggregate/count data

Spot

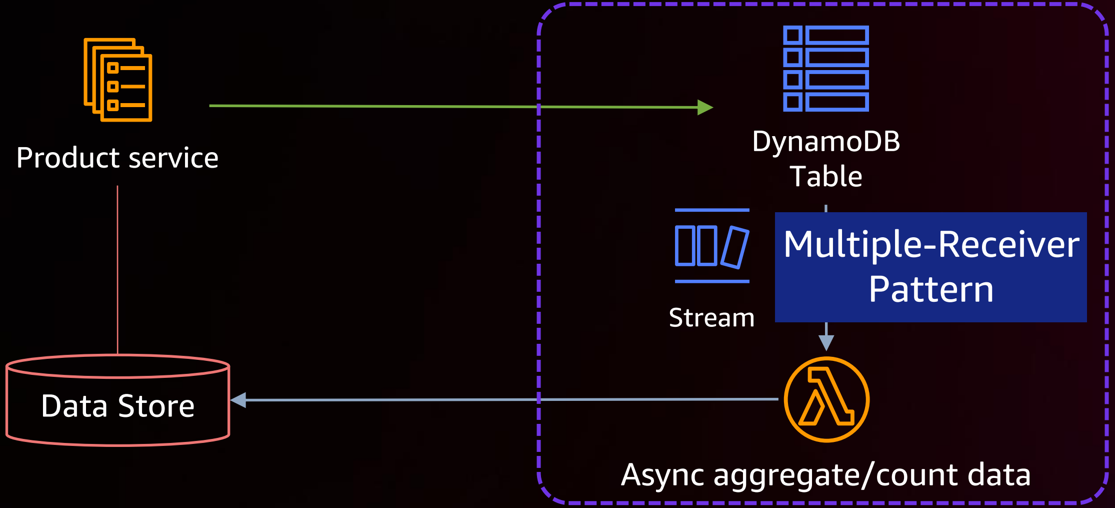

Product service
DynamoDB Table
Stream
Single-Receiver Pattern
Data Store
Async aggregate/count data

Multiple Tasks



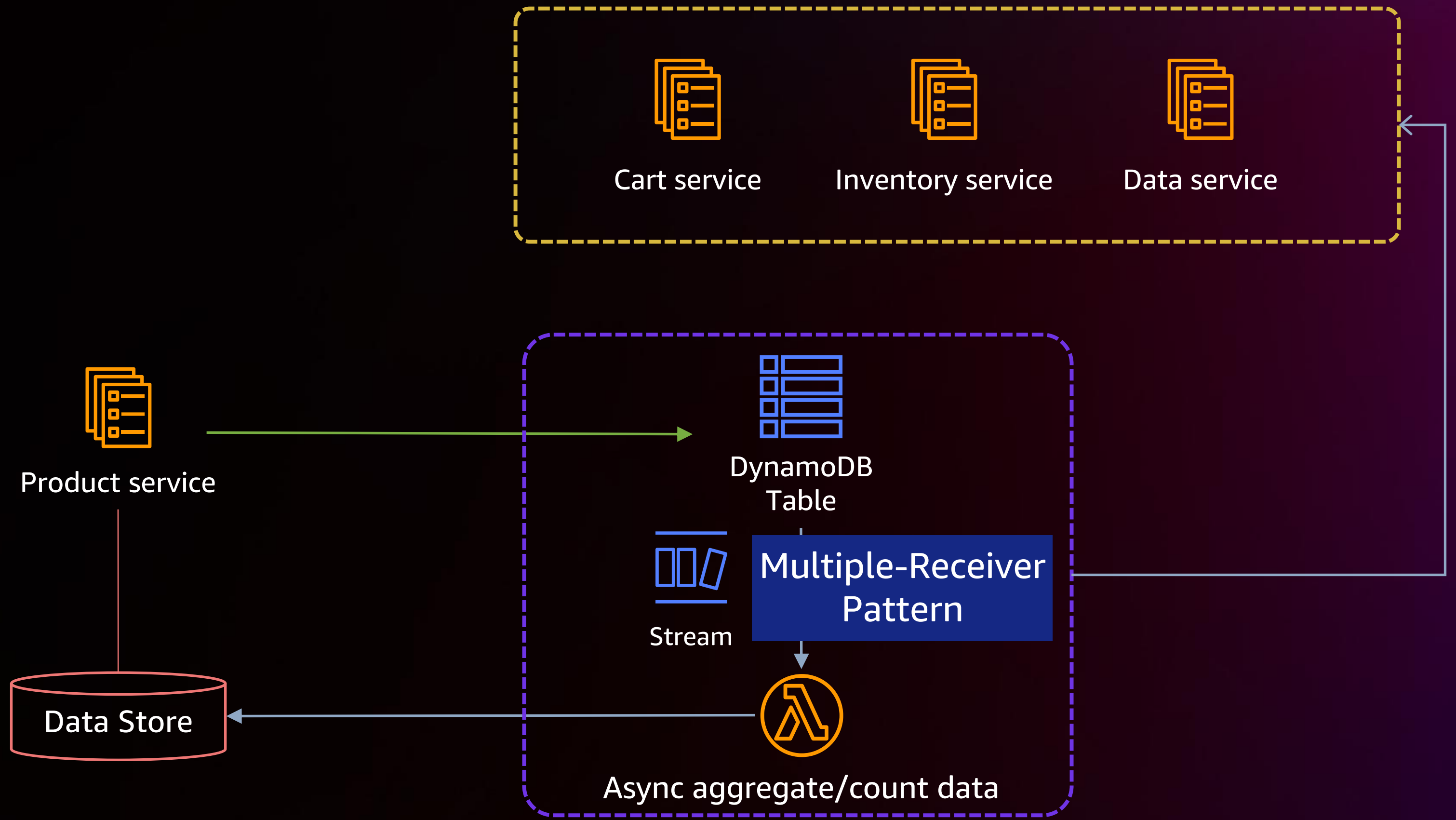
> Extensibility

Multiple-Receiver

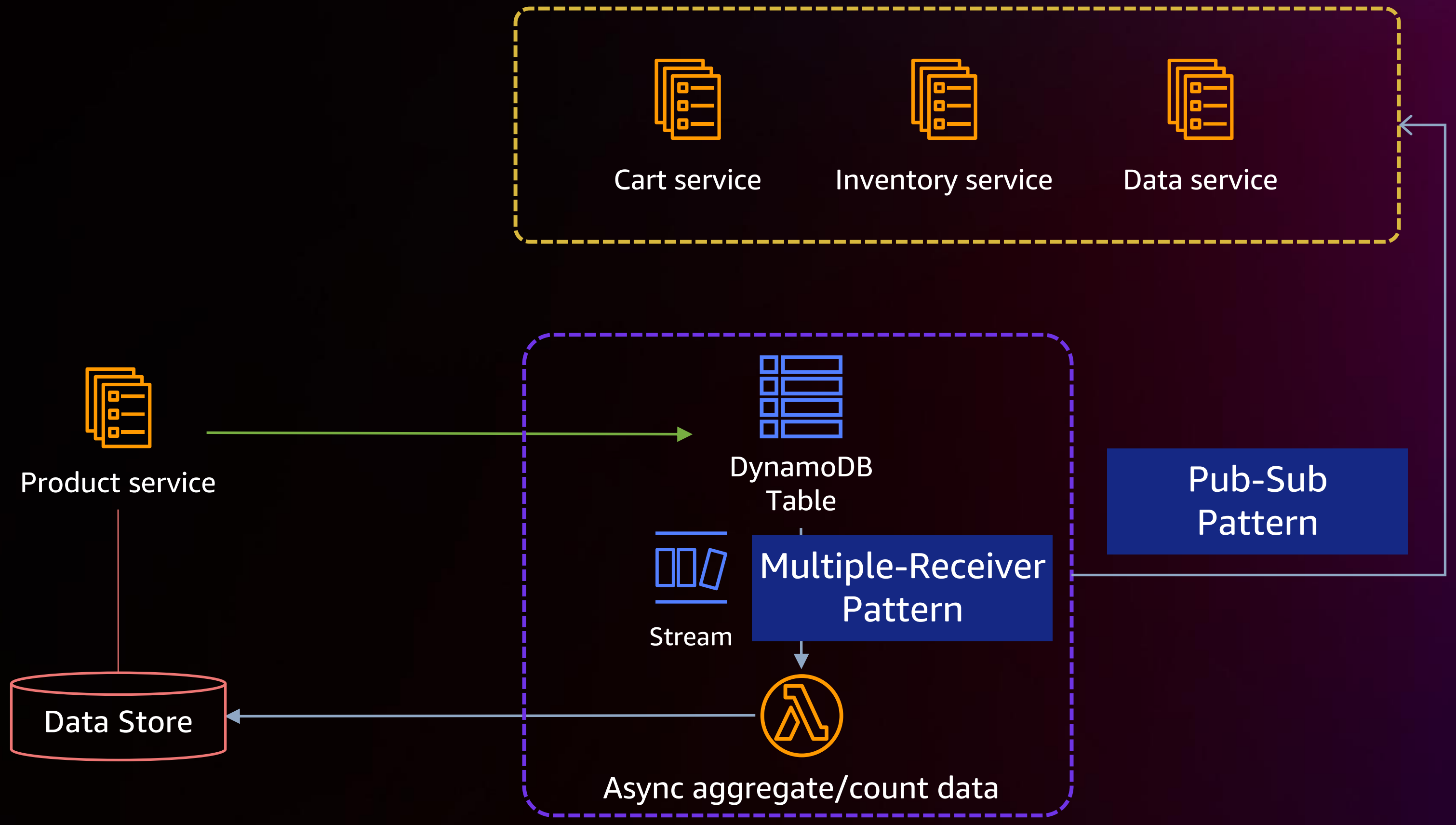


> Integration

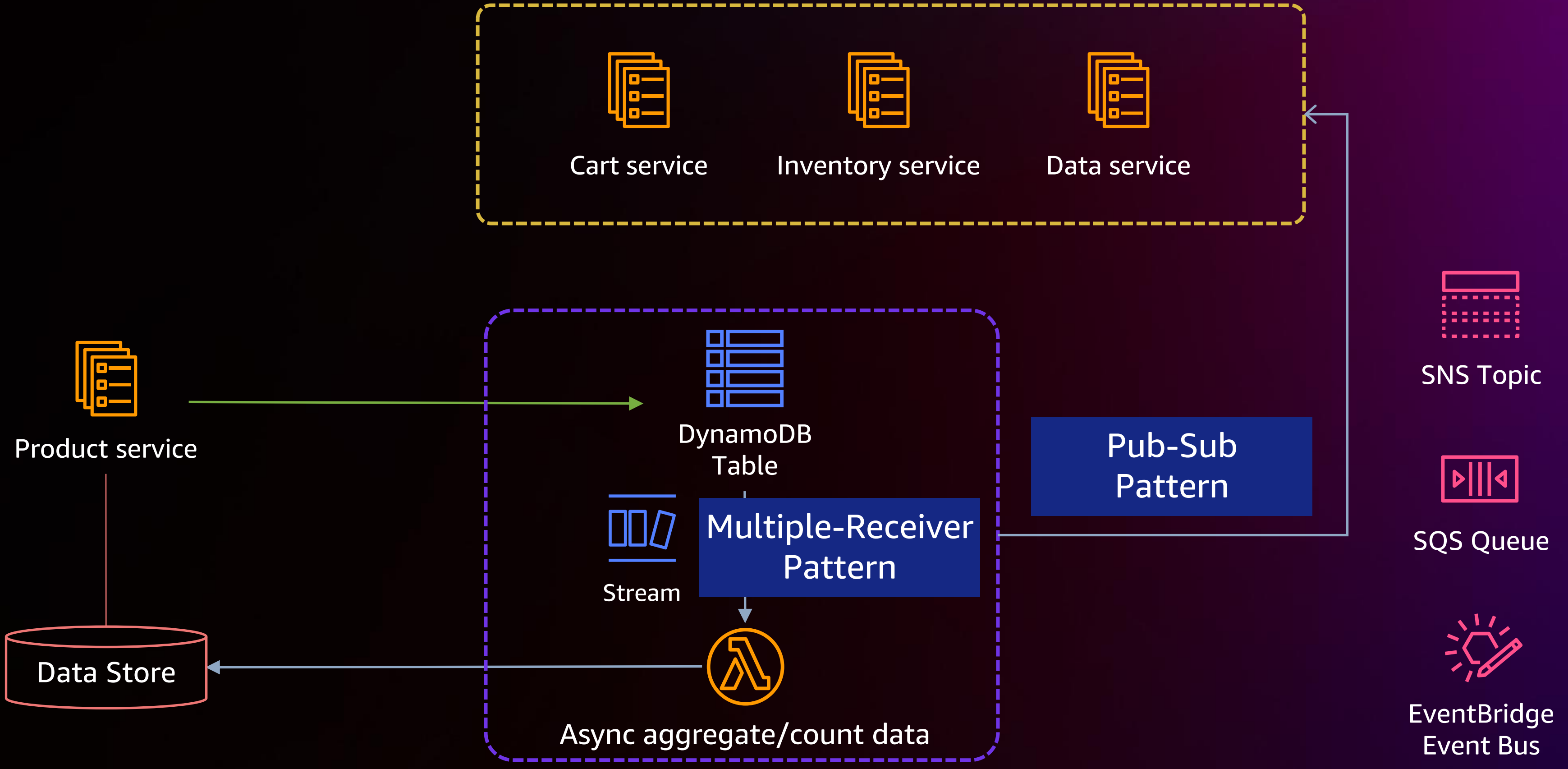
Multiple-Receiver



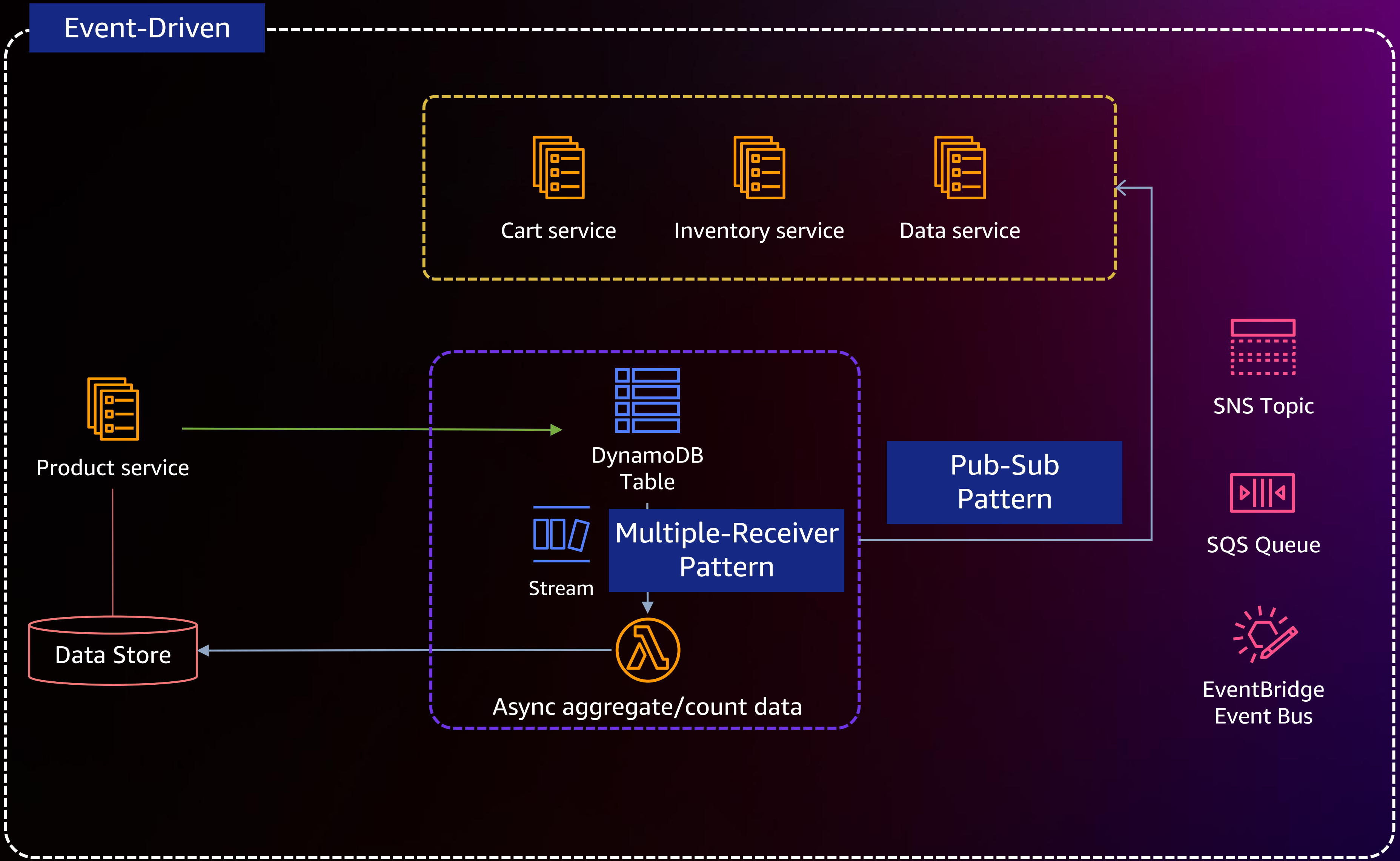
Pub-Sub


Cart service
Inventory service
Data service
Product service
DynamoDB Table
Stream
Multiple-Receiver Pattern
Pub-Sub Pattern
Data Store
Async aggregate/count data

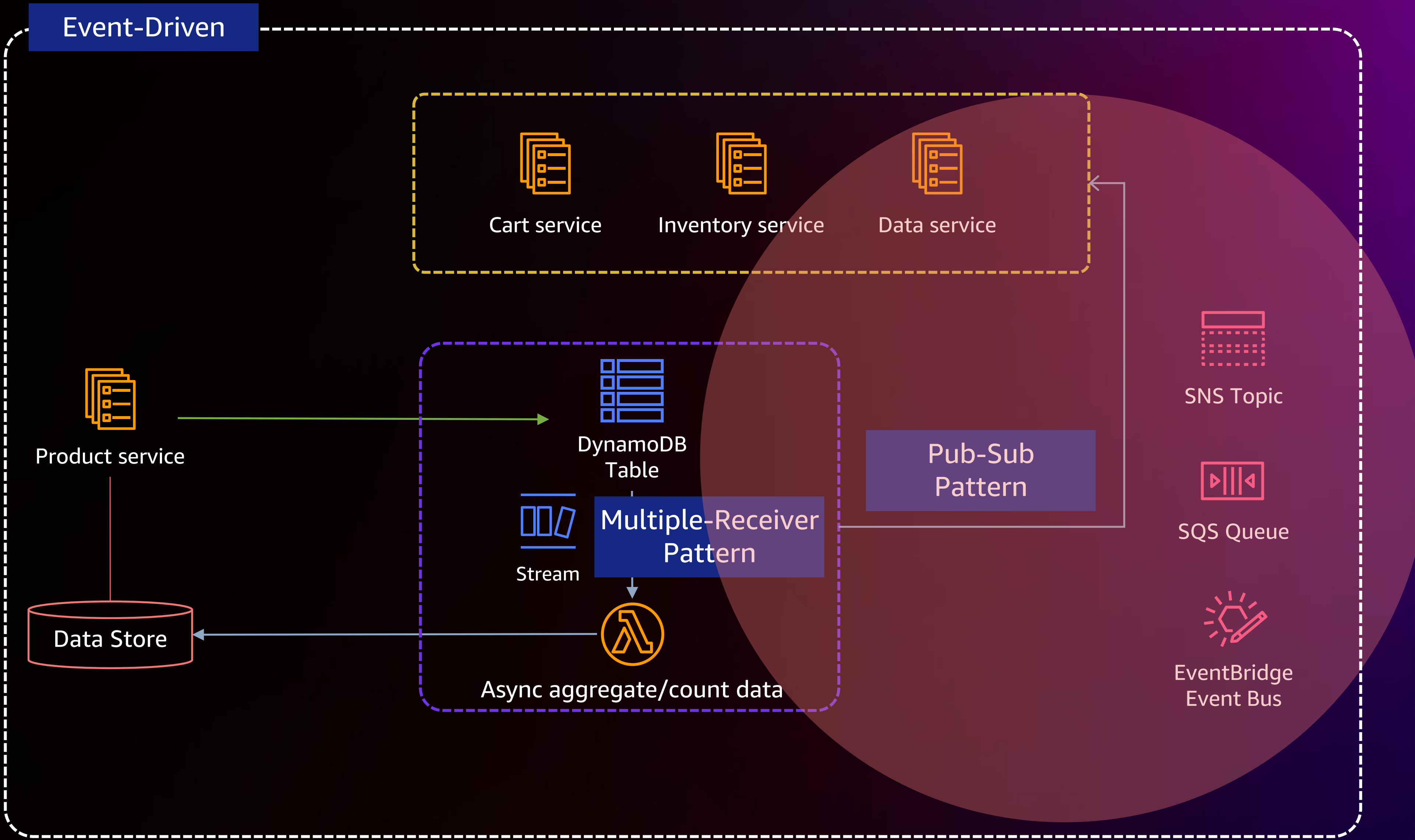
AWS Services



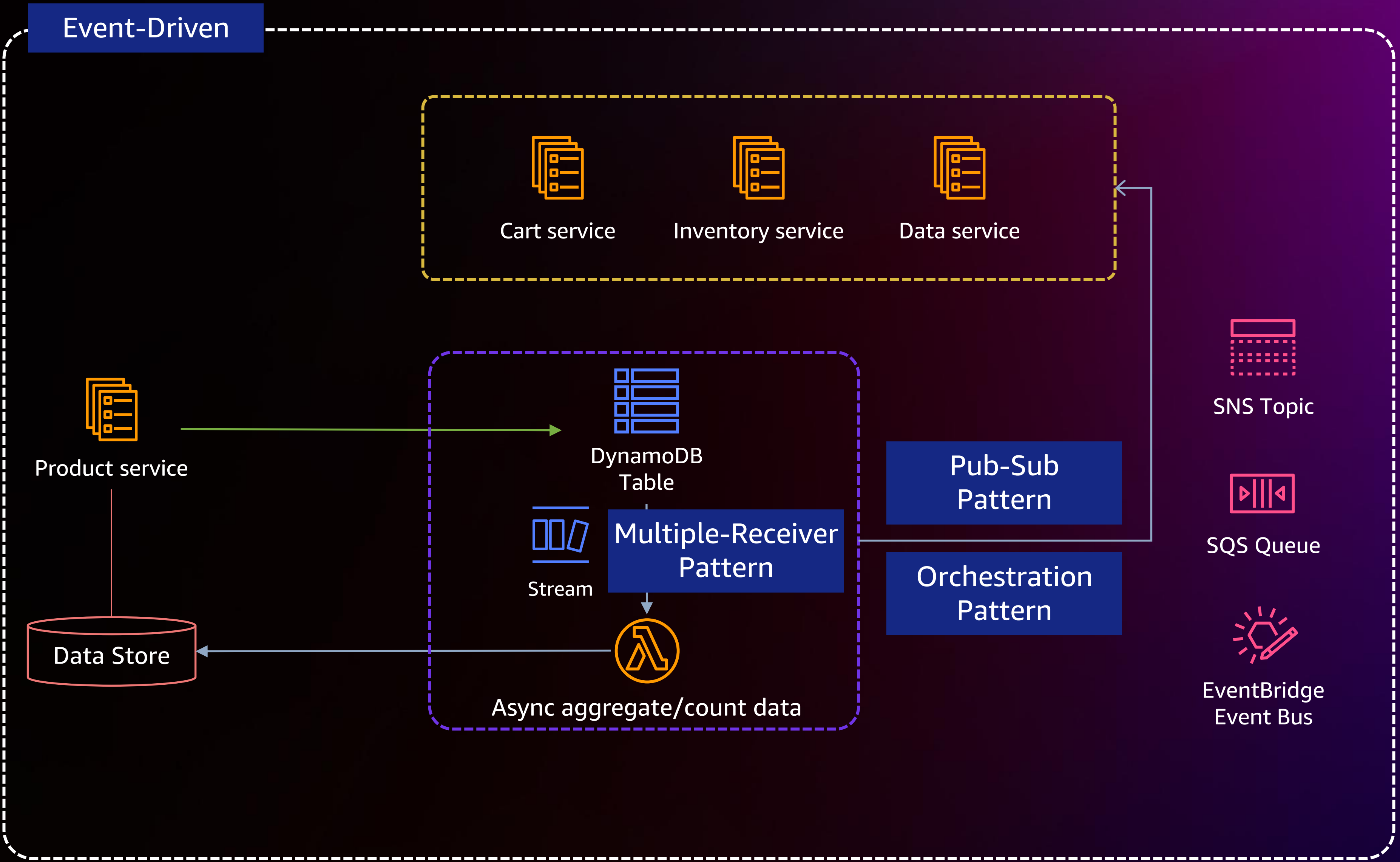
Event-Driven



Growing Complexity



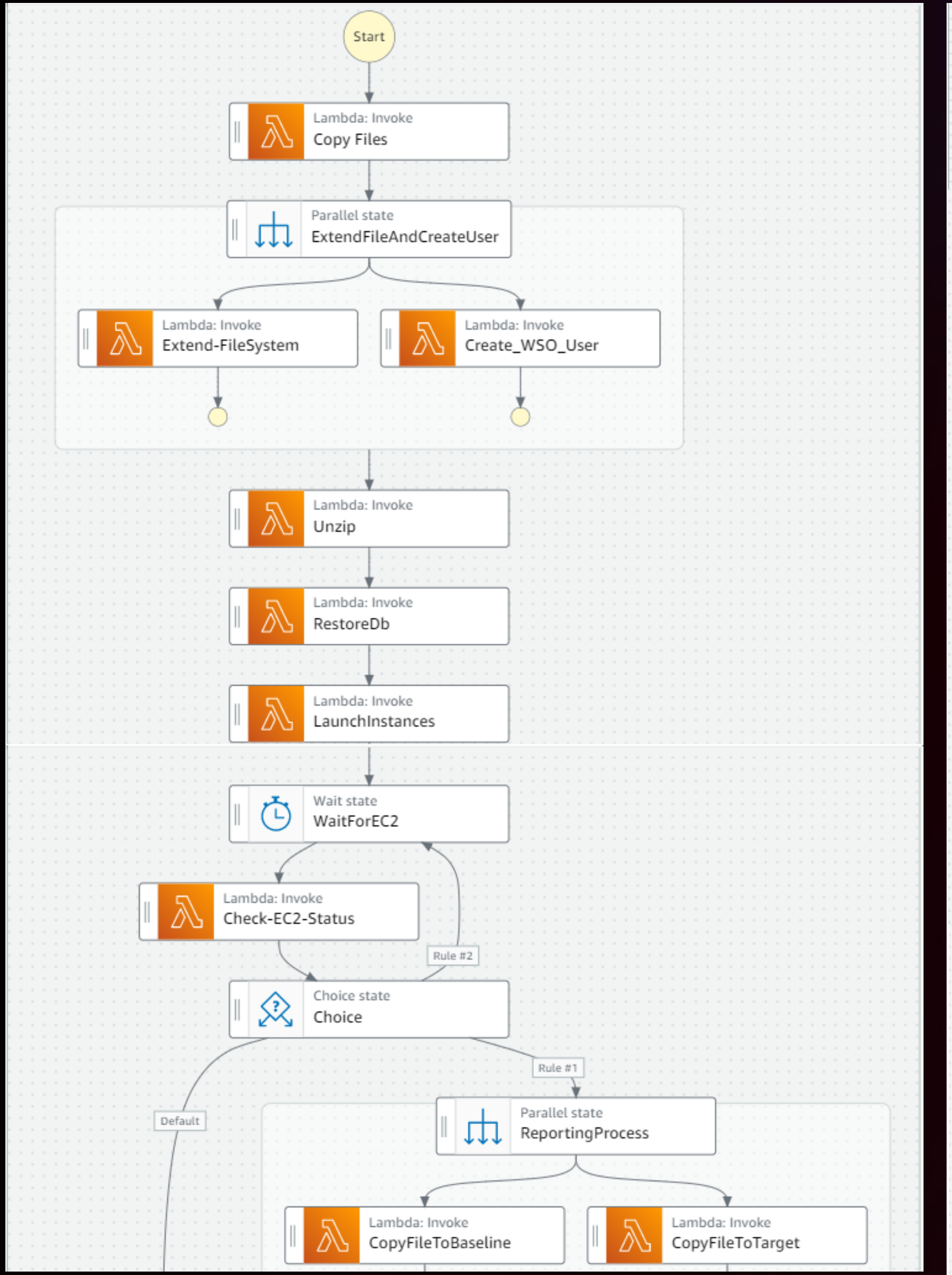
Orchestration Pattern

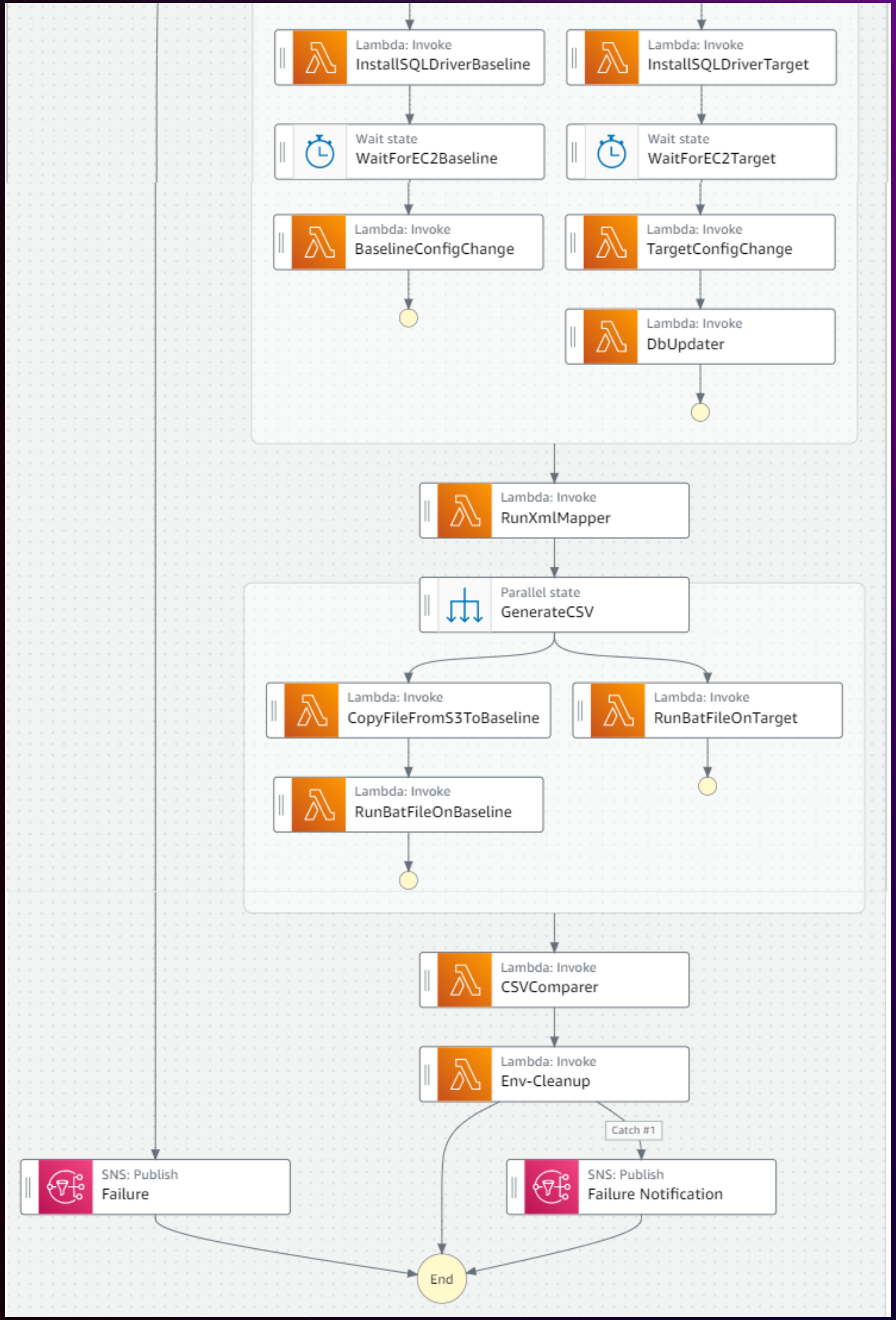


> Integration

Workflow Orchestration

AWS Step Functions





Integration
- Rule of thumbs

- Make as much as possible asynchronous
- Adopt eventual consistency as much as possible
- Decouple workflow/business logic from the code and core system

Key Takeaway

Understand the requirements and characteristics of applications, services, and functionalities

Microservices architecture is still the first thing to consider

Patterns are not a silver bullet, but can bring immediate value

Find spots where simplified patterns can be applied to

Thank you!

Hyungil Kim
trentkim@amazon.com

Piljoong Kim
piljoong@amazon.com

Please complete the session survey in the **mobile app**