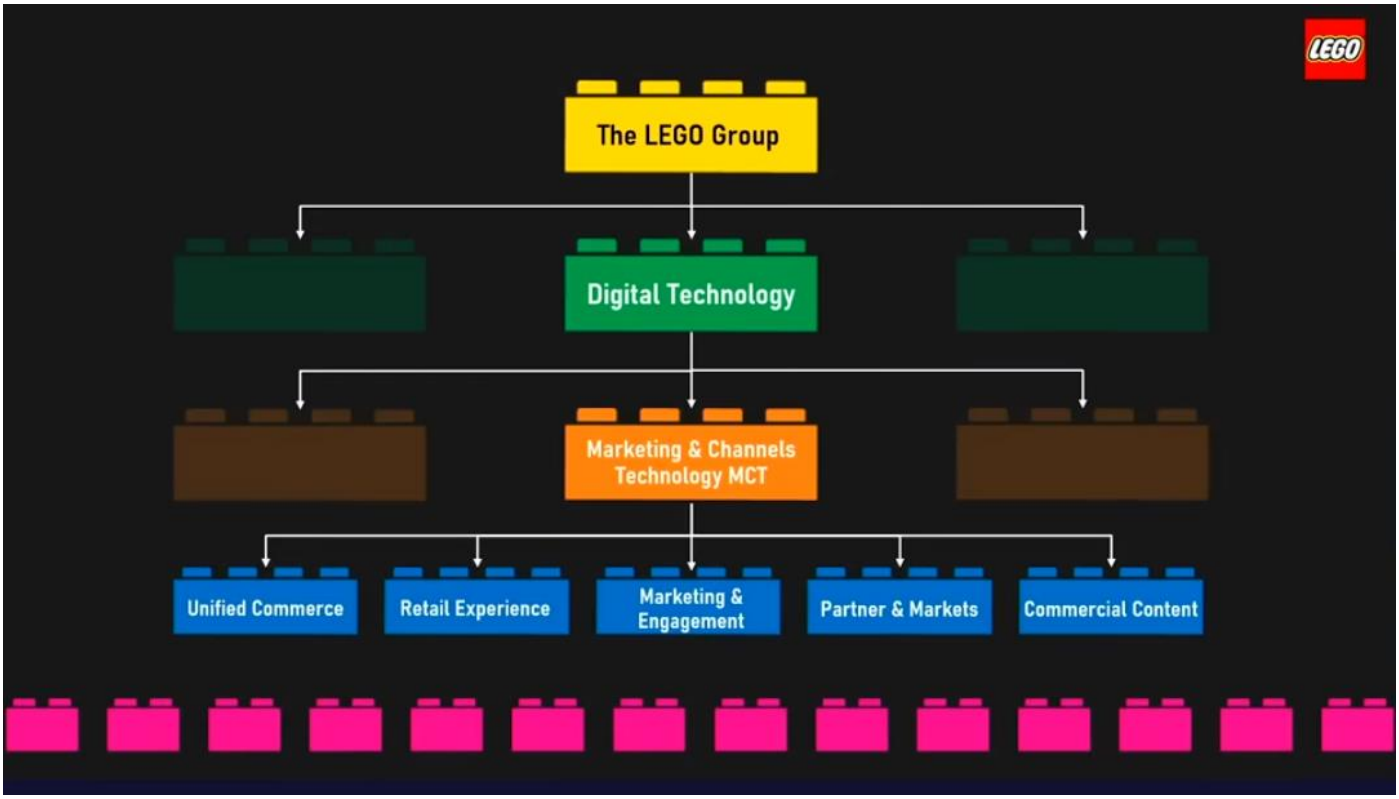
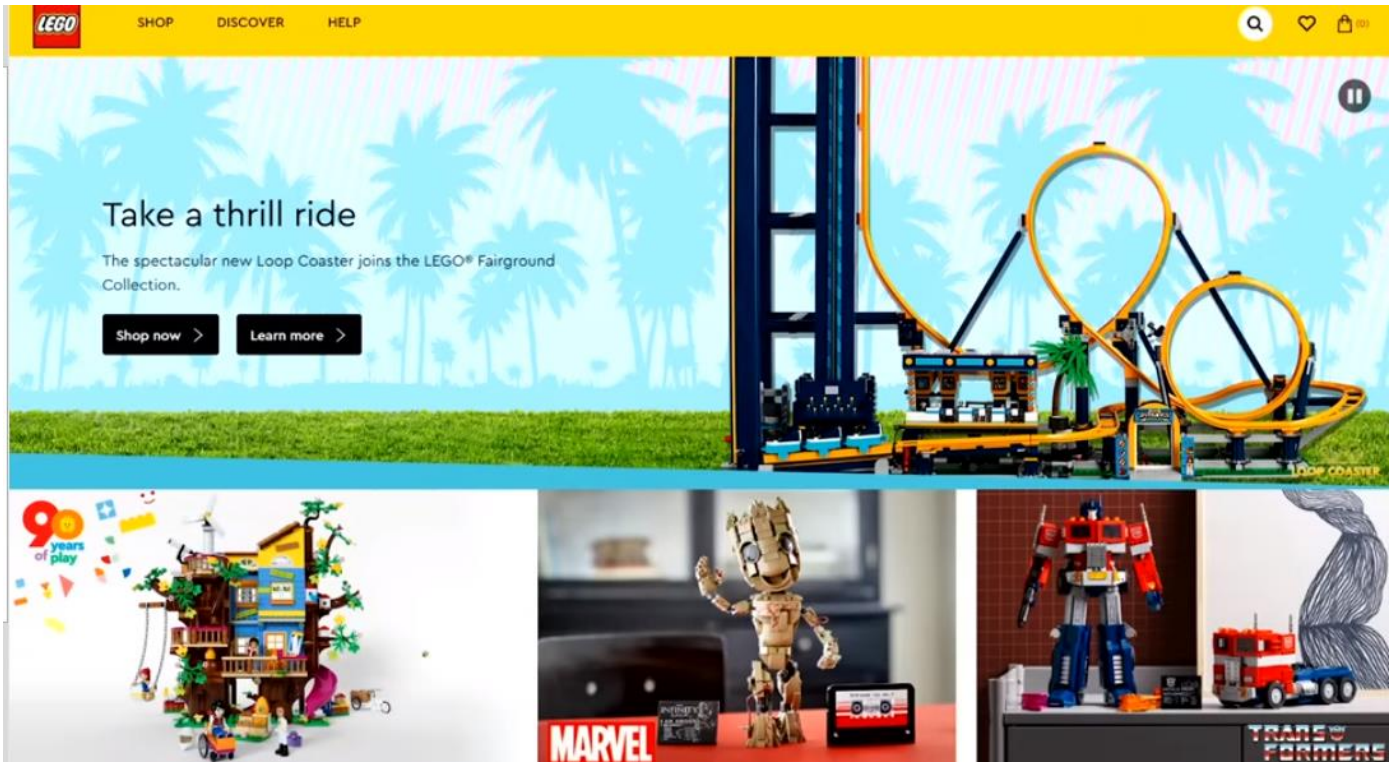
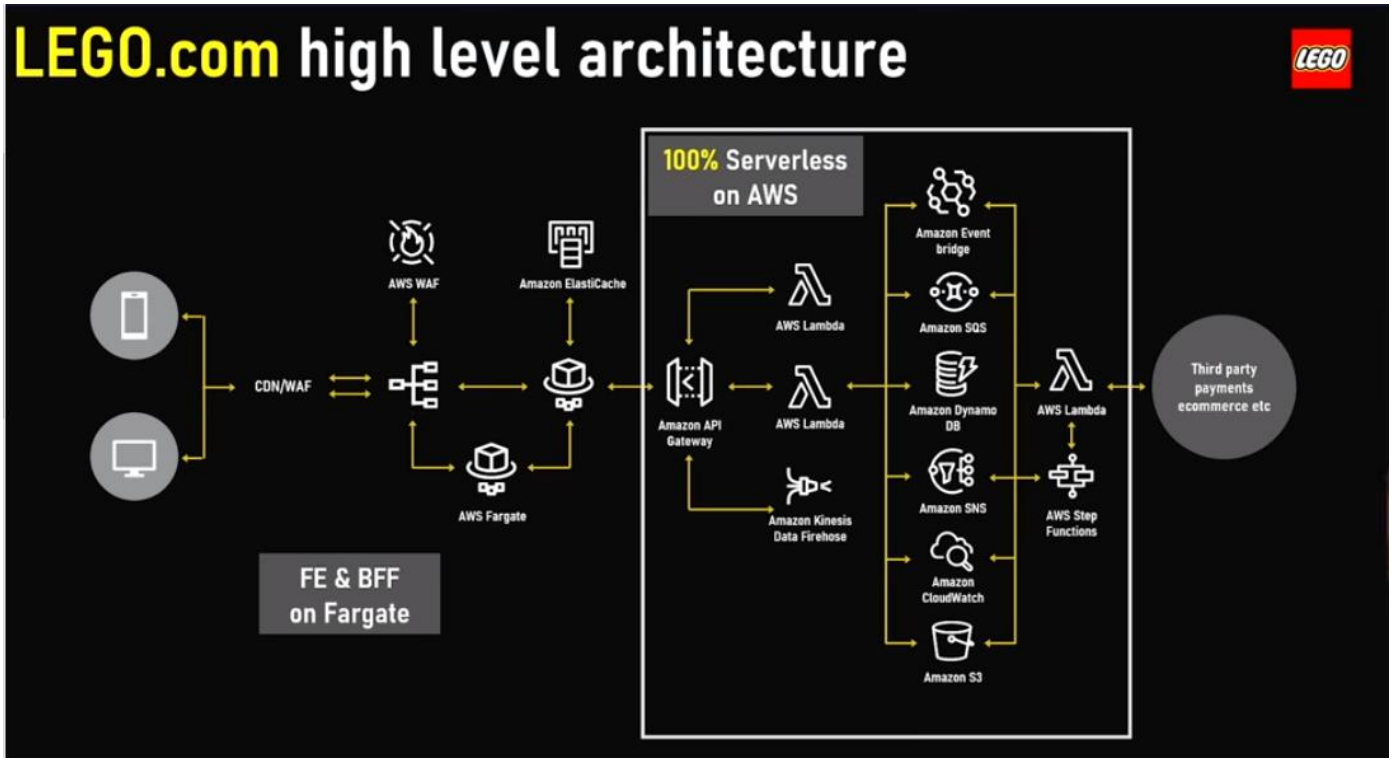
Reacting to events goes beyond the programming world. It's the norm for many things in life. In software, as we move away from legacy monolith applications to building modern distributed microservices, the need for event-driven computing has become ever more important. There are many facets to why event-driven architecture is at the center of a serverless application we build and operate in the cloud. Could the reason be the cost, performance, team velocity, sustainability, or a combination of many? In serverless, why do we even think of event-driven computing? If event-driven architecture is the way forward, then how do we make a start and become successful?

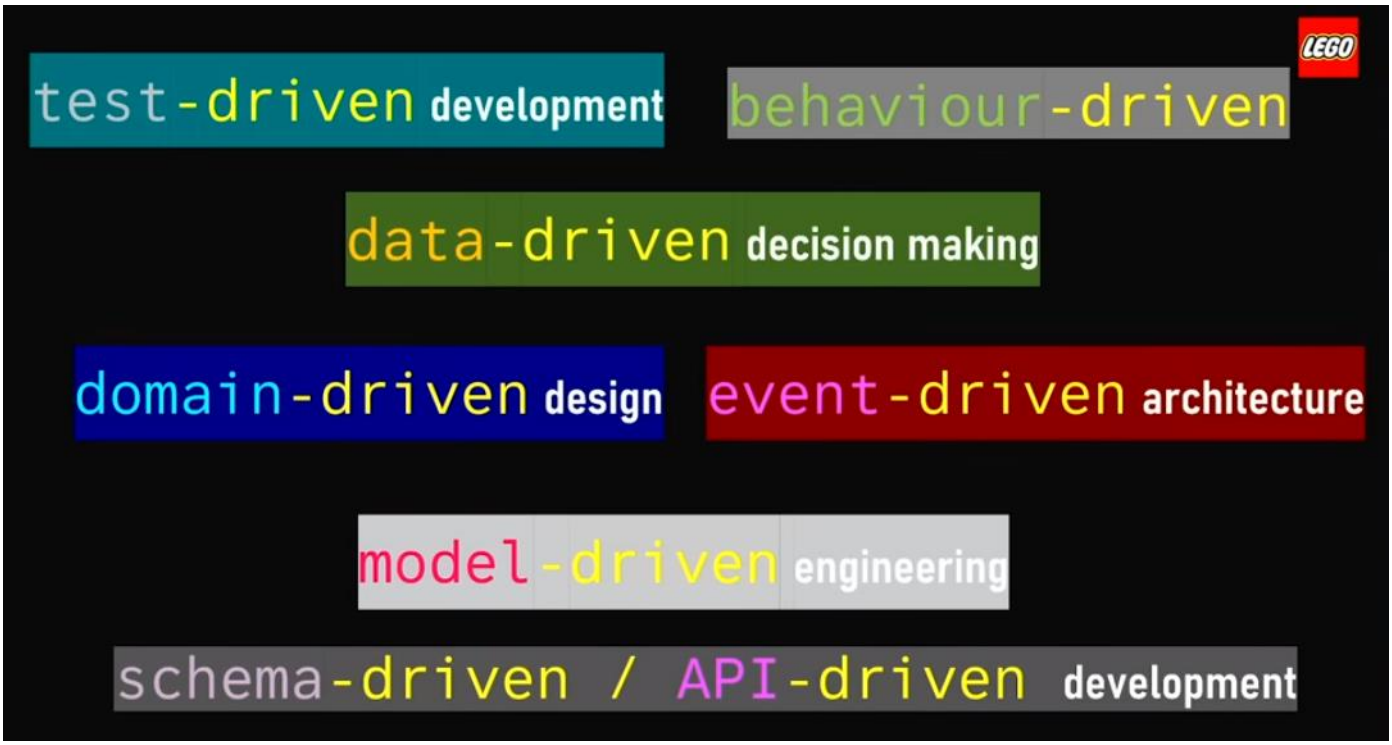



SHOP
DISCOVER
HELP
Take a thrill ride
The spectacular new Loop Coaster joins the LEGO® Fairground Collection.
Shop now
Learn more
90 years of play
MARVEL
TRANSFORMERS


LEGO.com high level architecture
100% Serverless on AWS
AWS WAF
Amazon ElastiCache
CDN/WAF
AWS Fargate
FE & BFF on Fargate
AWS Lambda
Amazon API Gateway
AWS Lambda
Amazon Kinesis Data Firehose
Amazon Event bridge
Amazon SQS
Amazon Dynamo DB
Amazon SNS
Amazon CloudWatch
Amazon S3
AWS Lambda
AWS Step Functions
Third party payments ecommerce etc

In our gravity-driven world,

everything is driven by something!



data-driven decision making

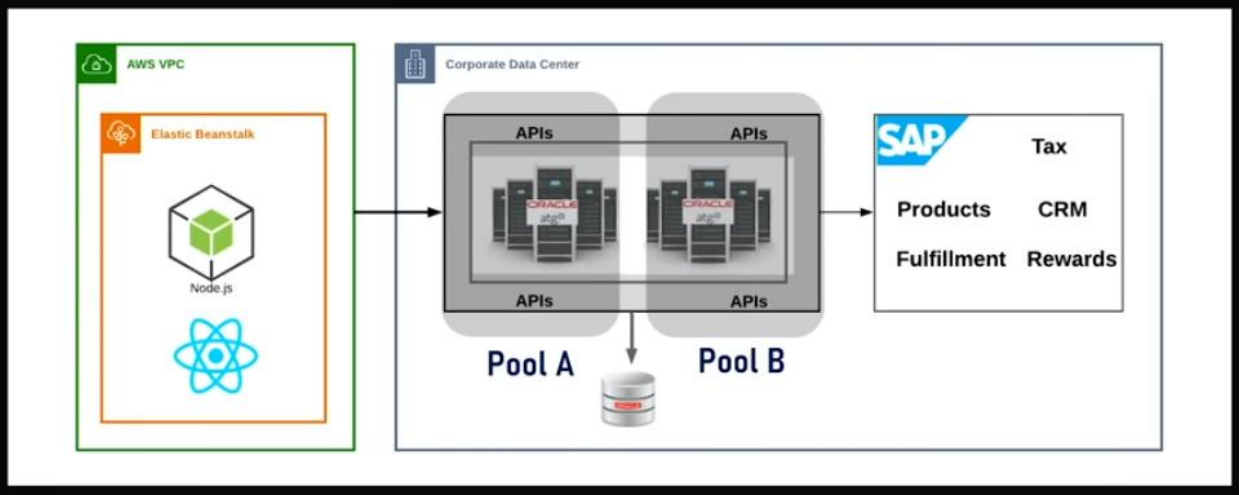
do failure-driven cture

The Runway To EDA

- Monolith & Painful Failures
- Future Wishlist & Bold Start
- EDA – Expansion & Acceleration
- Microservices & Set-Pieces

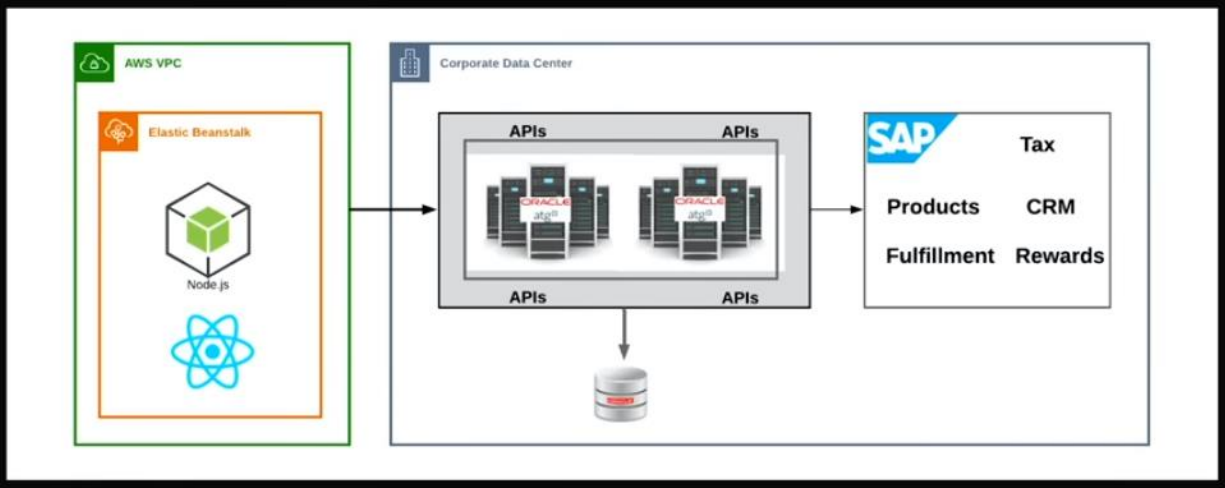
Monolith Days 2016

The then shop.LEGO.com



Drain → Shutdown → Deploy → Test → Release

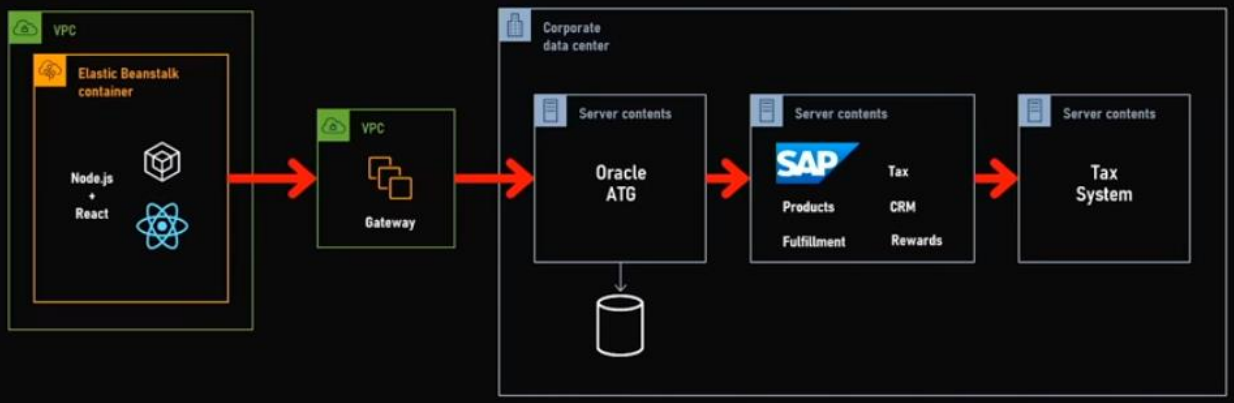
Journey of a bug fix



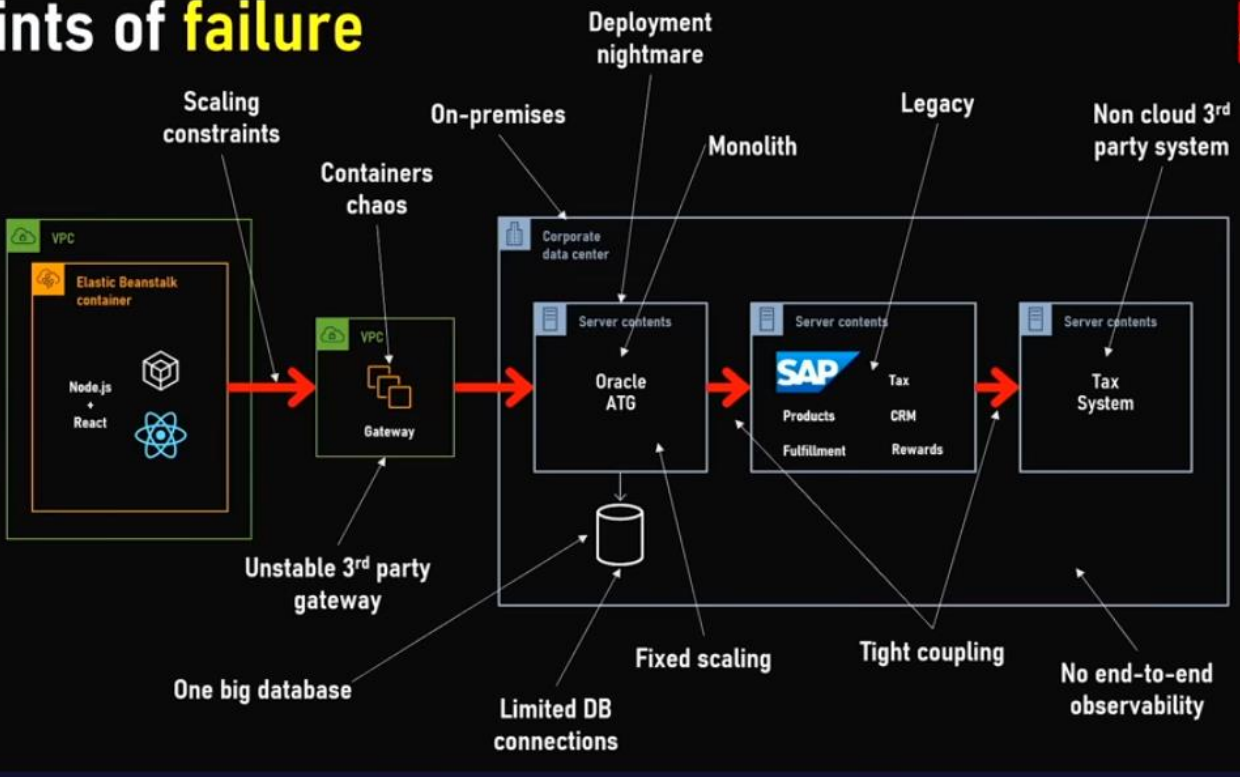
WEB-107 Raised	Dev	QA	Done	8.11 Release
28 Nov 2016	16 Mar 2017	24 Mar 2017	27 Mar 2017	24 May 2017

Failure Days 2017

Black Friday with a monolith



Points of failure



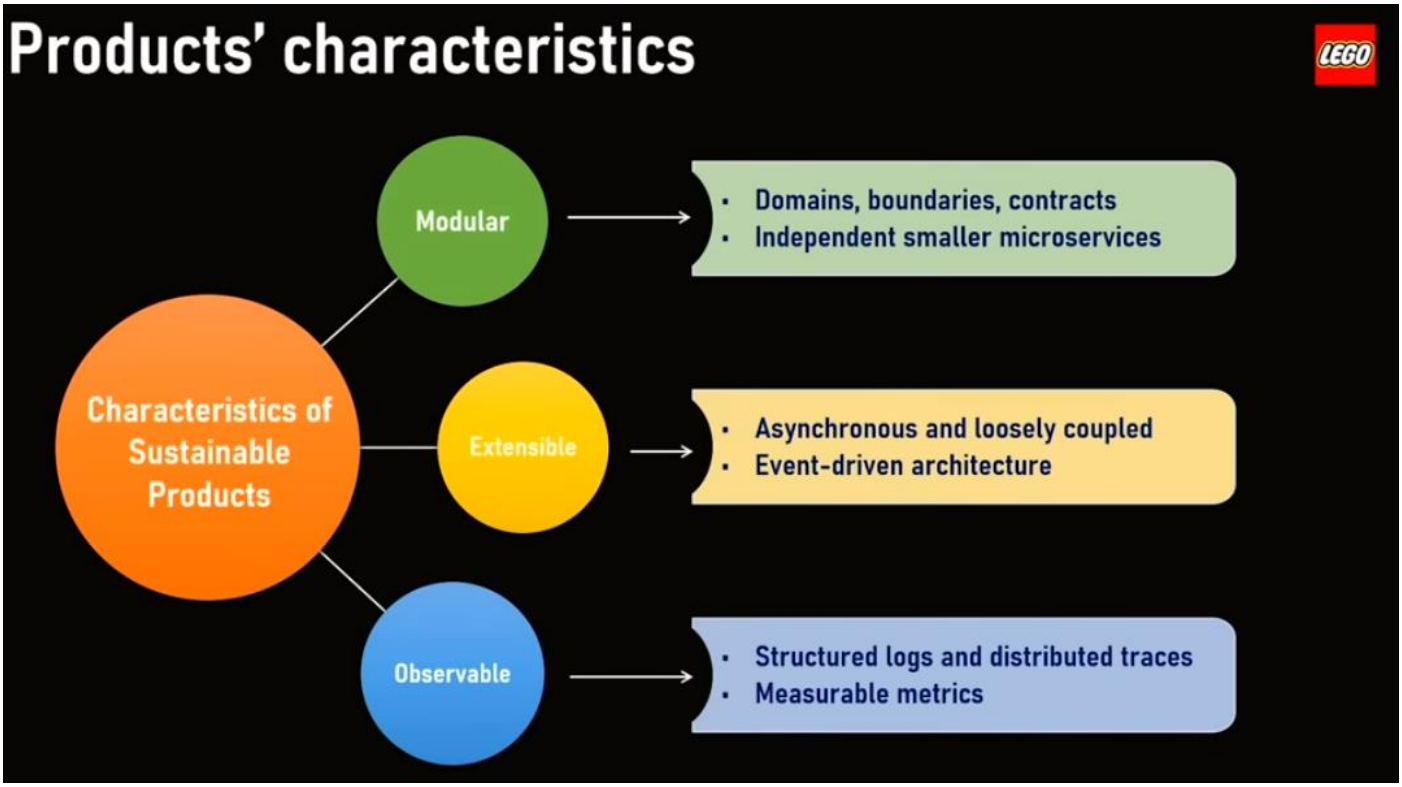
Future Thinking

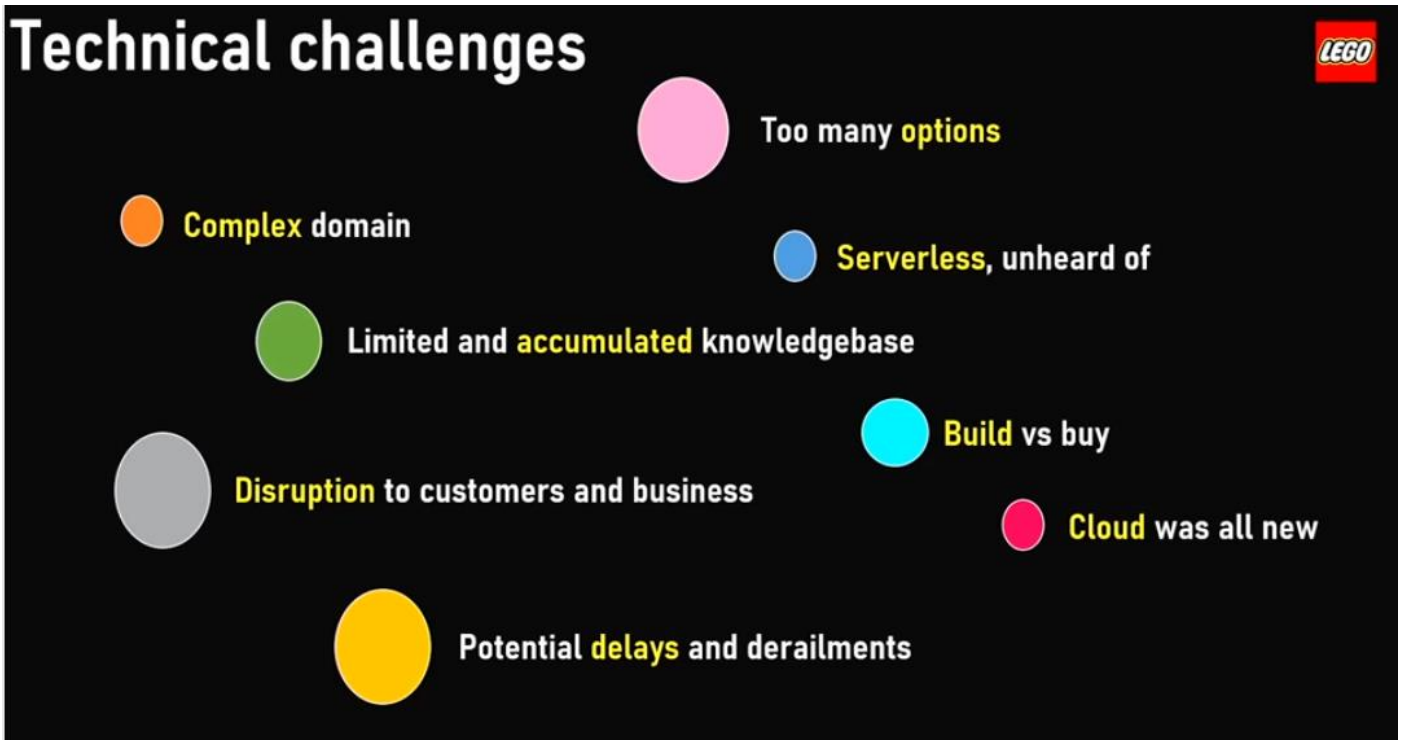
Business demands

Business Expansion	Reach more children - Enter new markets
Development Agility	Minimum Viable Product Product innovation – Team velocity – MVP to MVP Maximum Valuable Product
Operational Visibility	Platform observability – Proactive monitoring - DevOps

Technical needs

- Scalability → How can the platform scale independently?
- Resiliency → How can handle peaks & troughs automatically?
- Extensibility → How can innovate for customers with freedom?
- Availability → How can global customers use the platform 24x7?
- Observability → How can we be proactive & make data-driven decisions?


Products' characteristics
Characteristics of Sustainable Products
Modular
Domains, boundaries, contracts
Independent smaller microservices
Extensible
Asynchronous and loosely coupled
Event-driven architecture
Observable
Structured logs and distributed traces
Measurable metrics


Technical challenges
Too many options
Complex domain
Serverless, unheard of
Limited and accumulated knowledgebase
Build vs buy
Disruption to customers and business
Cloud was all new
Potential delays and derailments

Speedy adoption

start simple

async & event-driven

iterate, evolve

follow proven patterns

Racing ahead

solution first;
perfection next

can't take risk

vendor lock-in fear

continuous
improvement

Falling behind

wait for a perfect
moment

invest in expansive
processes

judgmental

Endless experimentation

The first step!

"No one starts perfect with Serverless, but strive to be better at every iteration, and that is important."

Do It

Do It Right

Do It Better

- Allen Helton

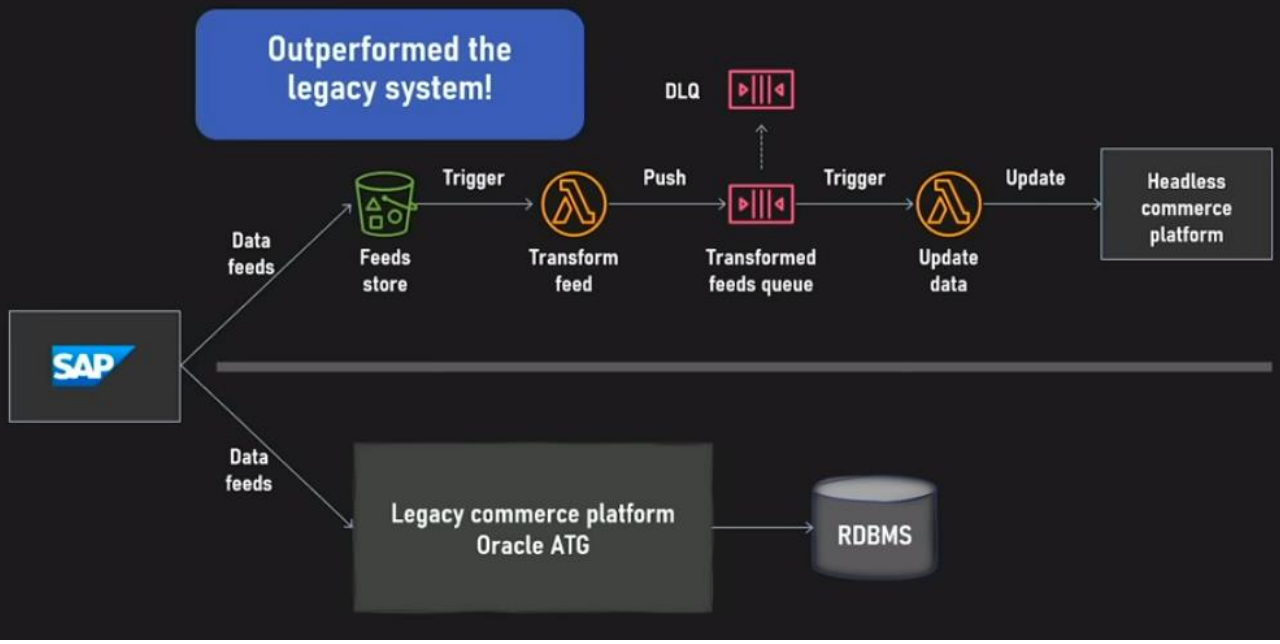
https://bit.ly/3wfmwLr

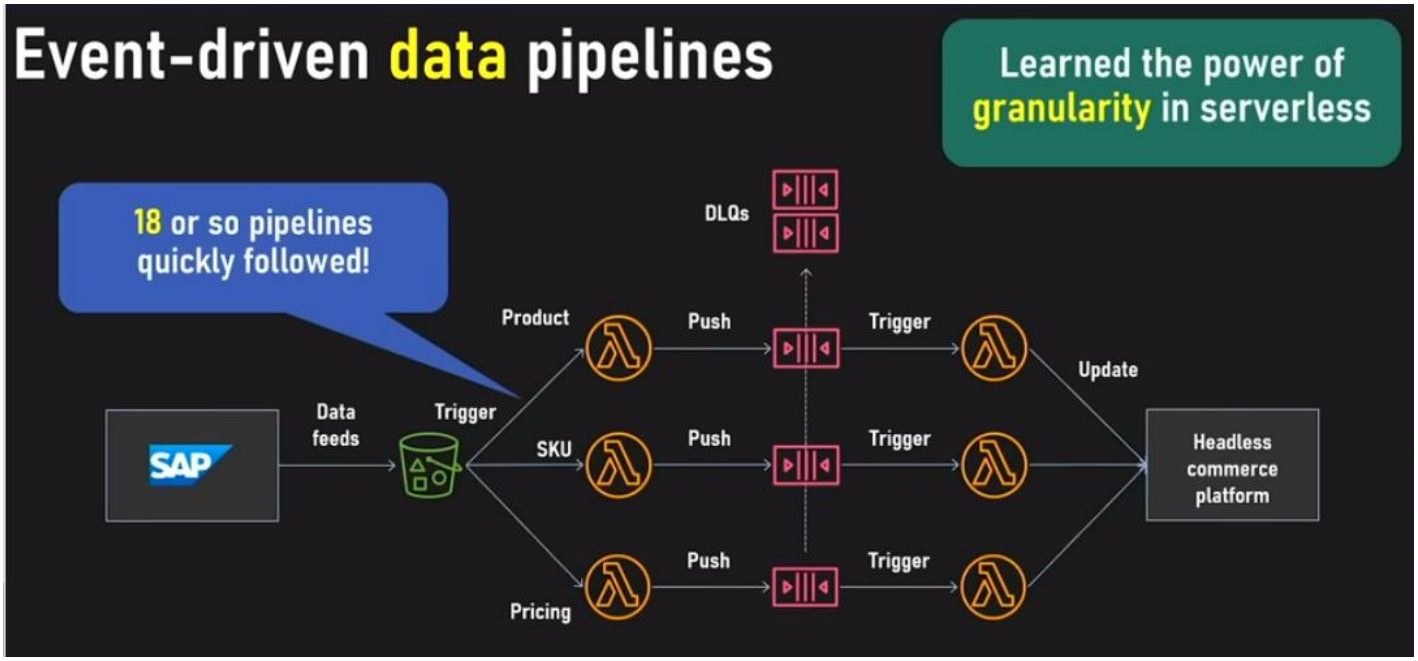
If you spend too much thinking about a thing,

you will never get it done

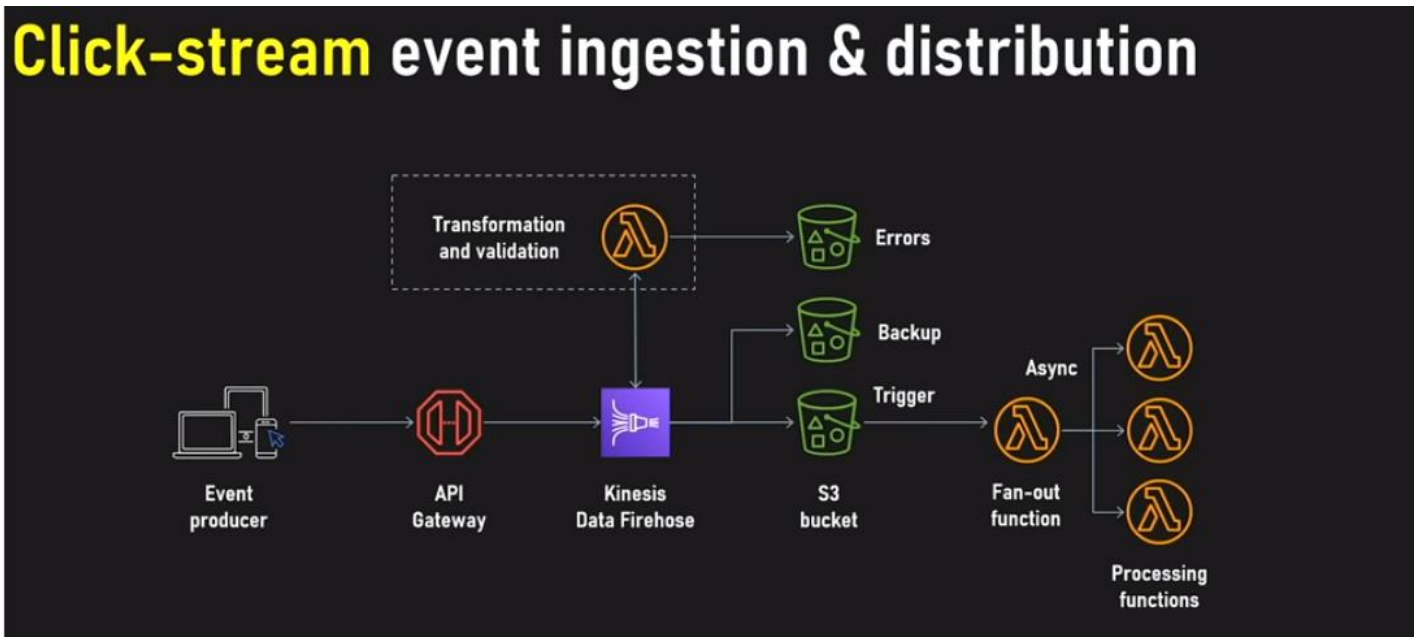
- Bruce Lee

First taste of event-driven computing

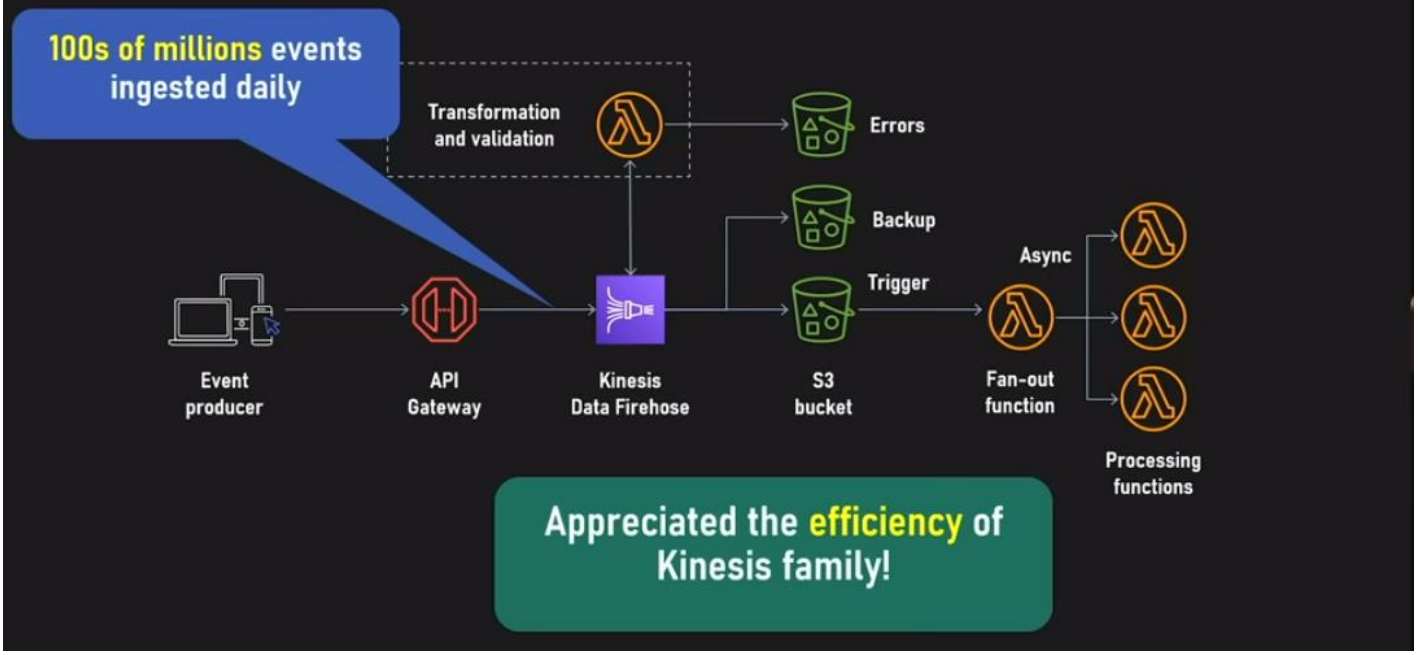




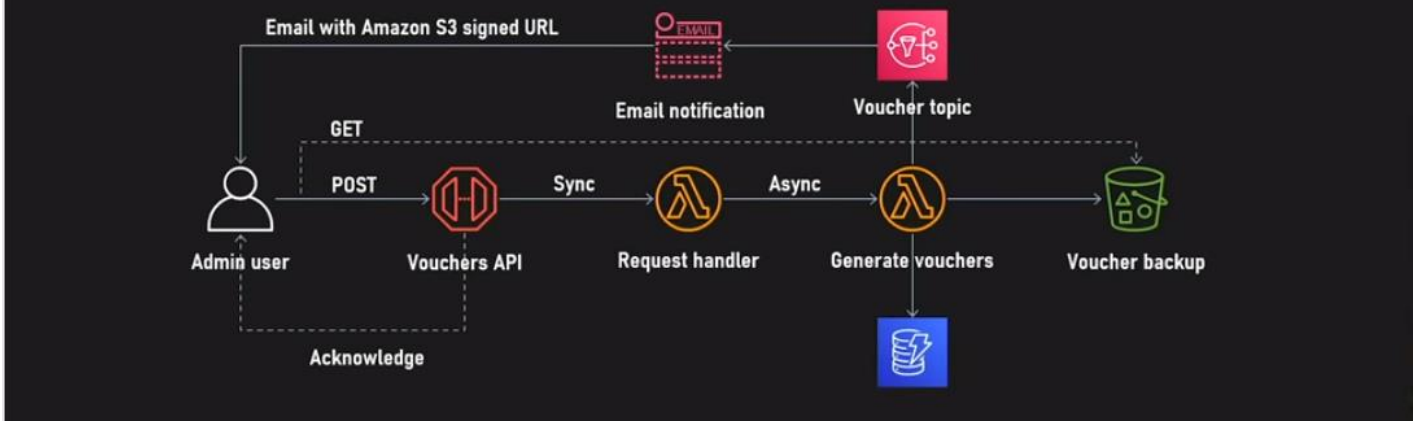
We were able to individually tuned these different queues and lambda functions as required for each pipeline



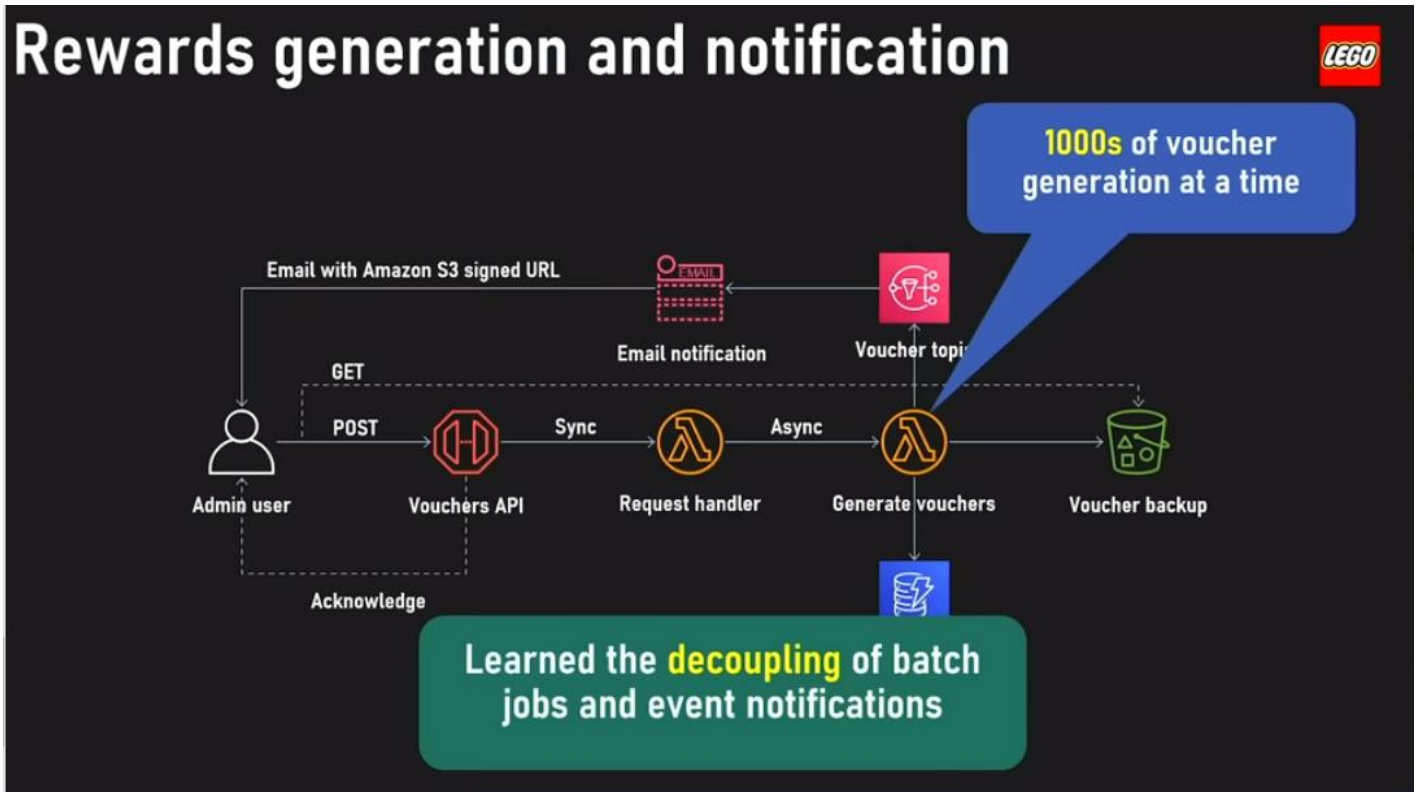
Click-stream event ingestion & distribution



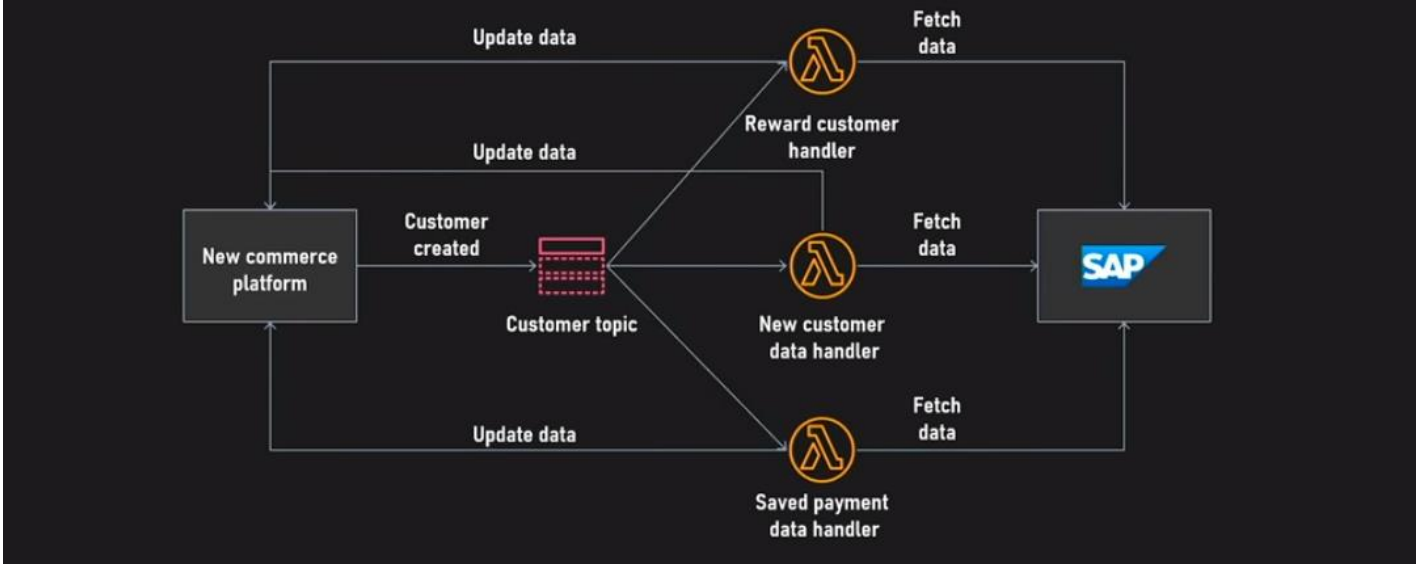
Rewards generation and notification



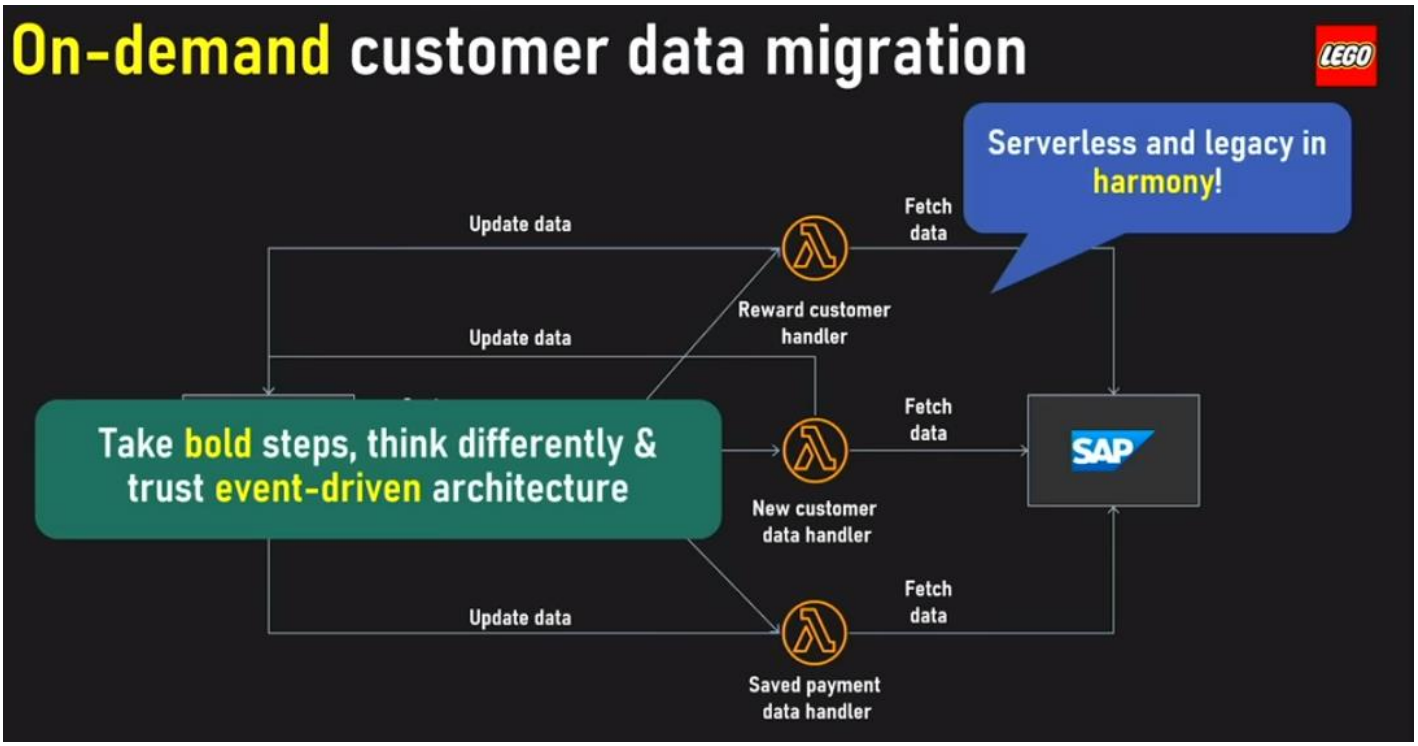
Rewards generation and notification

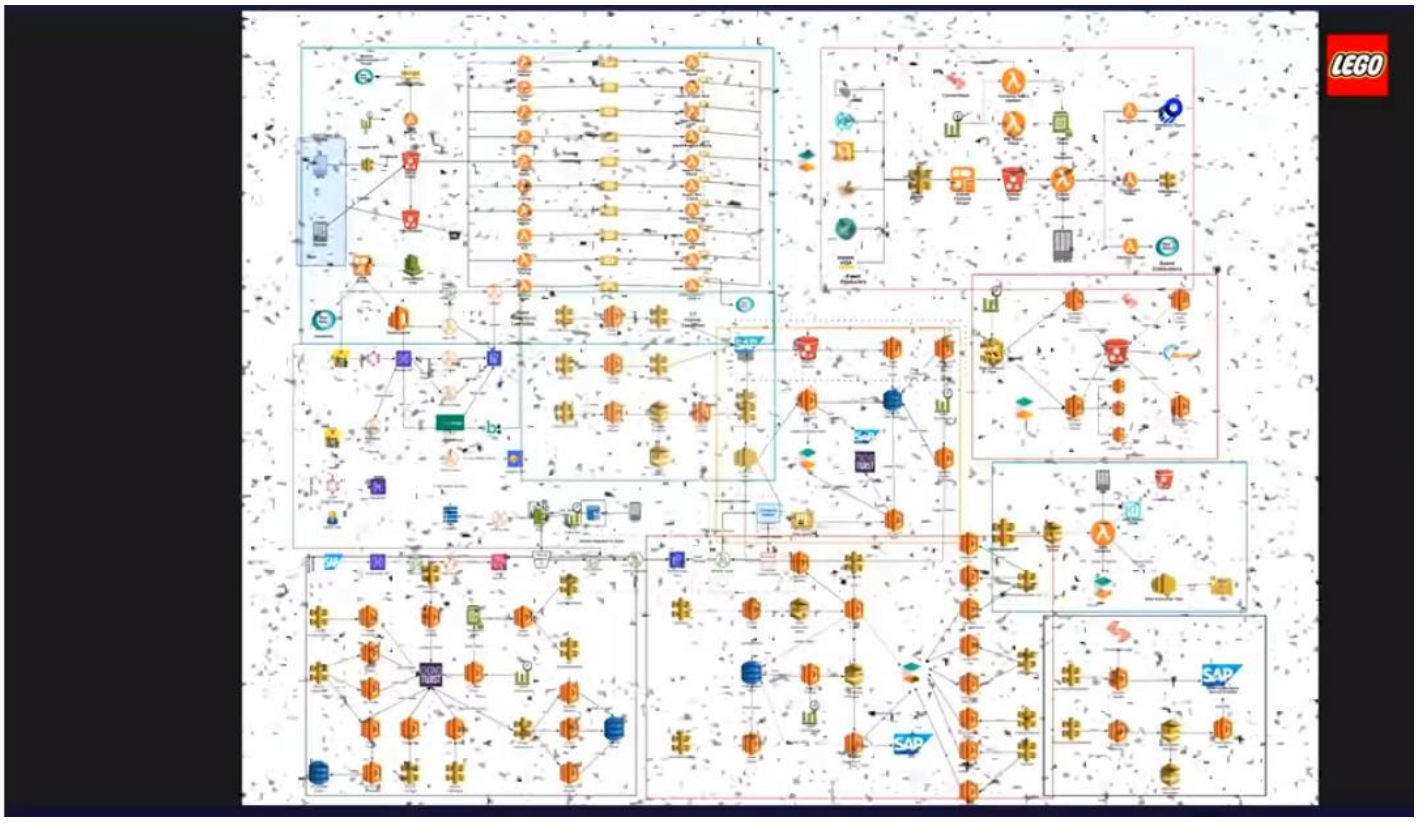


On-demand customer data migration



On-demand customer data migration





Wait… where is EventBridge?

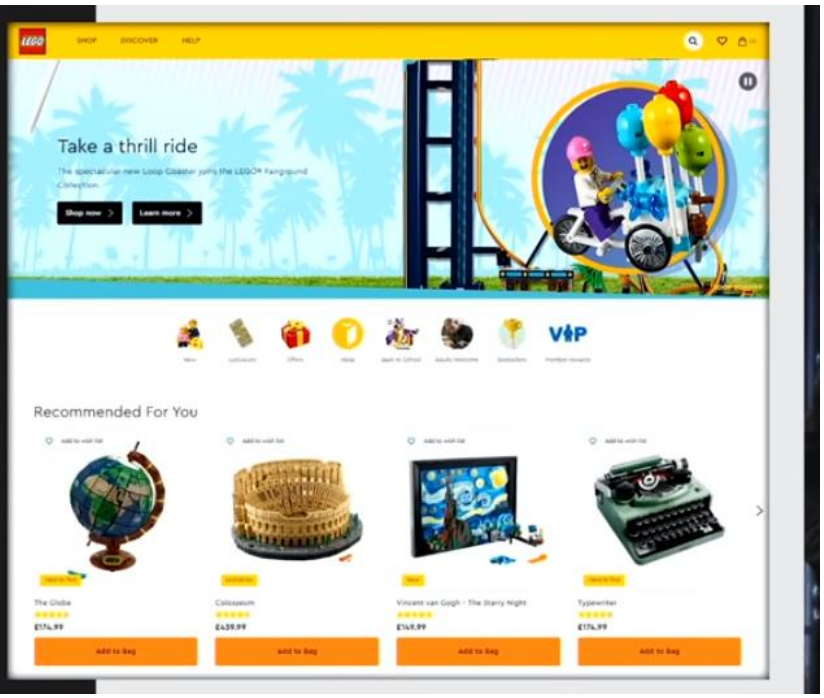
It wasn't there!

10, July 2019

LEGO.com

was switched to

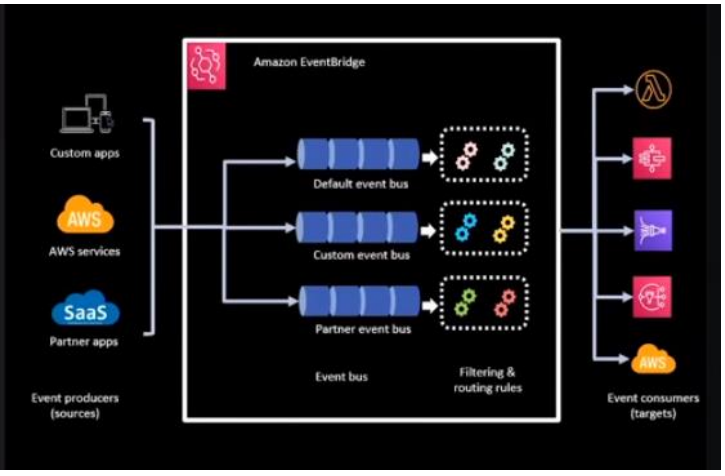
serverless



11, July 2019

Amazon EventBridge

was released

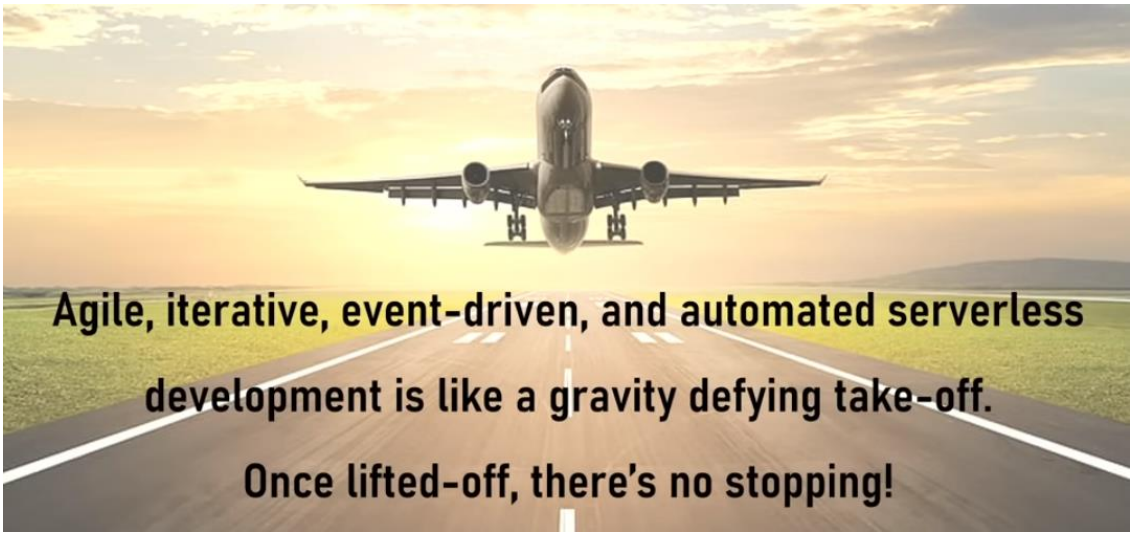


EDA Phase 2

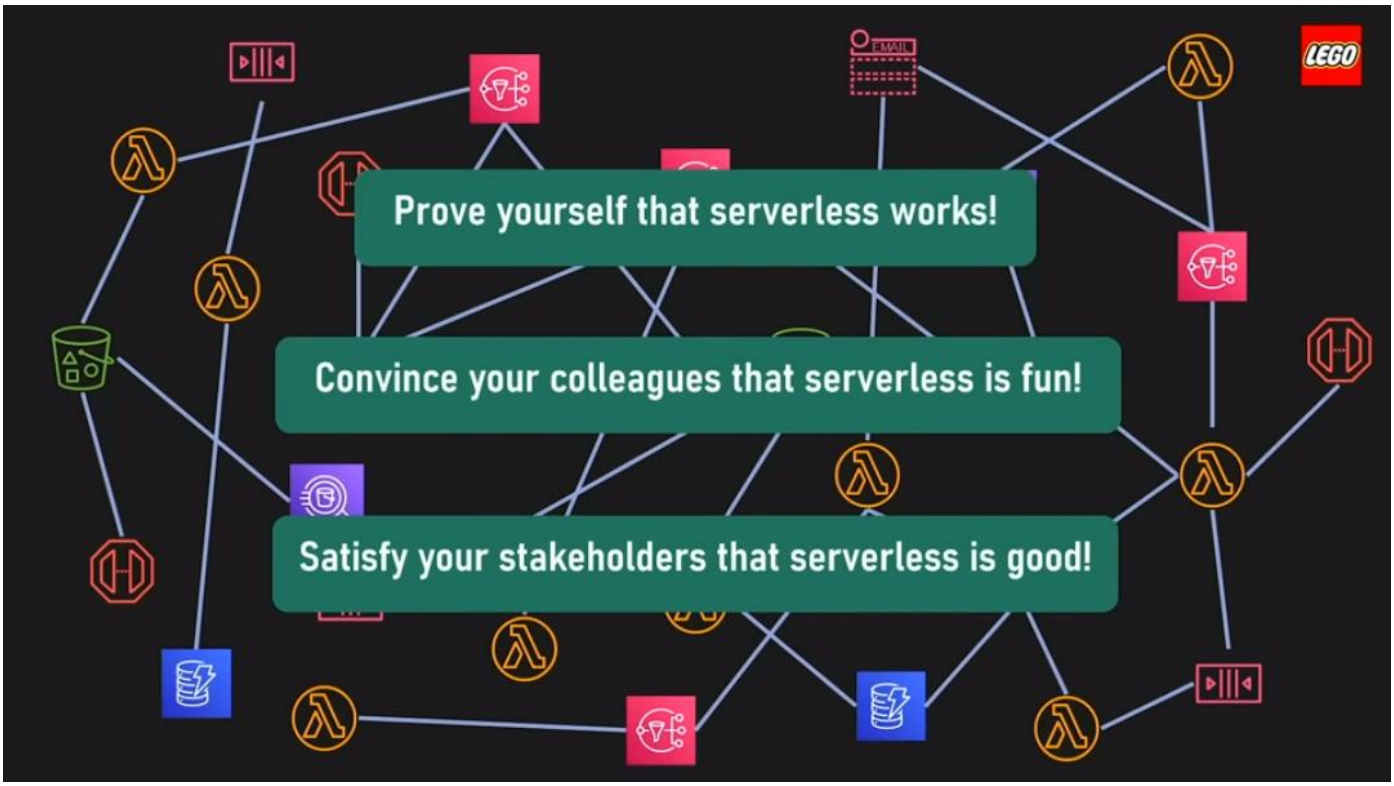
The event-driven acceleration

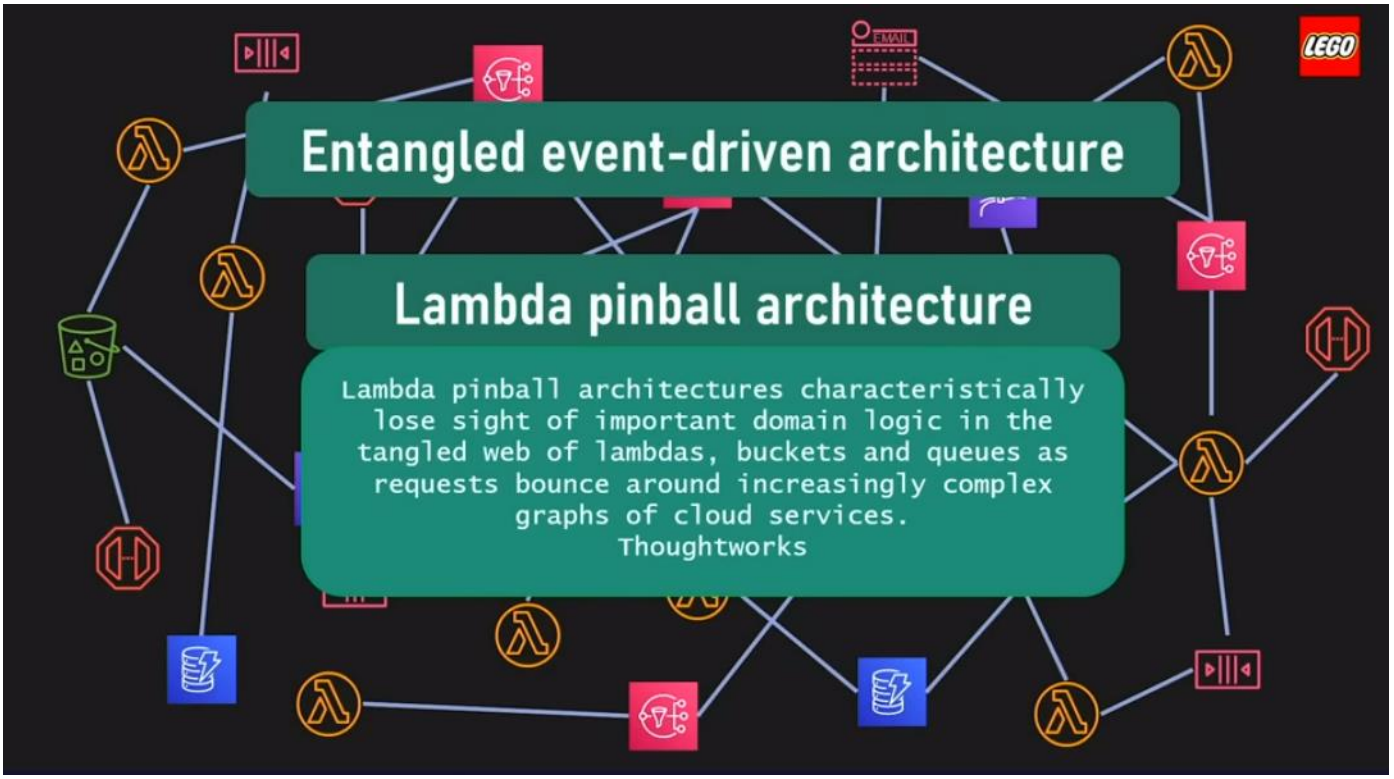
Serverless computing enables to build event-driven systems faster, because it speeds the feedback-cycle so that the system iterates more efficiently.

Dave Anderson, author of 'The Value Flywheel Effect'



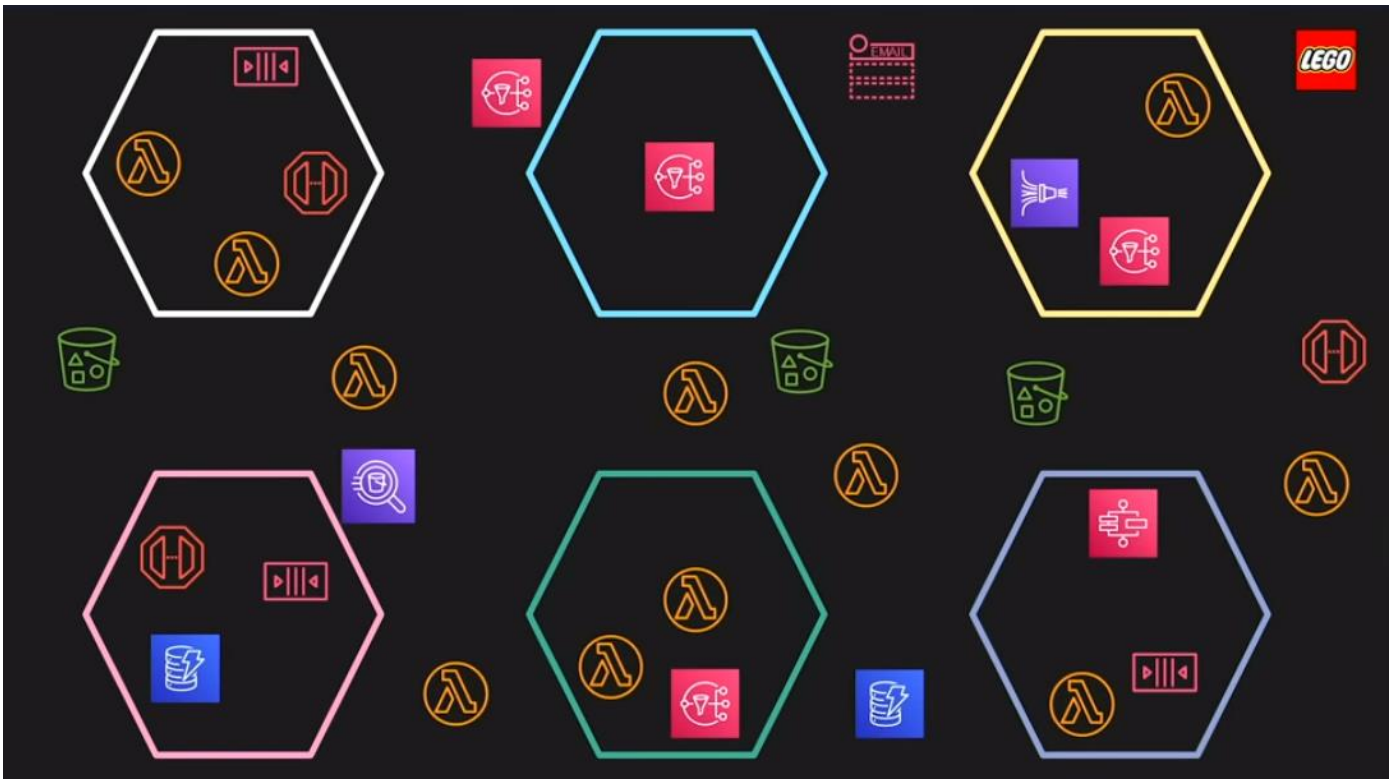
It's all about event-driven architecture

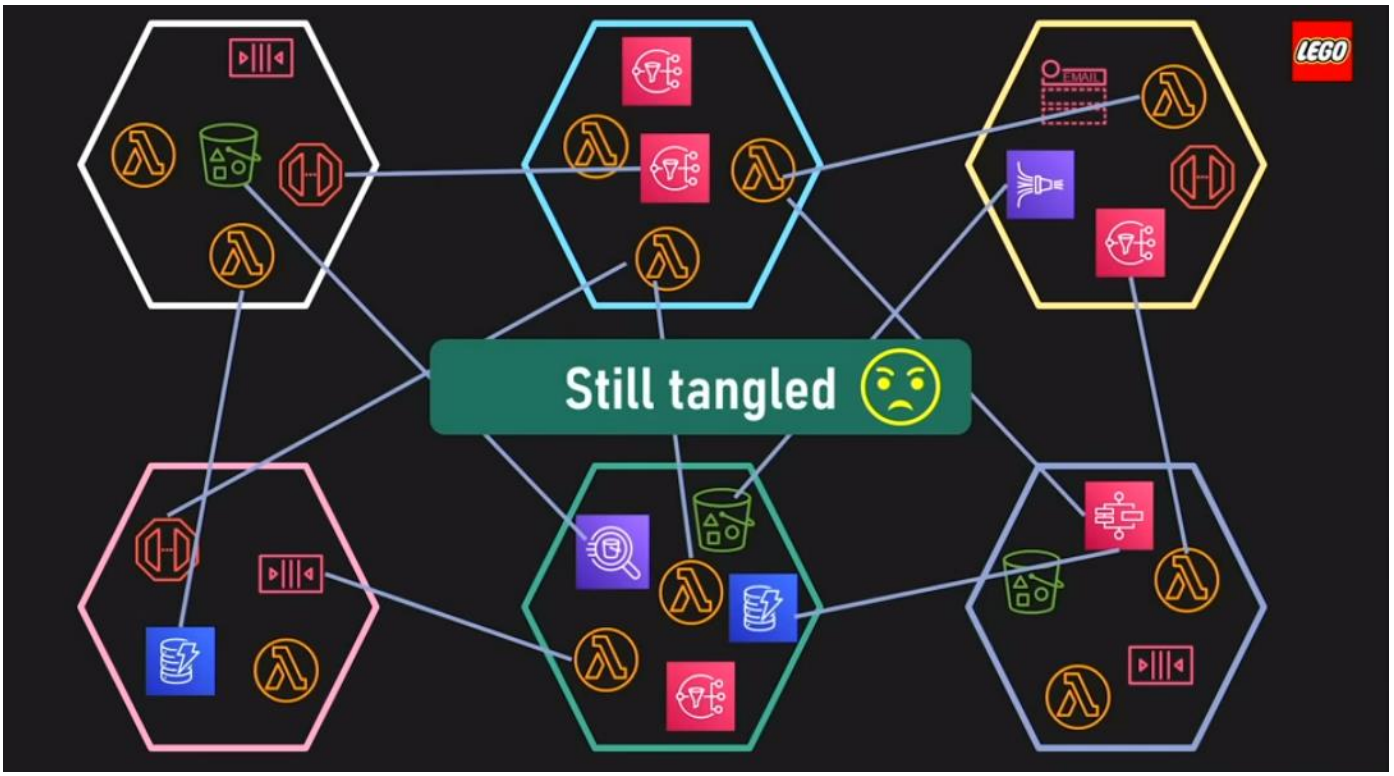


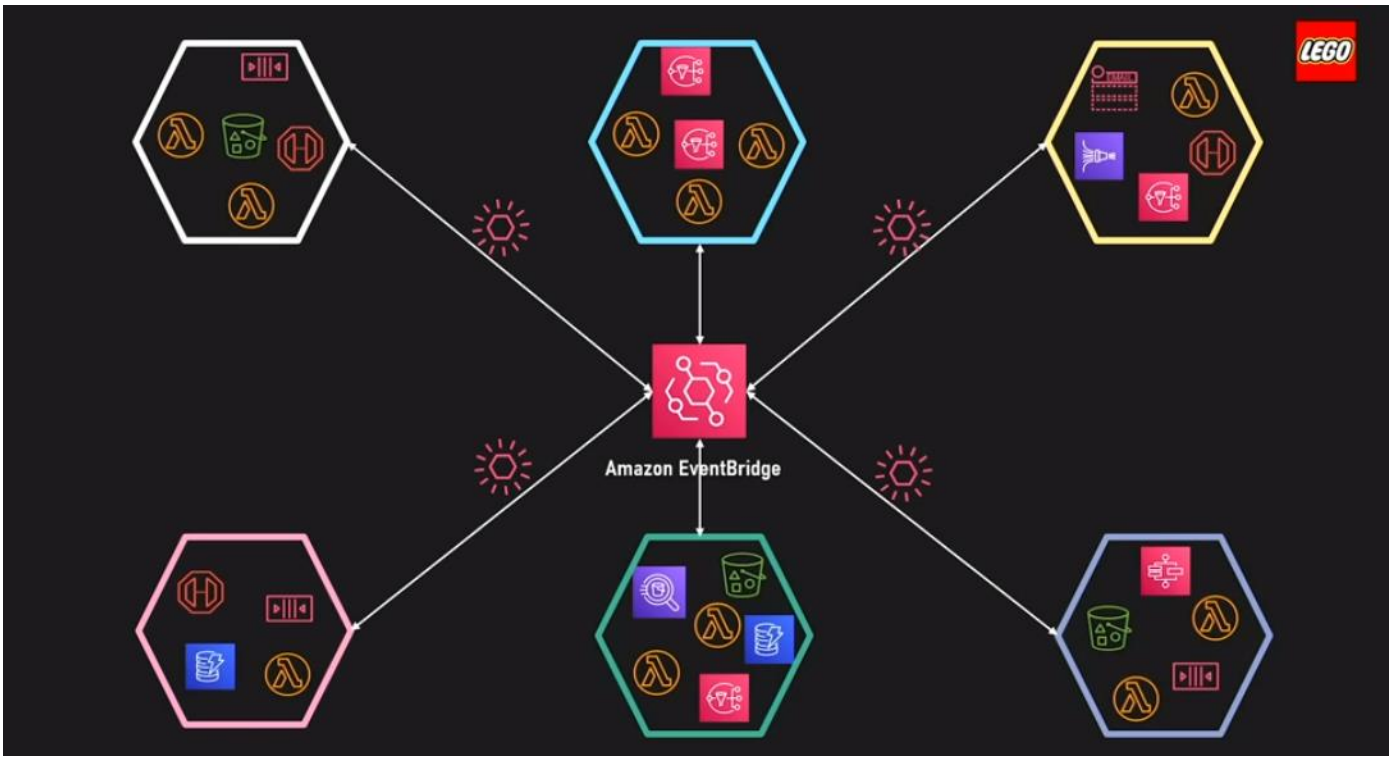


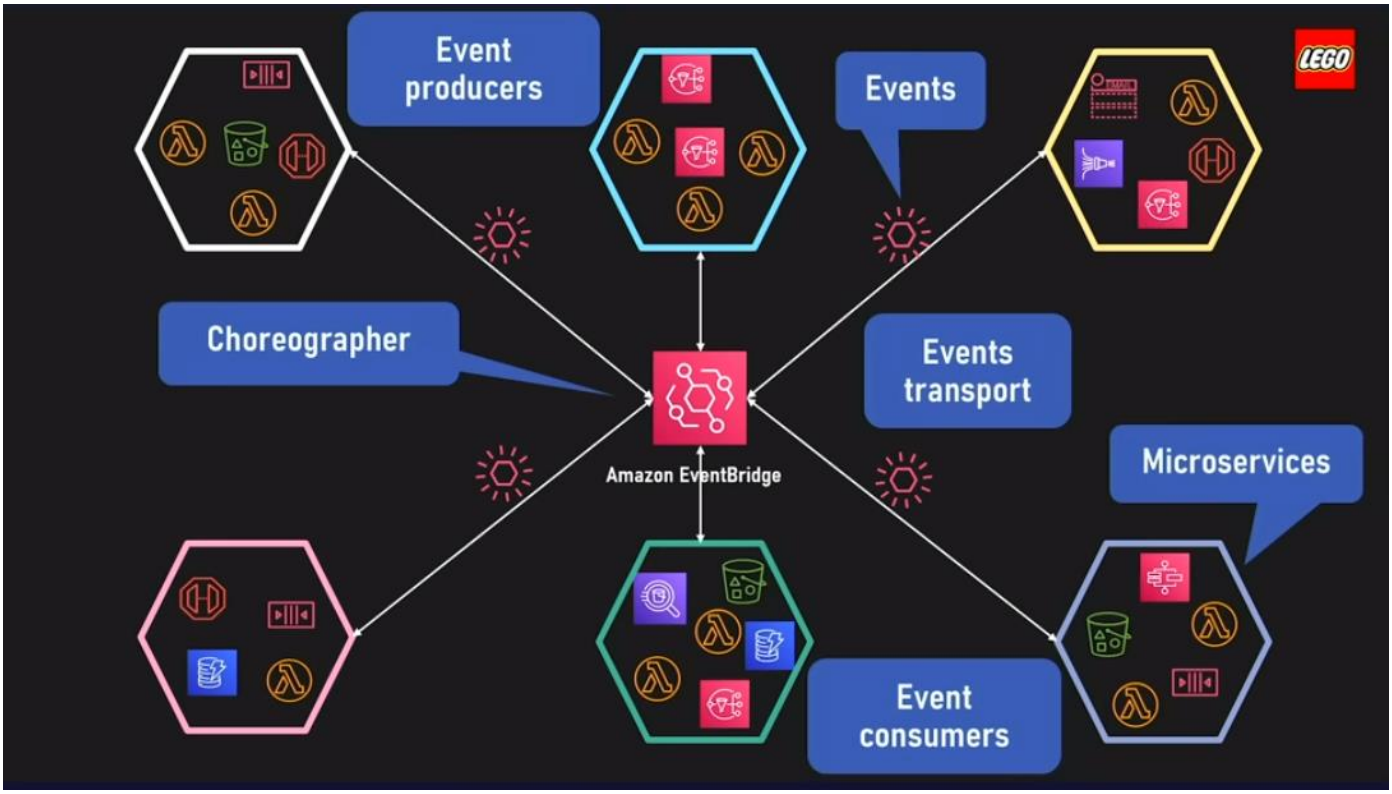
What's the solution?

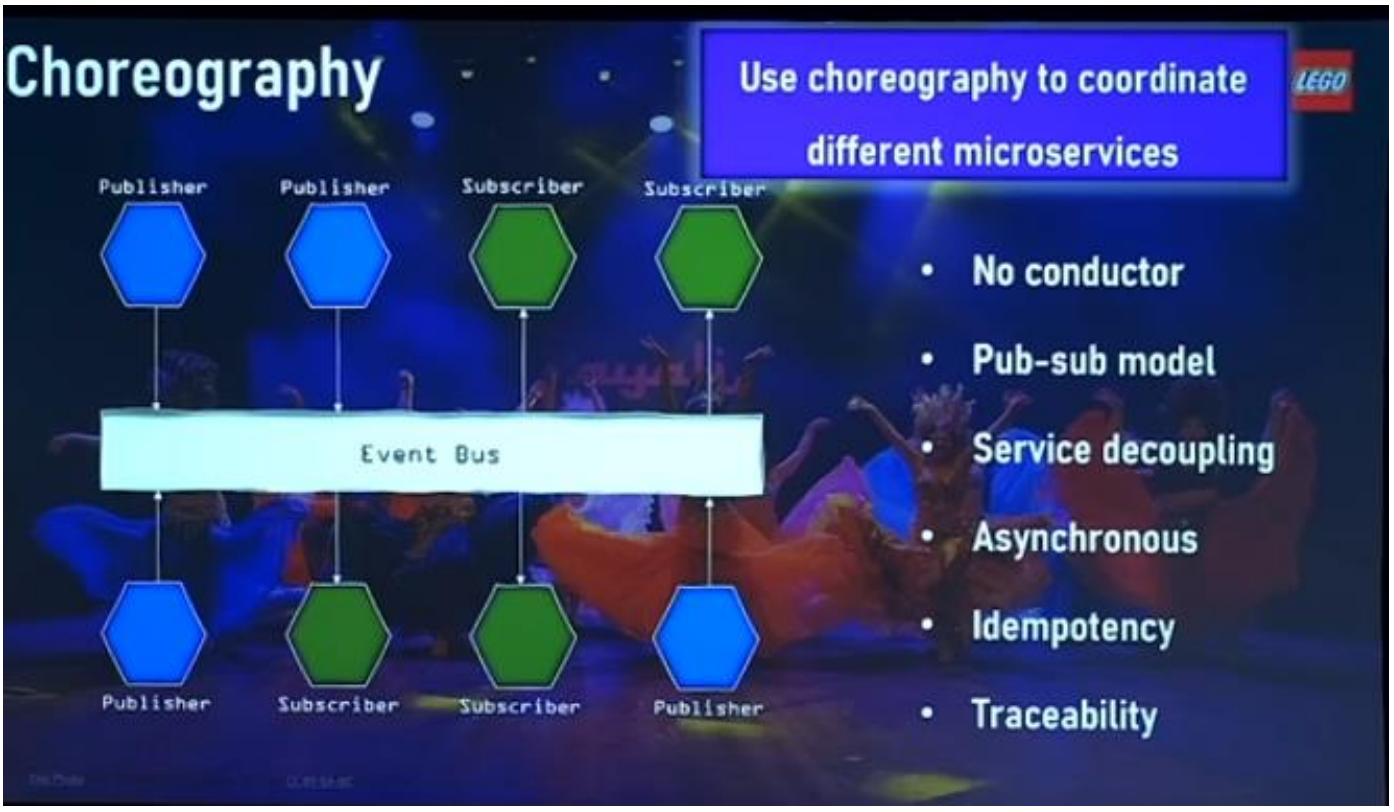
Microservices! Yay!!



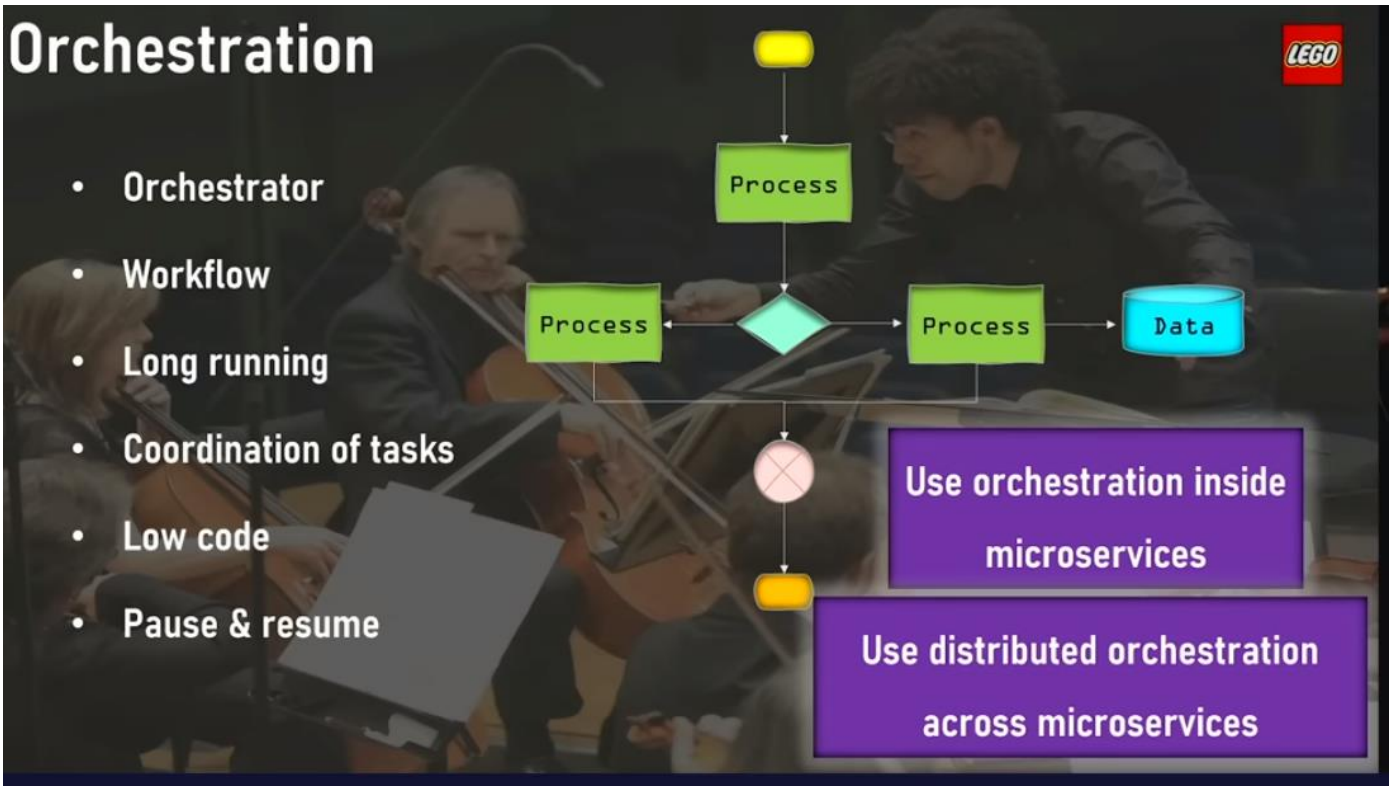

LEGO
Still tangled


LEGO
Amazon EventBridge





The subscribers and publishers know what to do when they send or get events

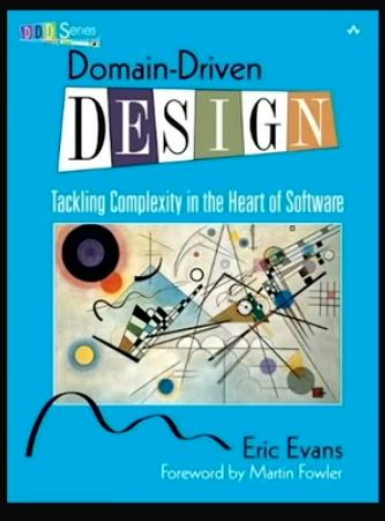

Orchestration
Orchestrator
Workflow
Long running
Coordination of tasks
Low code
Pause & resume
Process
Process
Process
Data
Use orchestration inside microservices
Use distributed orchestration across microservices


With choreography and orchestration, magical event-driven architectures evolve!

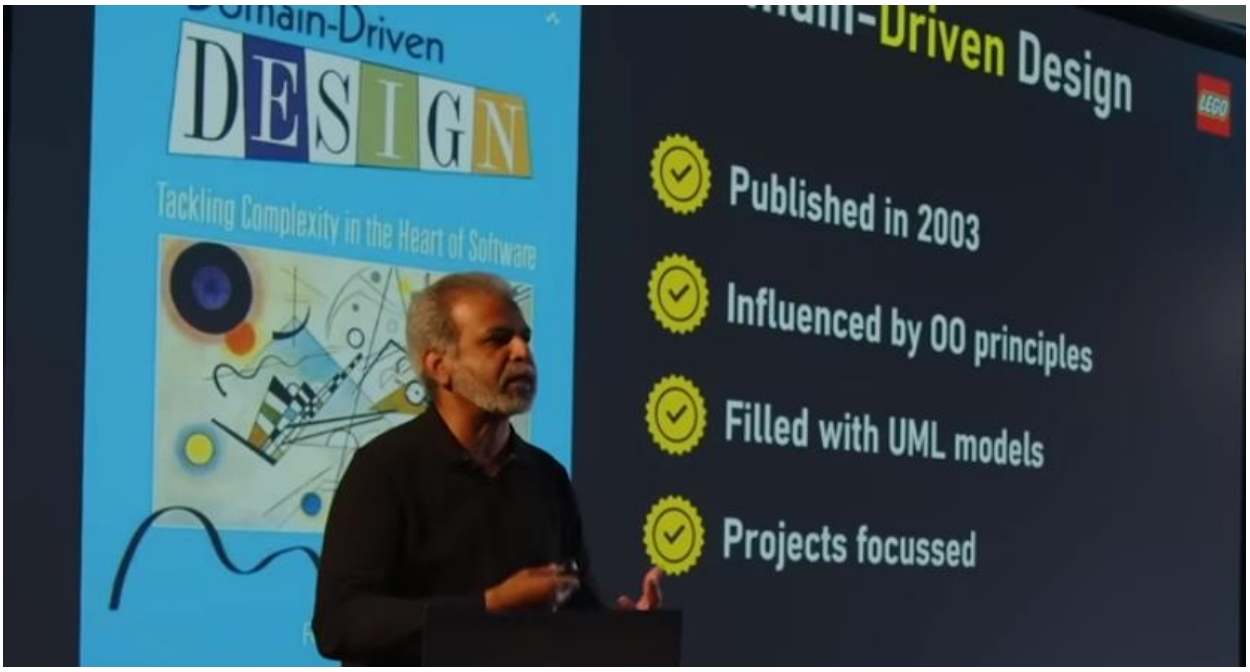
Microservices

Thinking in set-pieces

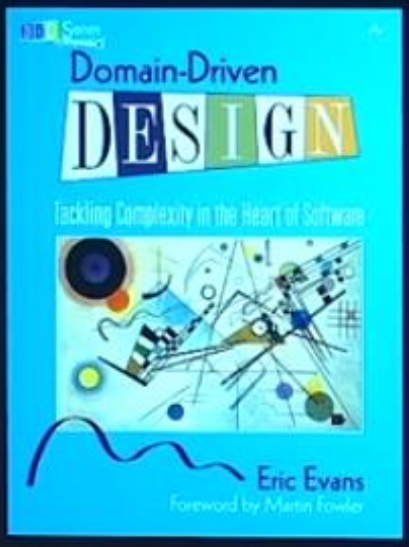
Domain-Driven Design



It's not a book. It's an industry!



Domain-Driven Design



- Domain model
- Supple design
- Bounded context
- Anticorruption layer
- Distillation of core domain

A system that is hard to understand is hard to change

Someone on a talk about DDD and an overload of Java code examples...

I would have gotten excited about this ... 20 years ago!

My recommendation to new generation developers:

- switch to microservices, focus on the API/contracts
- avoid sync calls between microservices
- try to do as much as possible event-driven
- go functional
- domain model has no dependency on anything

If domain thinking is like seeing the forest...

Set-piece thinking is like looking at the trees

See the forest before looking at the trees.

Visualise the whole but focus in parts.

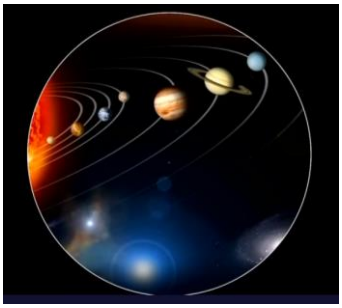
I just don't think it's that simple. Nothing is.

Everything is, when you break it down!

Mama Mia! Here We Go Again

Vision and Focus






What is a set-piece?

A scene or sequence of scenes whose execution requires logistical planning

A realistic piece of stage scenery built to stand independently as part of a stage set

A rehearsed team manoeuvre practised in training in advance of matches

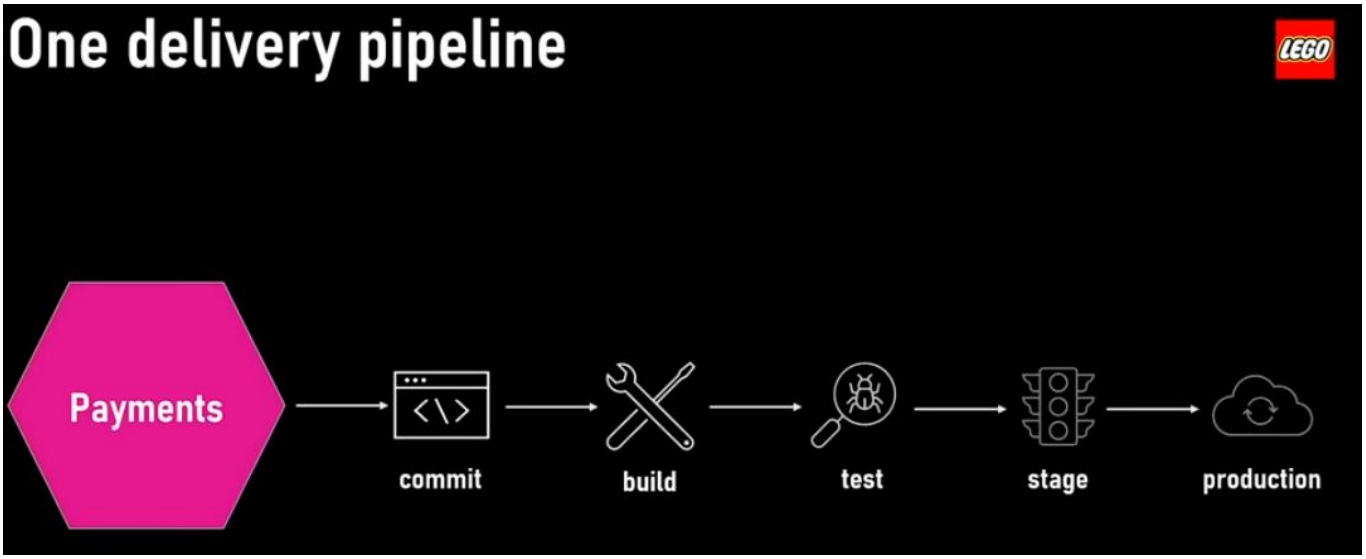
Parts of a model that are built separately and assembled together

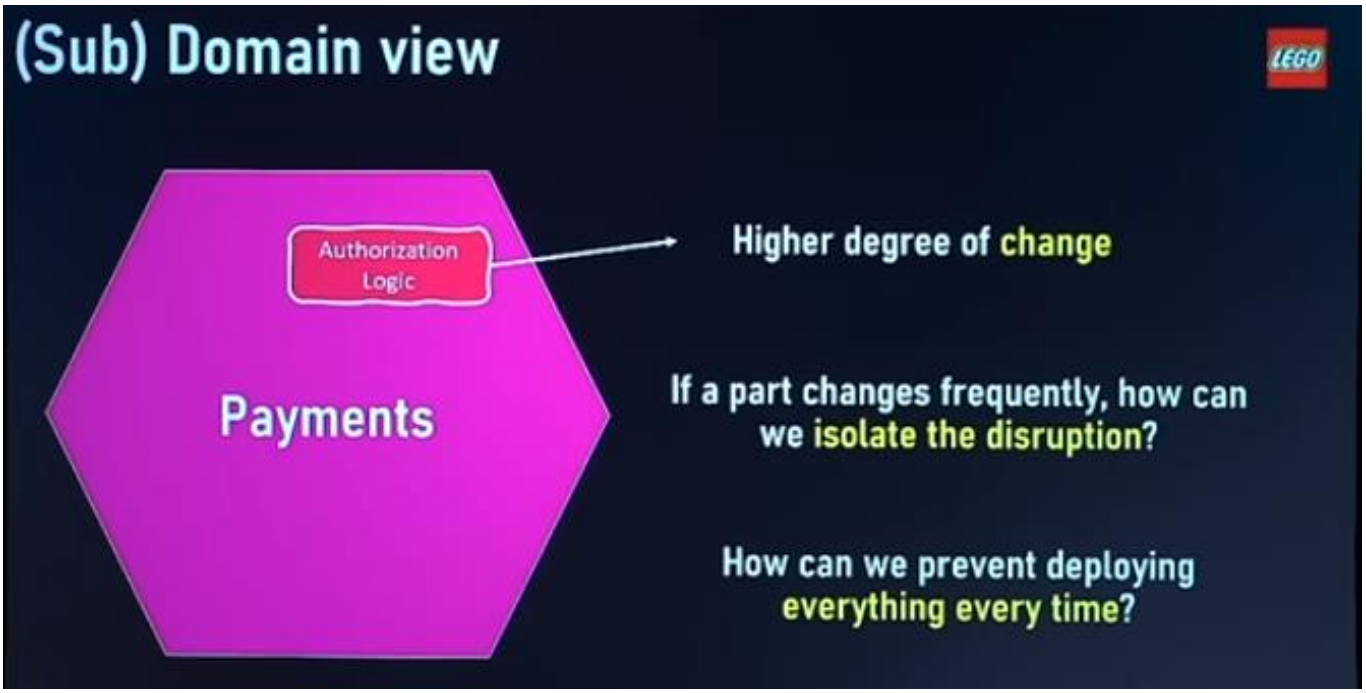
Developing as Set-Pieces requires planning and execution, but can be developed and tested independently, and brought together.

(Sub) Domain view

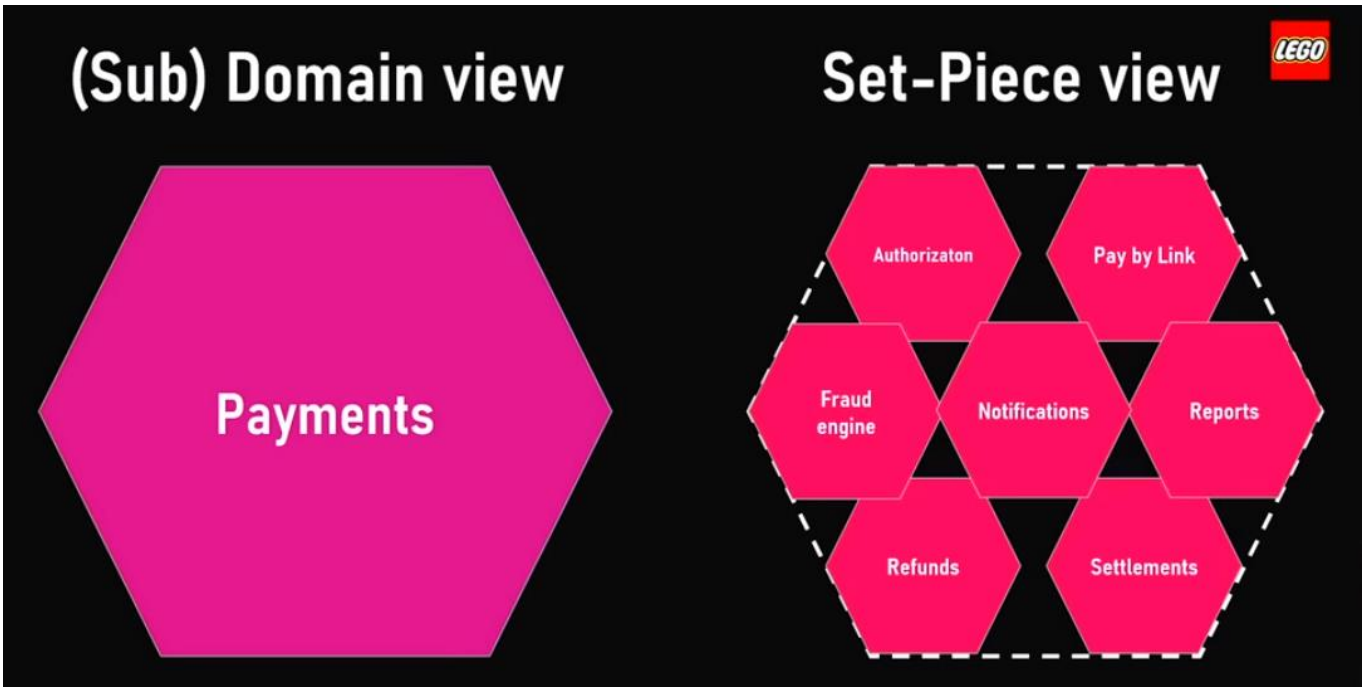
Payments

- One monolith single service
- One delivery pipeline
- One unified complex view

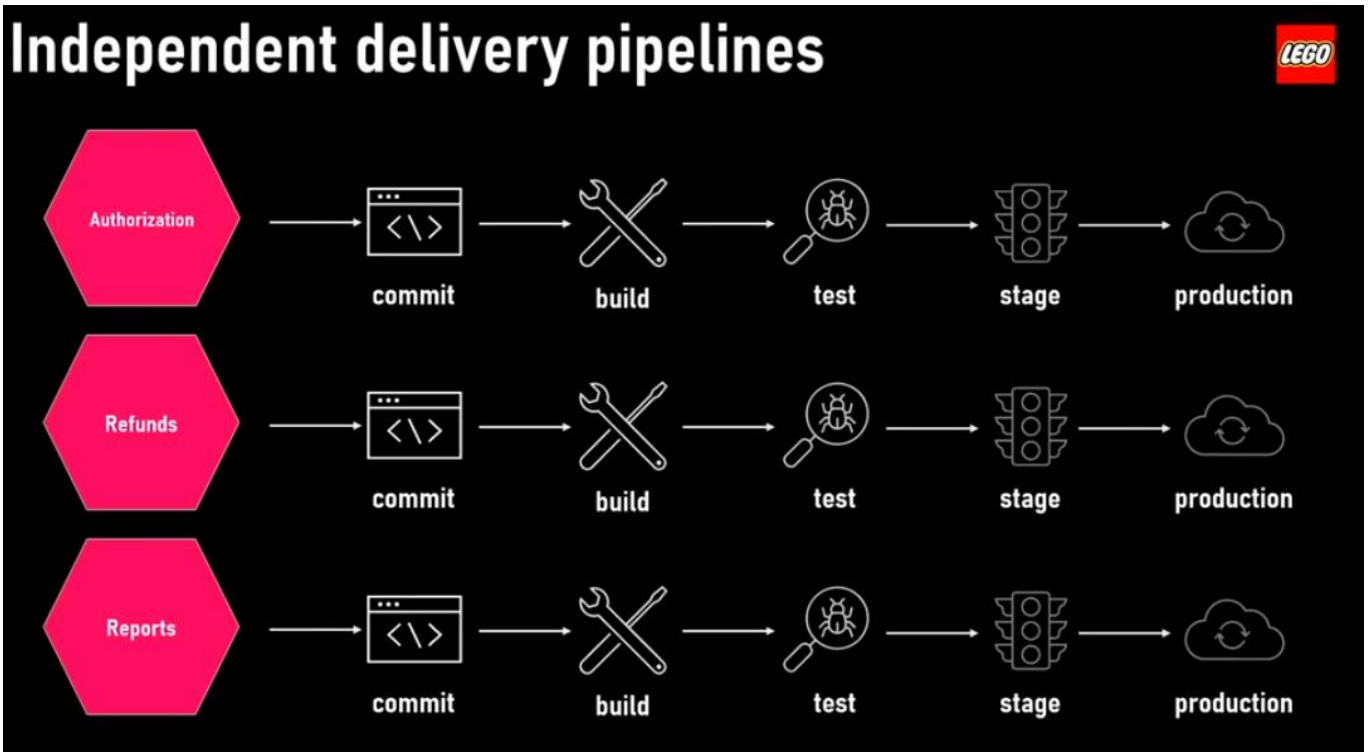


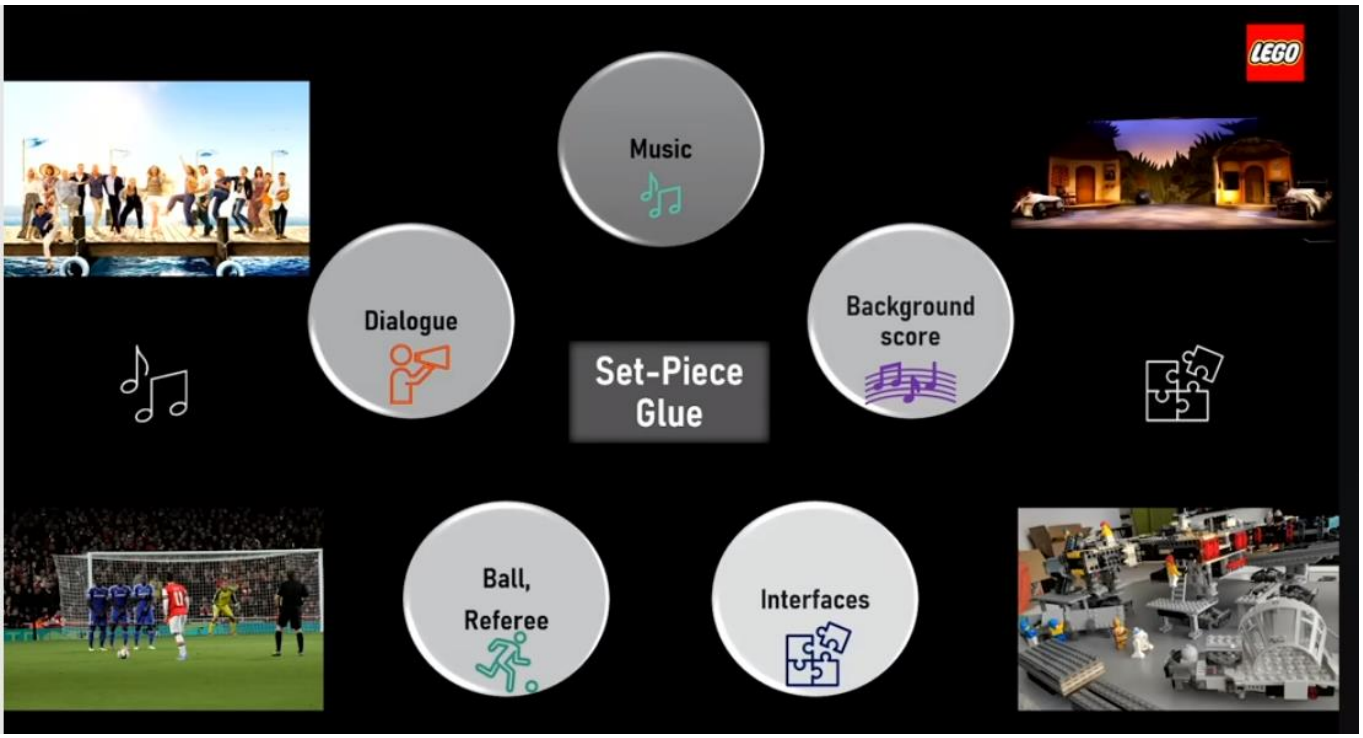


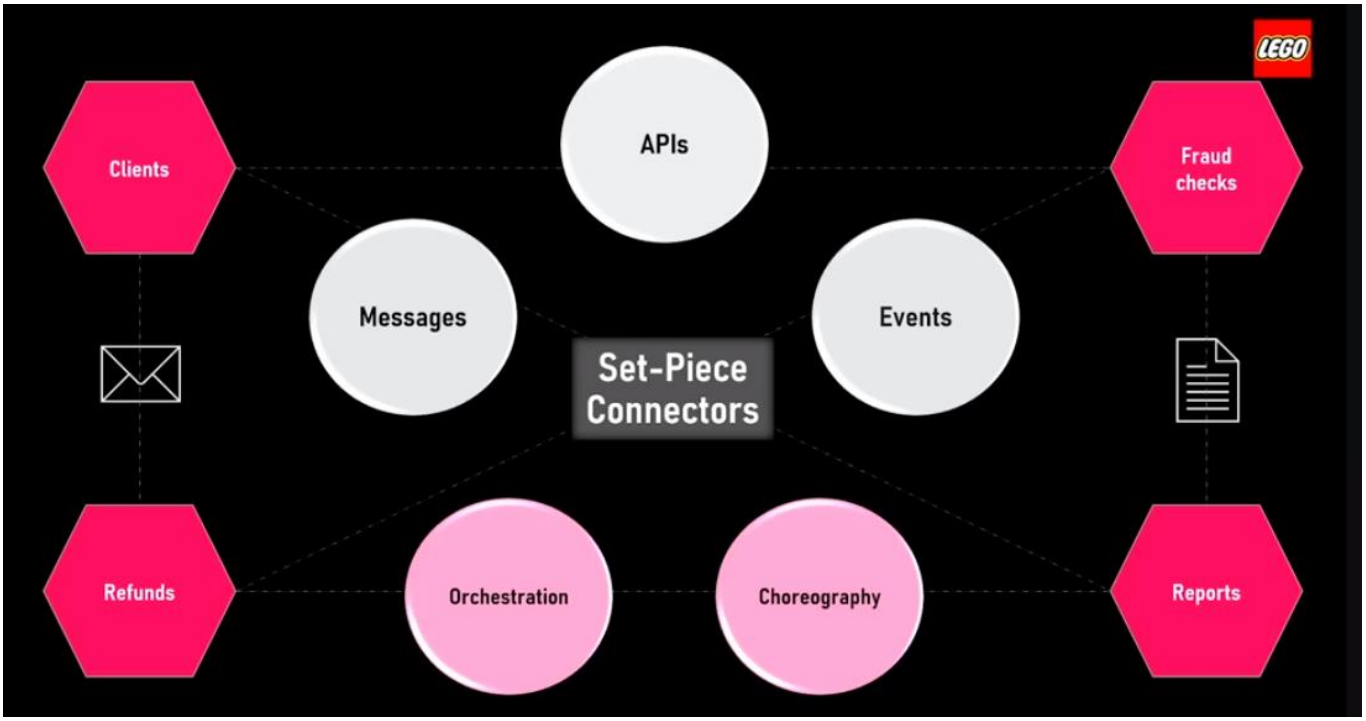
What if a part of it changes all the time? Can you separate it out?

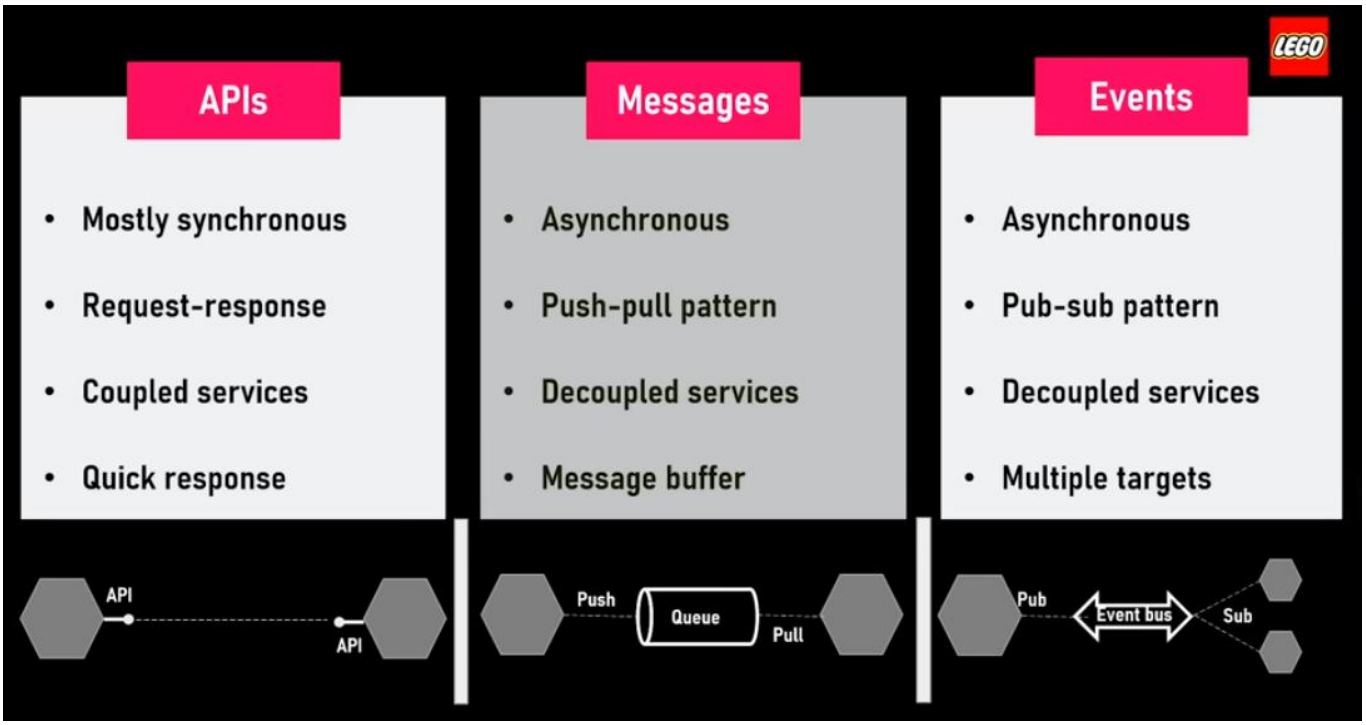


Note that they are all still within the boundary and context of the Payments service

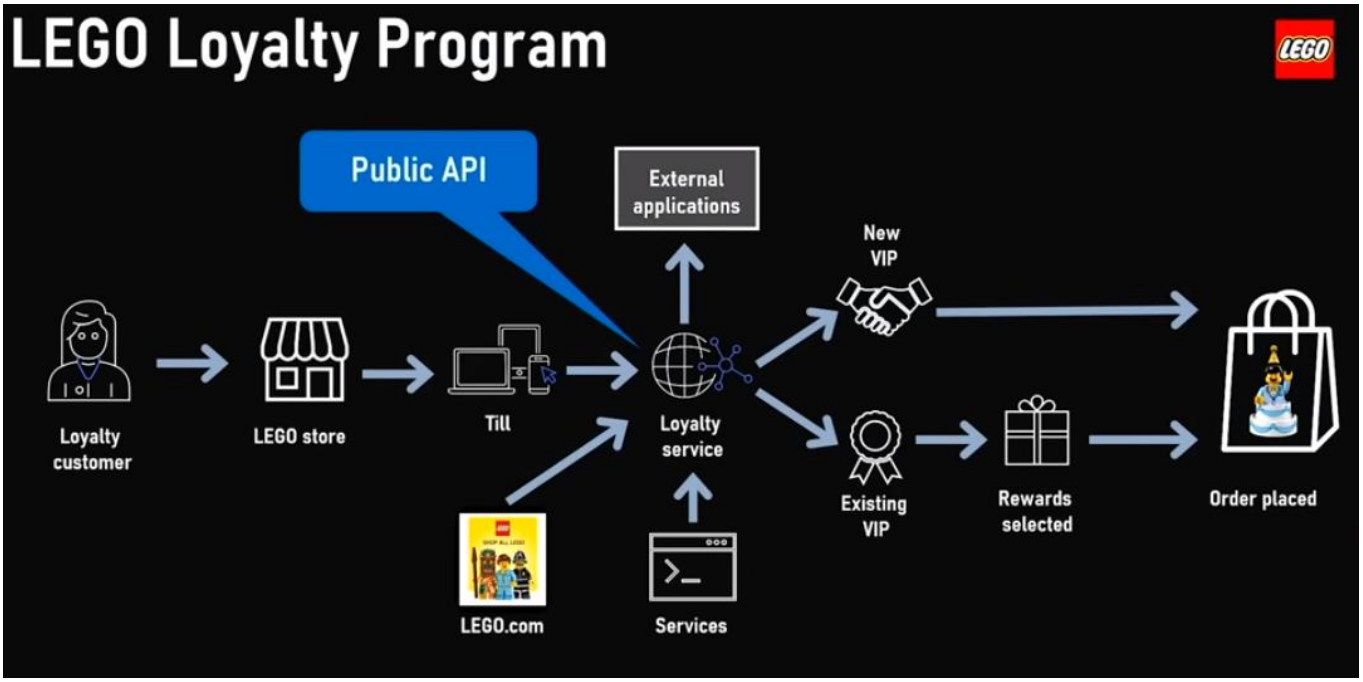


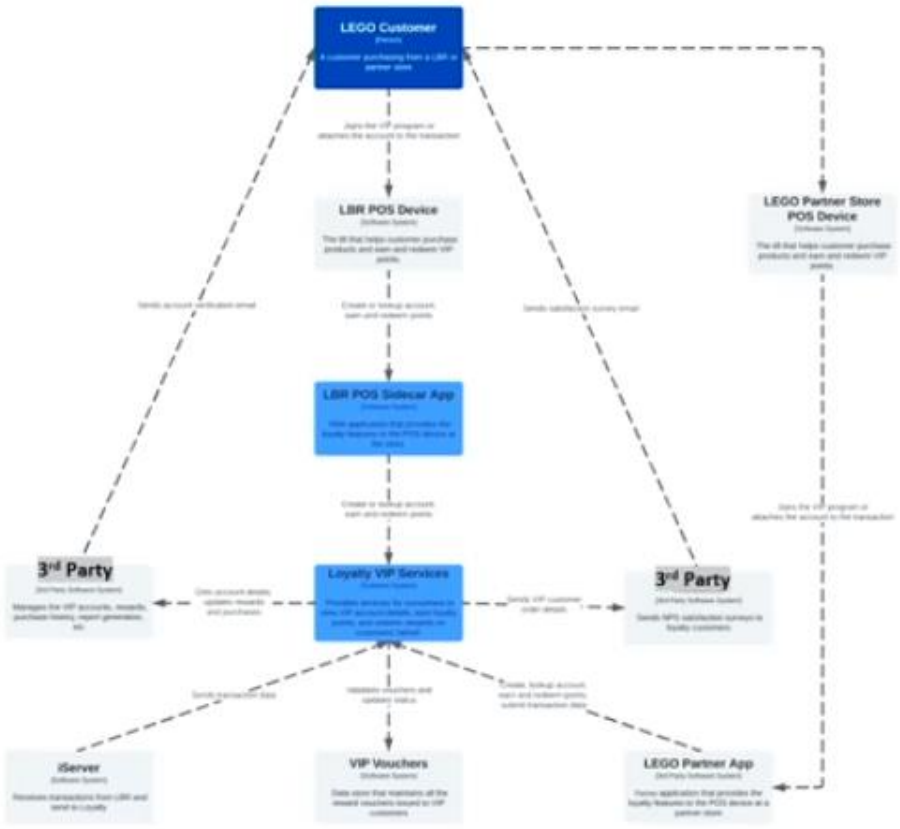

Music
Dialogue
Background score
Set-Piece Glue
Ball, Referee
Interfaces


APIs
Clients
Fraud checks
Messages
Events
Set-Piece Connectors
Refunds
Orchestration
Choreography
Reports


APIs
Messages
Events
• Mostly synchronous
• Request-response
• Coupled services
• Quick response
• Asynchronous
• Push-pull pattern
• Decoupled services
• Message buffer
• Asynchronous
• Pub-sub pattern
• Decoupled services
• Multiple targets
API
API
Push
Queue
Pull
Pub
Event bus
Sub


Set-Piece Case Study
The Loyalty Service

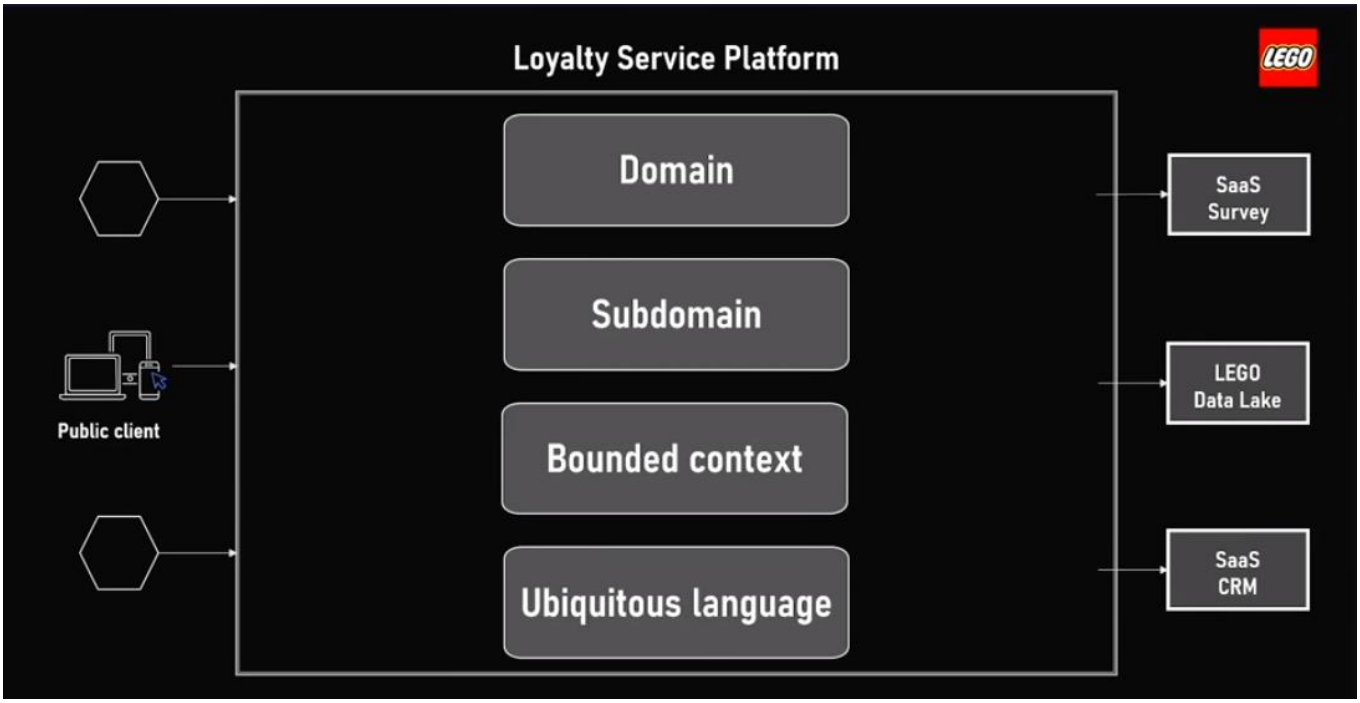

LEGO Loyalty Program
Public API
External applications
New VIP
Loyalty customer
LEGO store
Till
Loyalty service
Existing VIP
Rewards selected
Order placed
LEGO.com
Services

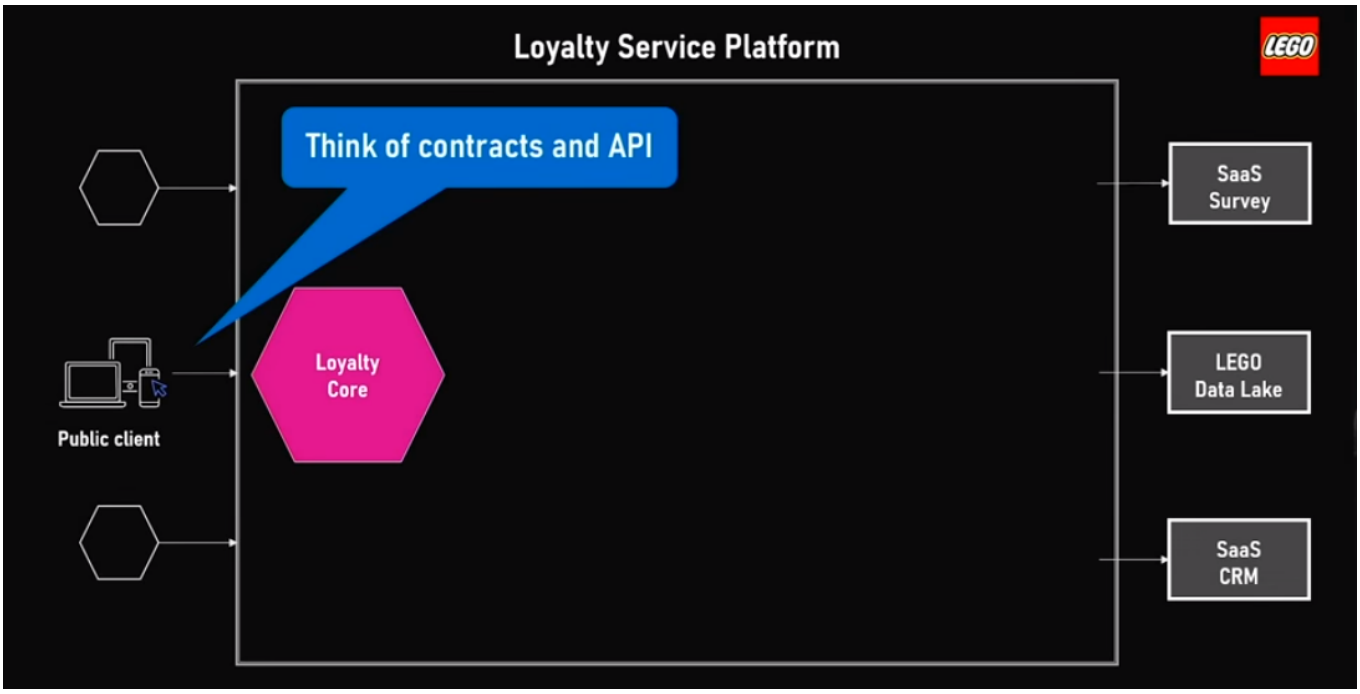


Loyalty System Context Diagram

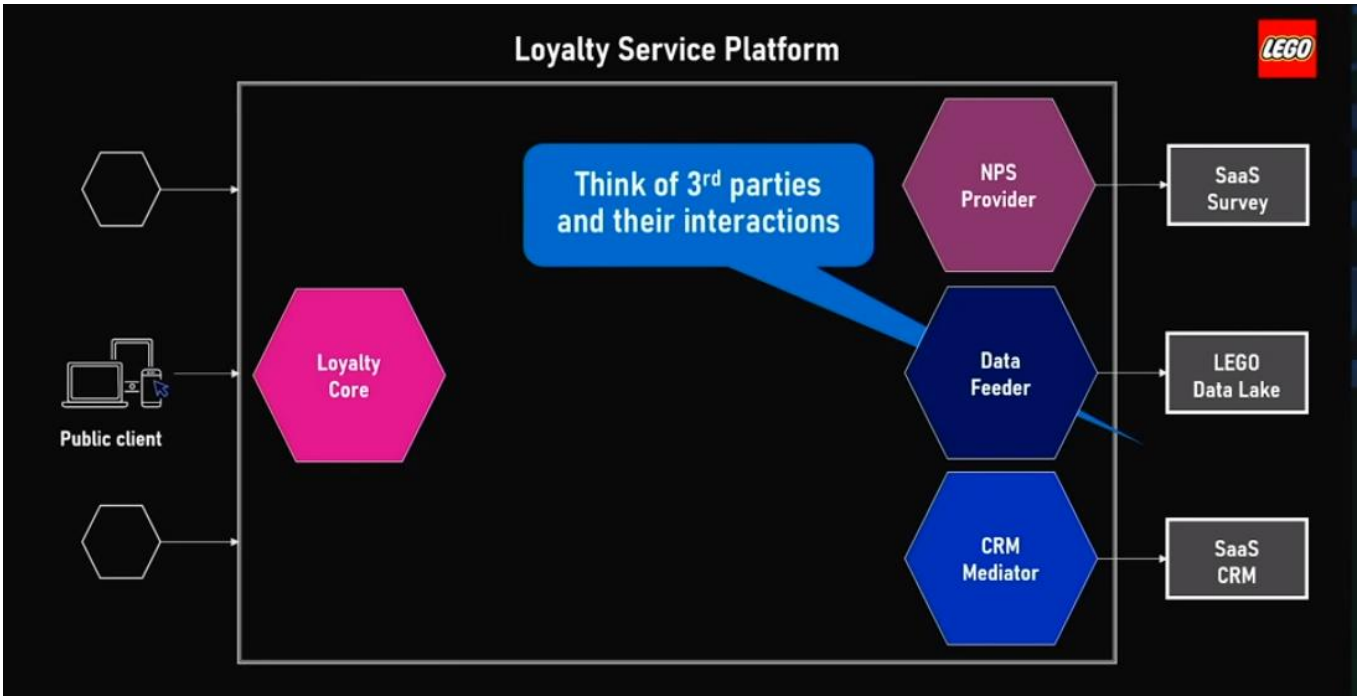
Loyalty Container Diagram


Loyalty Service Platform
SaaS Survey
LEGO Data Lake
SaaS CRM
Public client

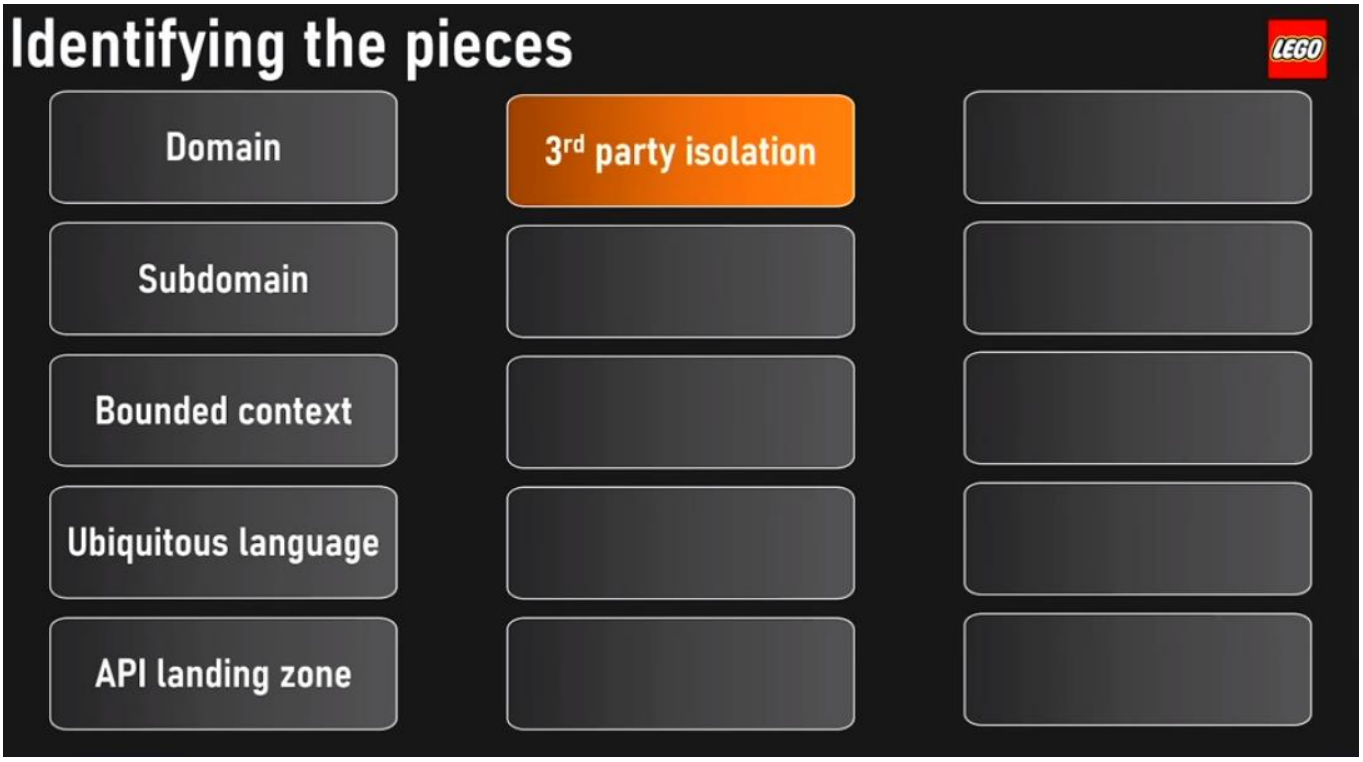

Loyalty Service Platform
Domain
Subdomain
Bounded context
Ubiquitous language
SaaS Survey
LEGO Data Lake
SaaS CRM
Public client

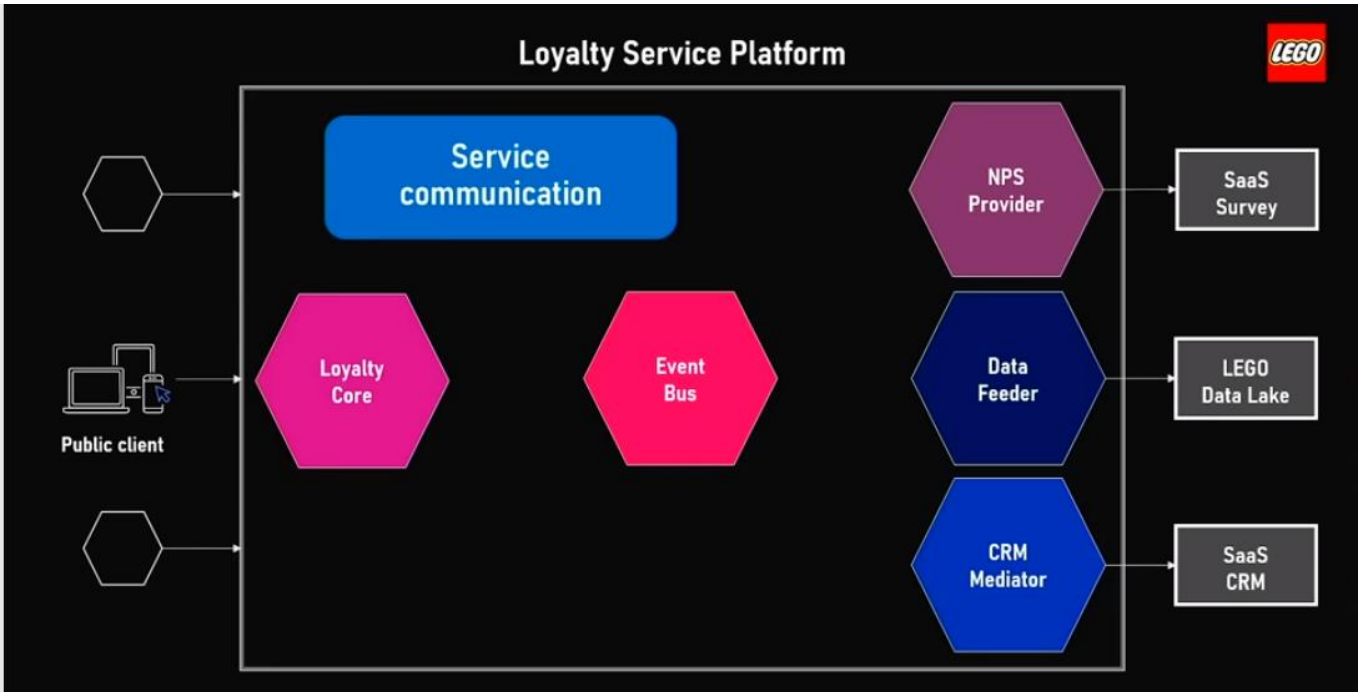


We can create a Loyalty Core microservice where all the requests land on entry, this service does just essential things.

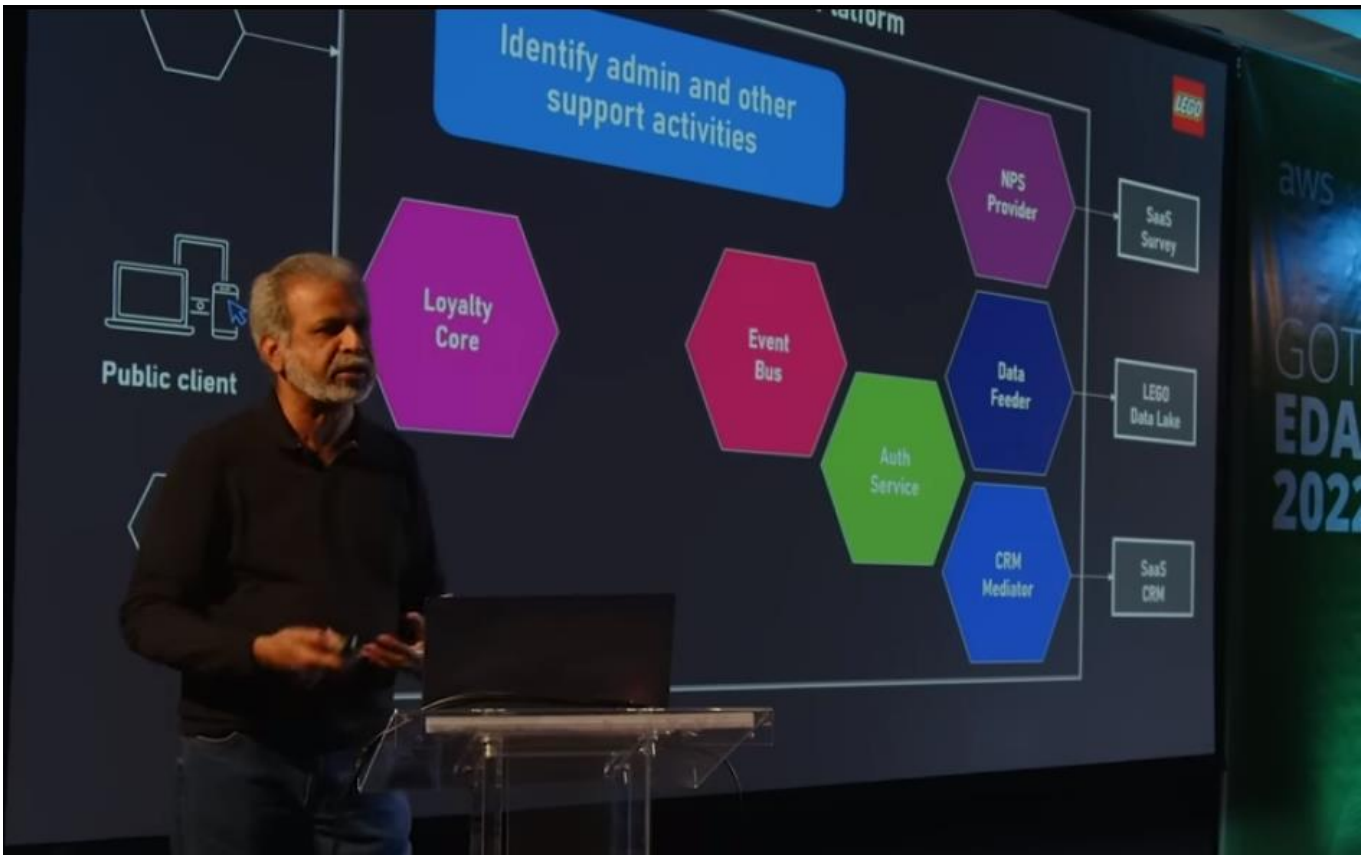


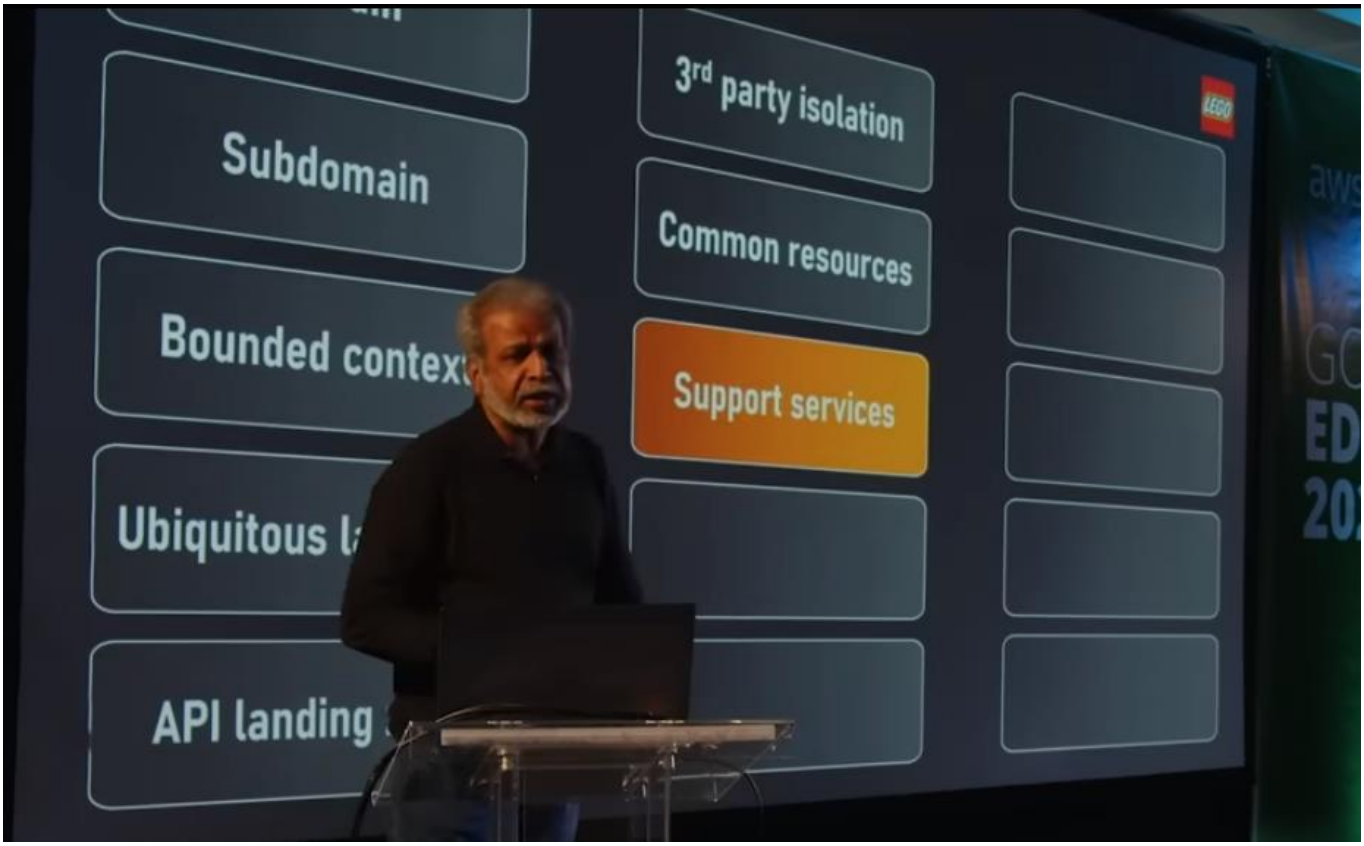
All isolate interactions with your 3rd-parties into one or few services because you might change 3rd-parties in future.

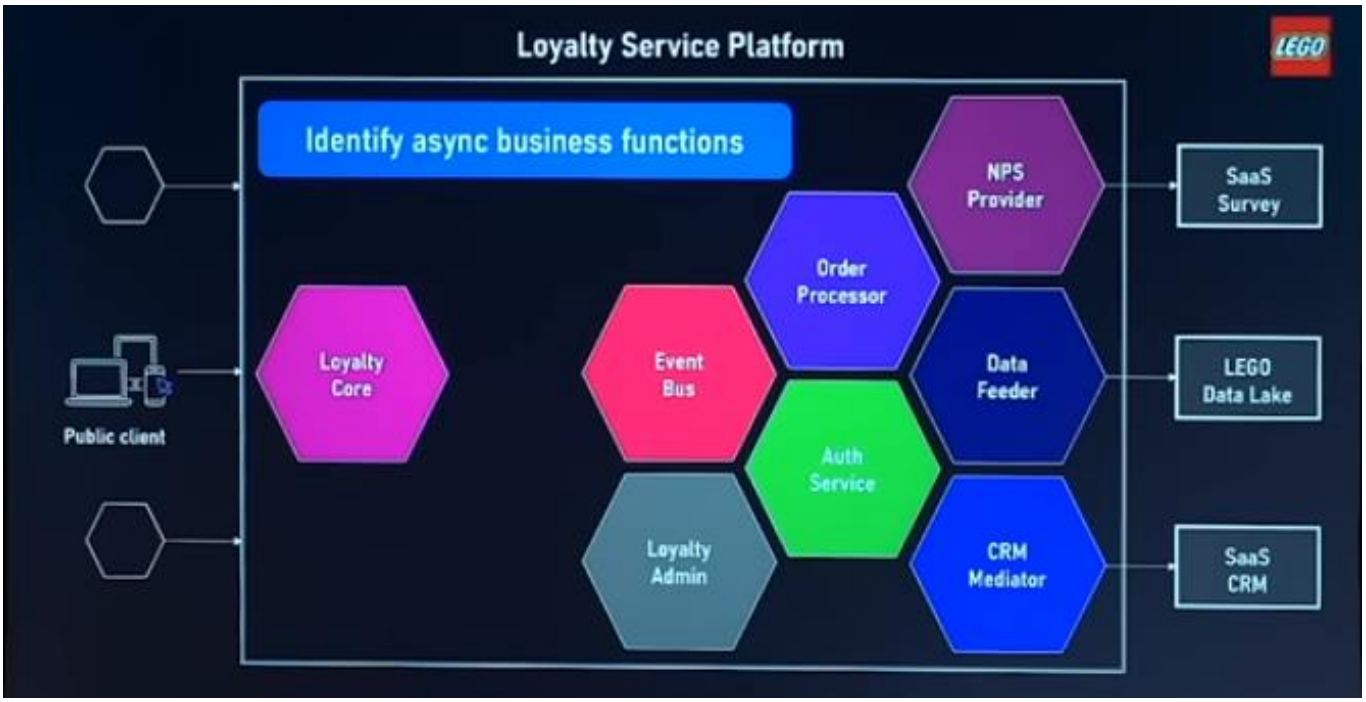



Loyalty Service Platform
Service communication
Loyalty Core
Event Bus
NPS Provider
Data Feeder
CRM Mediator
SaaS Survey
LEGO Data Lake
SaaS CRM
Public client

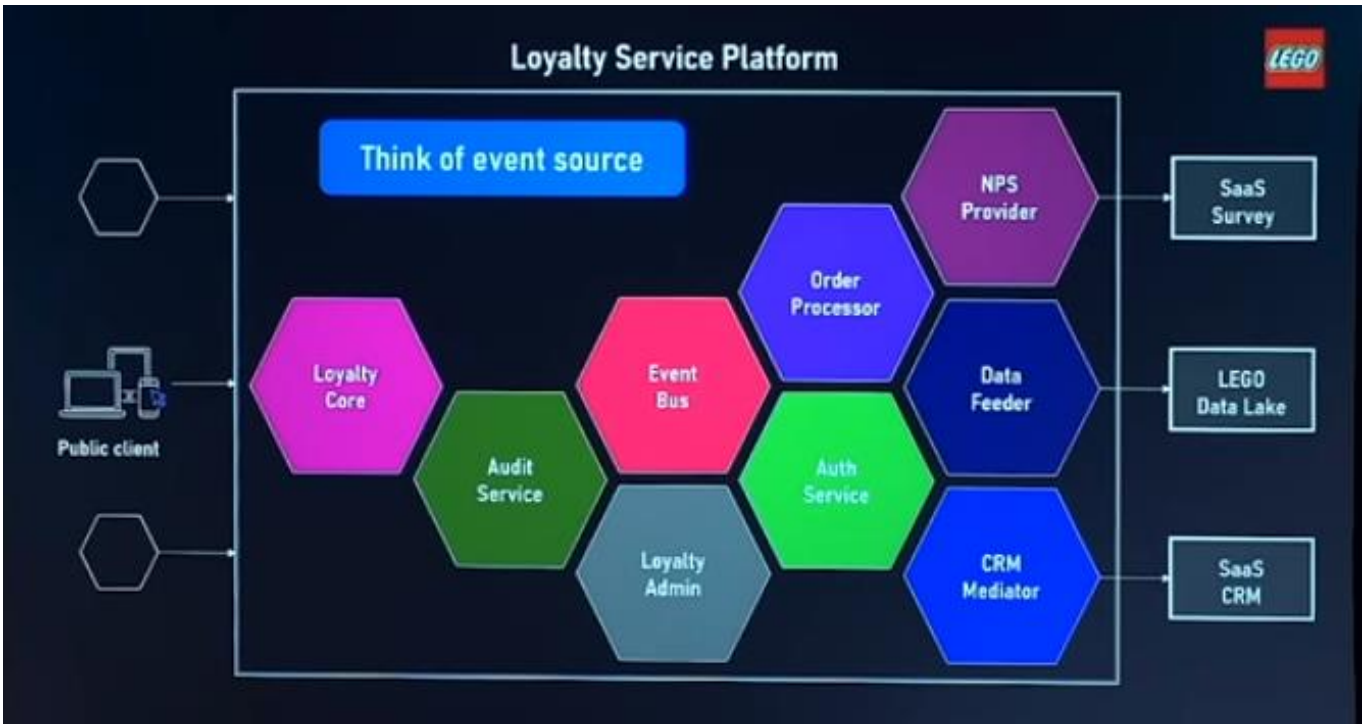

Subdomain
3rd party isolation
Common resources
aws

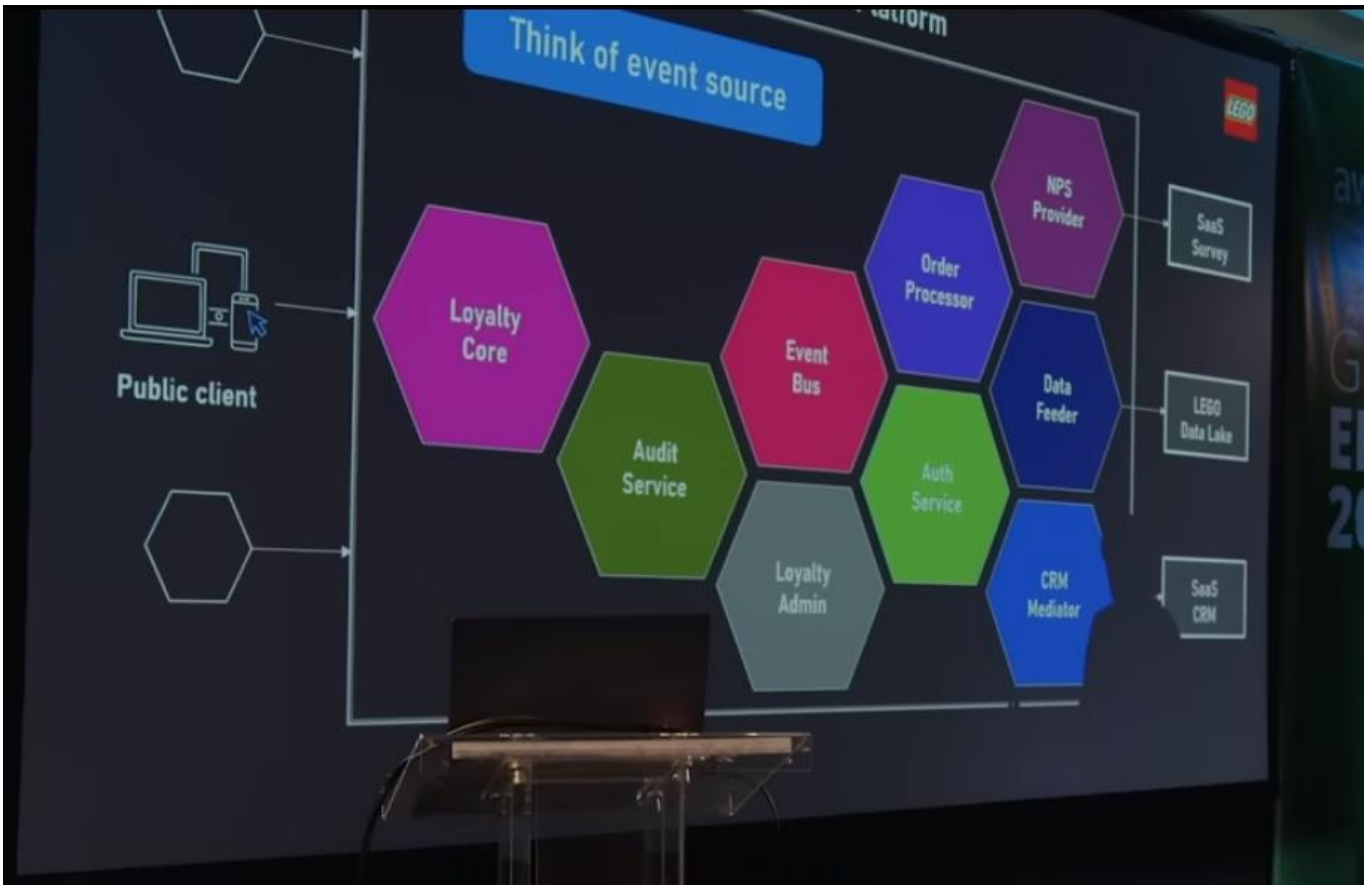

Identify admin and other support activities
Public client
Loyalty Core
Event Bus
Auth Service
NPS Provider
Data Feeder
CRM Mediator
SaaS Survey
LEGO Data Lake
SaaS CRM
LEGO
aws


Subdomain
Bounded contex
Ubiquitous l
API landing
3rd party isolation
Common resources
Support services
LEGO
aws

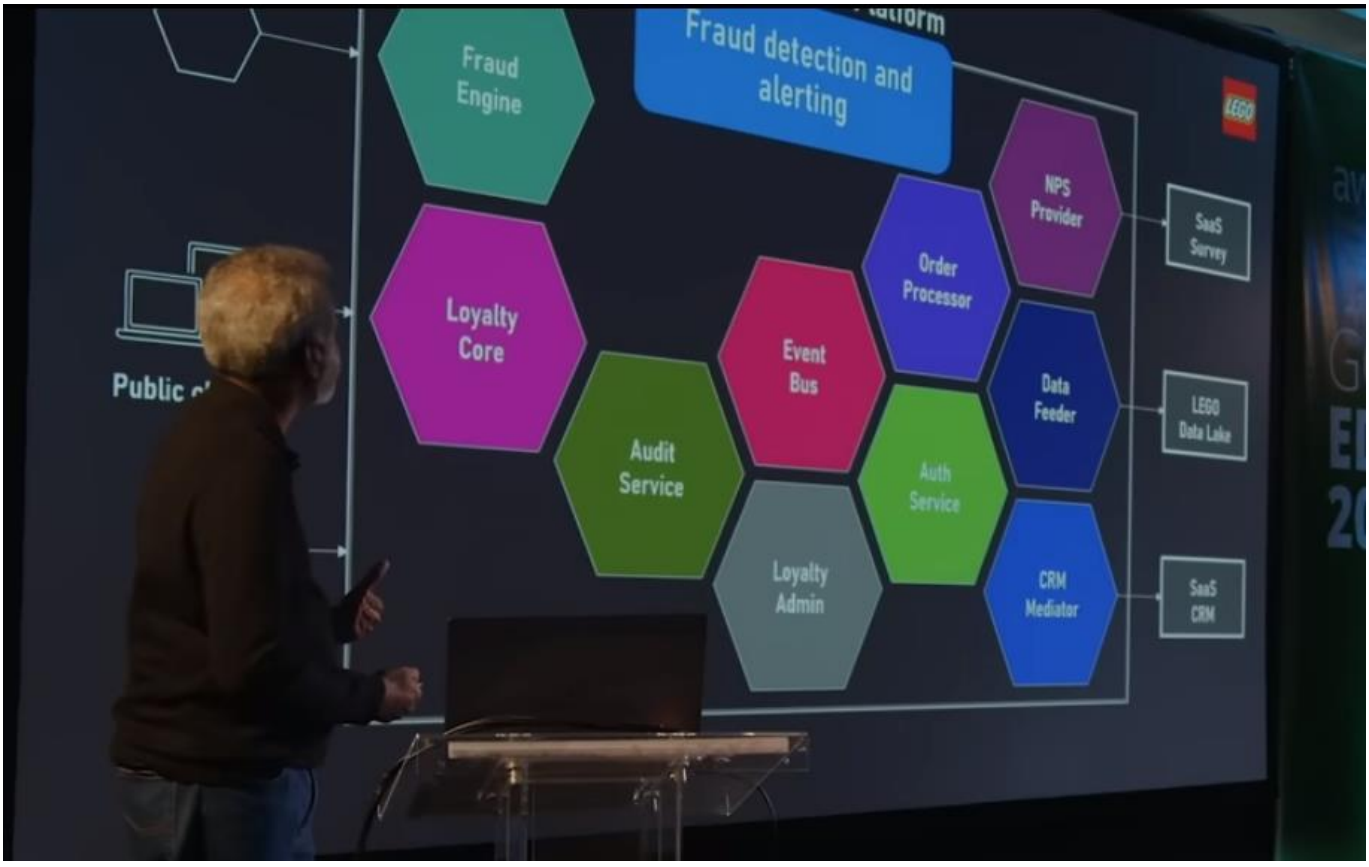

Loyalty Service Platform
Identify async business functions
NPS Provider
SaaS Survey
Order Processor
Loyalty Core
Event Bus
Data Feeder
LEGO Data Lake
Public client
Auth Service
Loyalty Admin
CRM Mediator
SaaS CRM

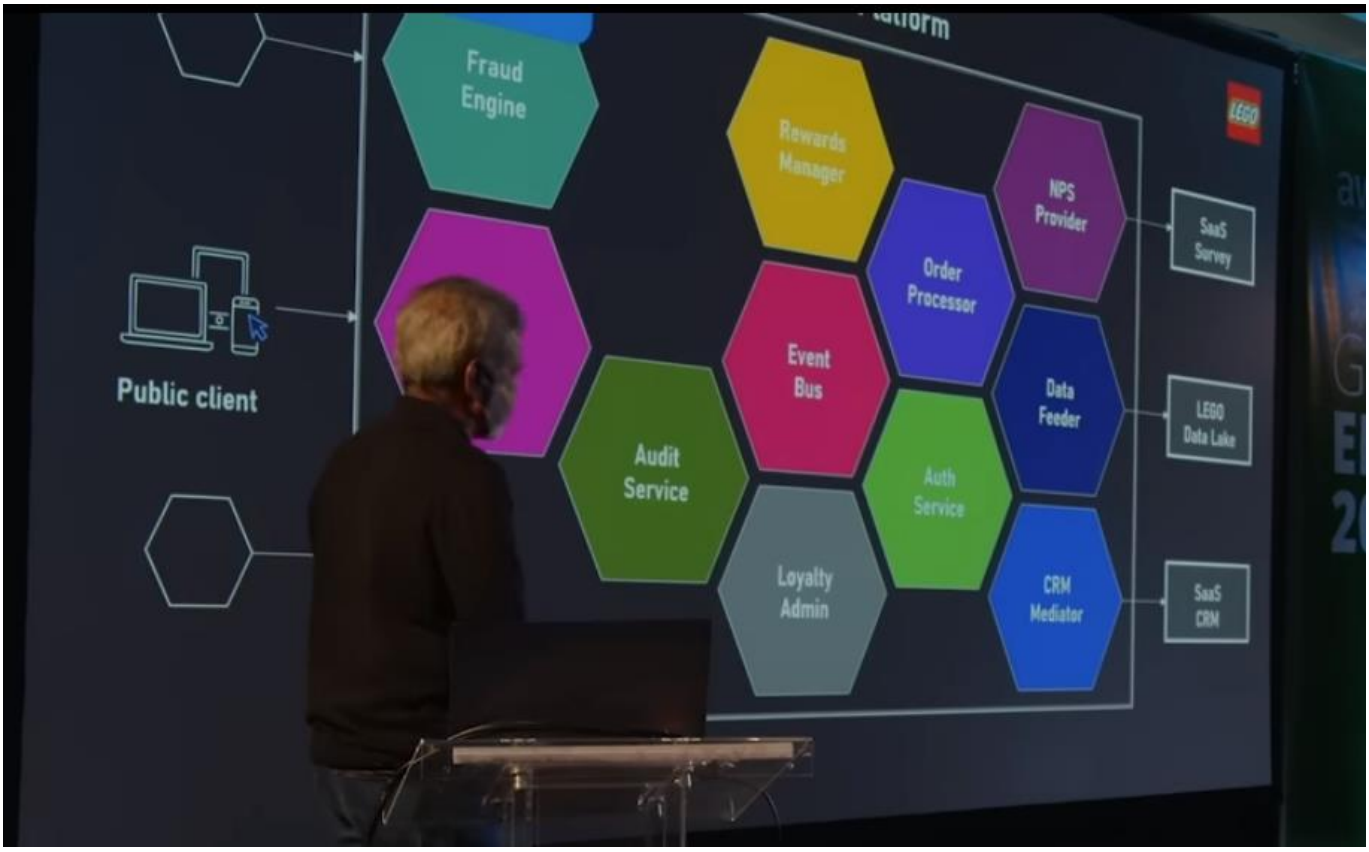

Identifying the pieces
Domain
3rd party isolation
Subdomain
Common resources
Bounded context
Support services
Ubiquitous language
Async activities
API landing zone

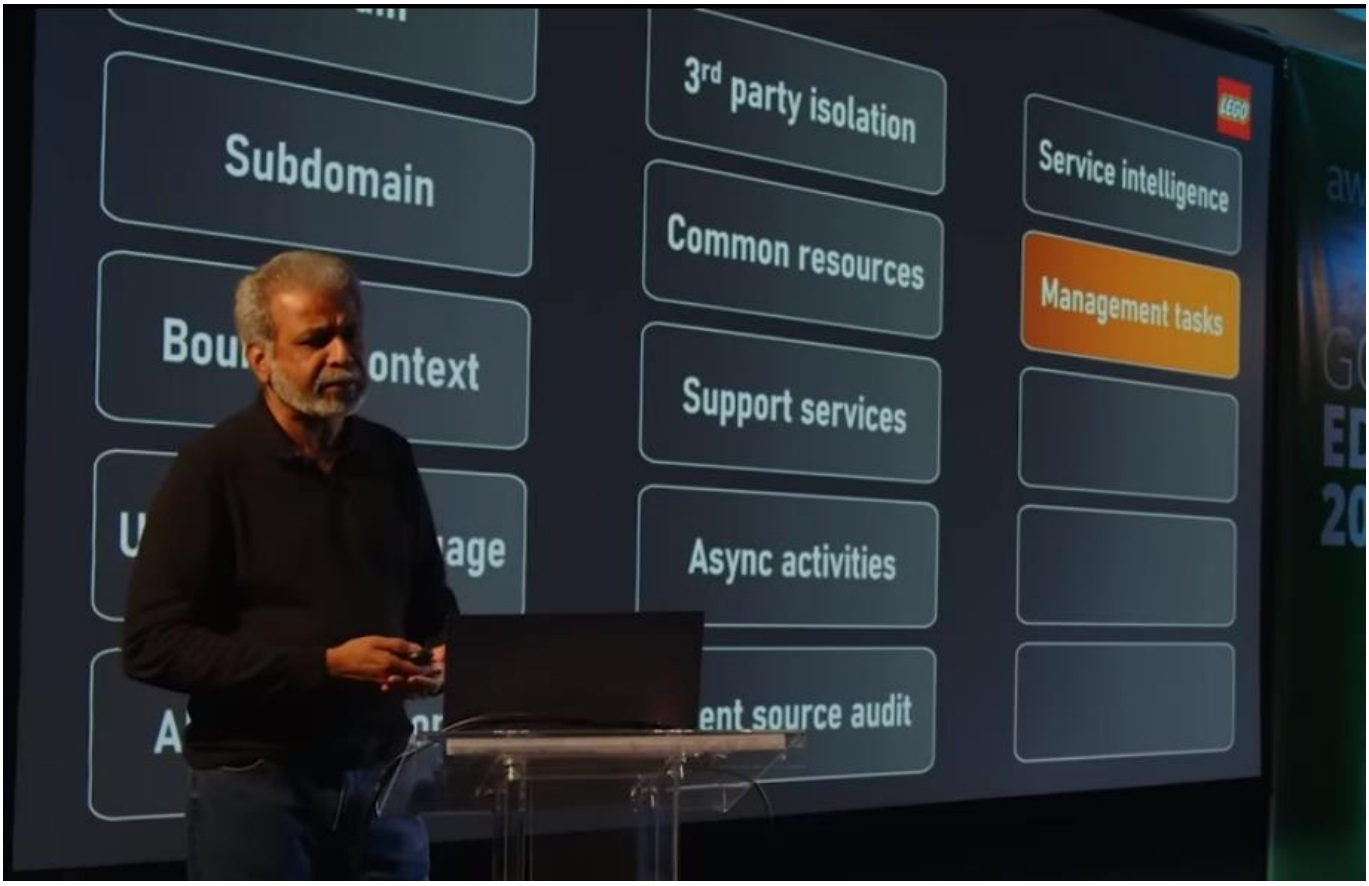

Loyalty Service Platform
Think of event source
Public client
Loyalty Core
Audit Service
Event Bus
Order Processor
NPS Provider
Auth Service
Loyalty Admin
Data Feeder
CRM Mediator
SaaS Survey
LEGO Data Lake
SaaS CRM

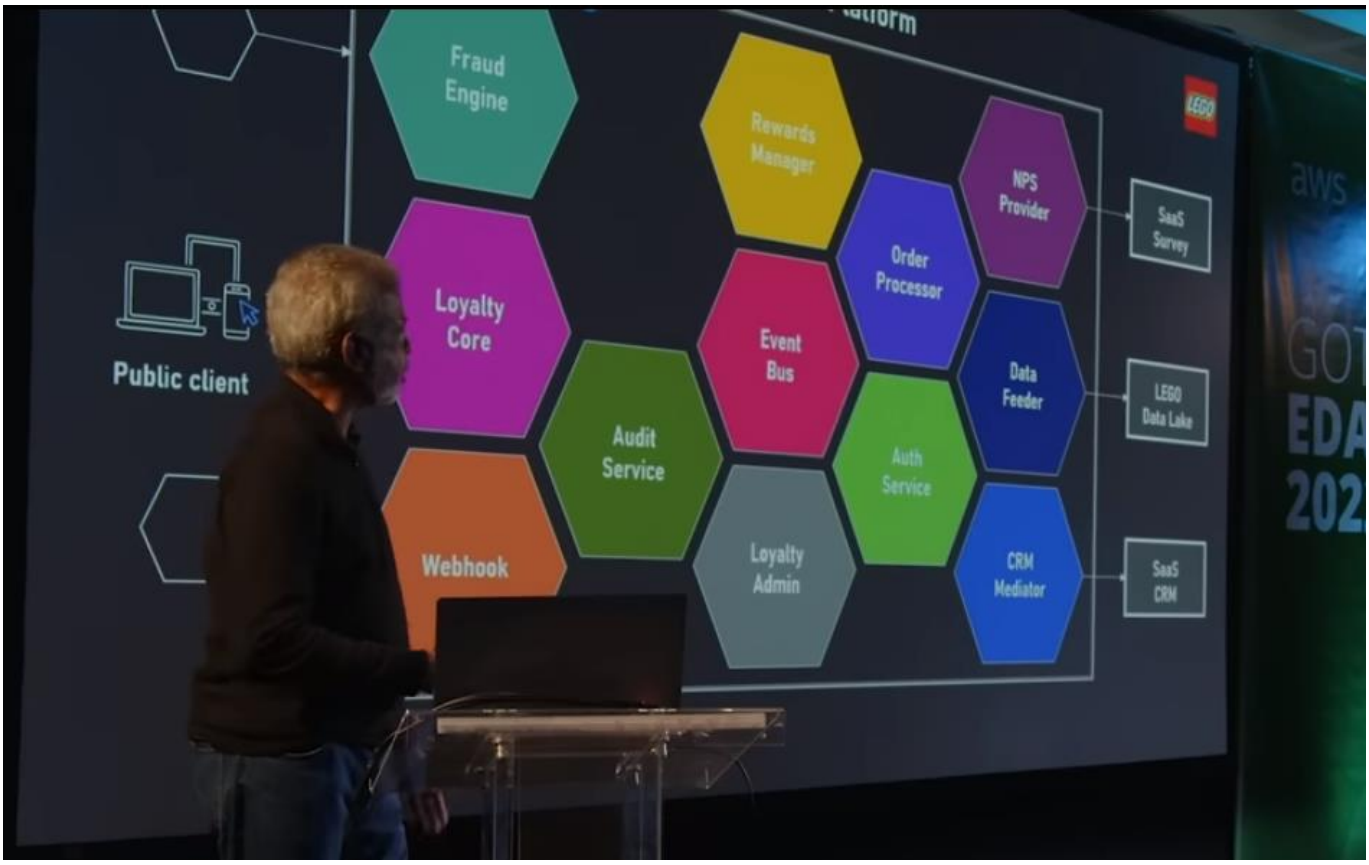

Think of event source
Public client
Loyalty Core
Audit Service
Event Bus
Order Processor
NPS Provider
Auth Service
Loyalty Admin
Data Feeder
CRM Mediator
SaaS Survey
LEGO Data Lake
SaaS CRM


Subdomain
3rd party isolation
Common resources
Support services
Async activities


Fraud Engine
Fraud detection and alerting
Loyalty Core
Event Bus
Order Processor
NPS Provider
SaaS Survey
Data Feeder
LEGO Data Lake
Audit Service
Auth Service
Loyalty Admin
CRM Mediator
SaaS CRM


Fraud Engine
Rewards Manager
NPS Provider
SaaS Survey
Order Processor
Public client
Event Bus
Data Feeder
LEGO Data Lake
Audit Service
Auth Service
Loyalty Admin
CRM Mediator
SaaS CRM

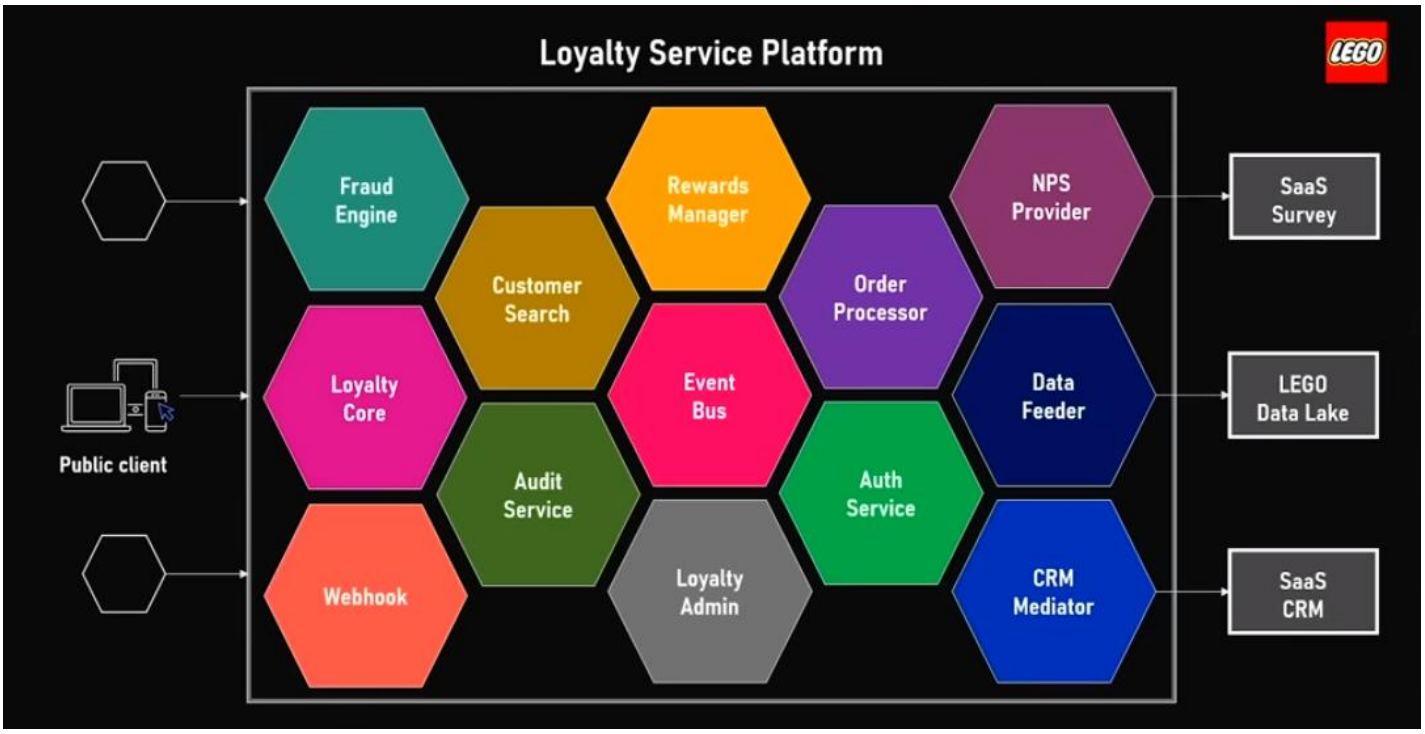

Subdomain
3rd party isolation
Service intelligence
Common resources
Management tasks
Support services
Async activities


Fraud Engine
Rewards Manager
NPS Provider
SaaS Survey
Loyalty Core
Order Processor
Event Bus
Data Feeder
LEGO Data Lake
Public client
Audit Service
Auth Service
Webhook
Loyalty Admin
CRM Mediator
SaaS CRM

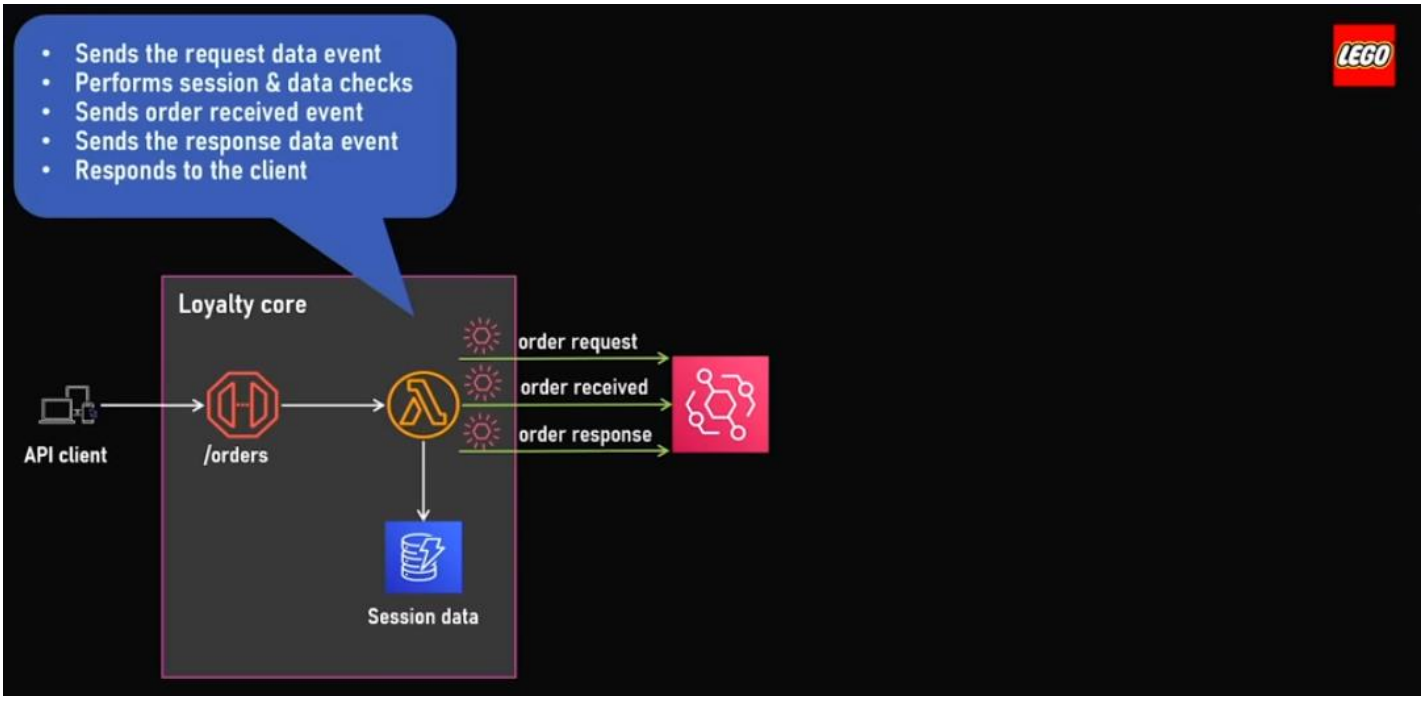

Subdomain
3rd party isolation
Service intelligence
Common resources
Management tasks
Support services
Notifications
Async activities

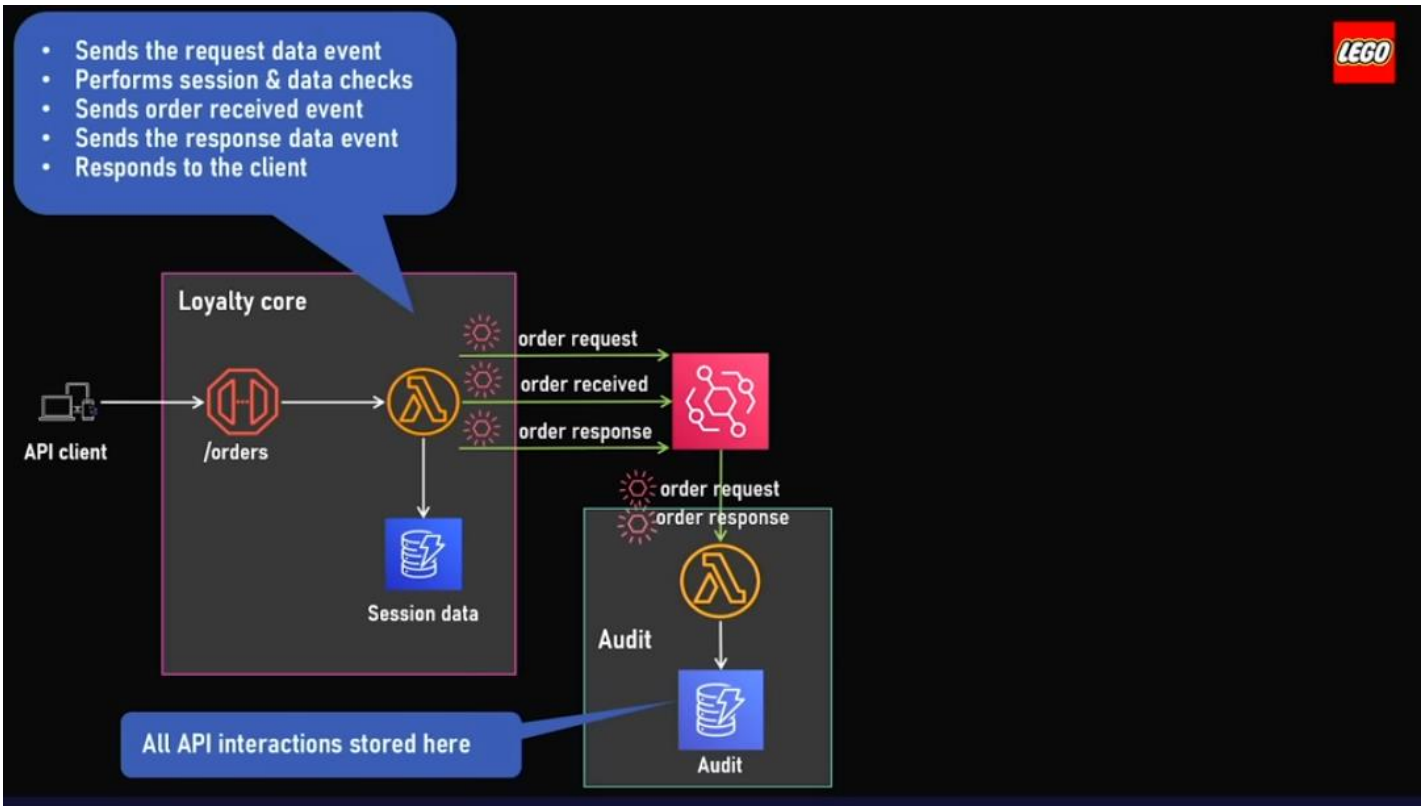
Identifying the pieces

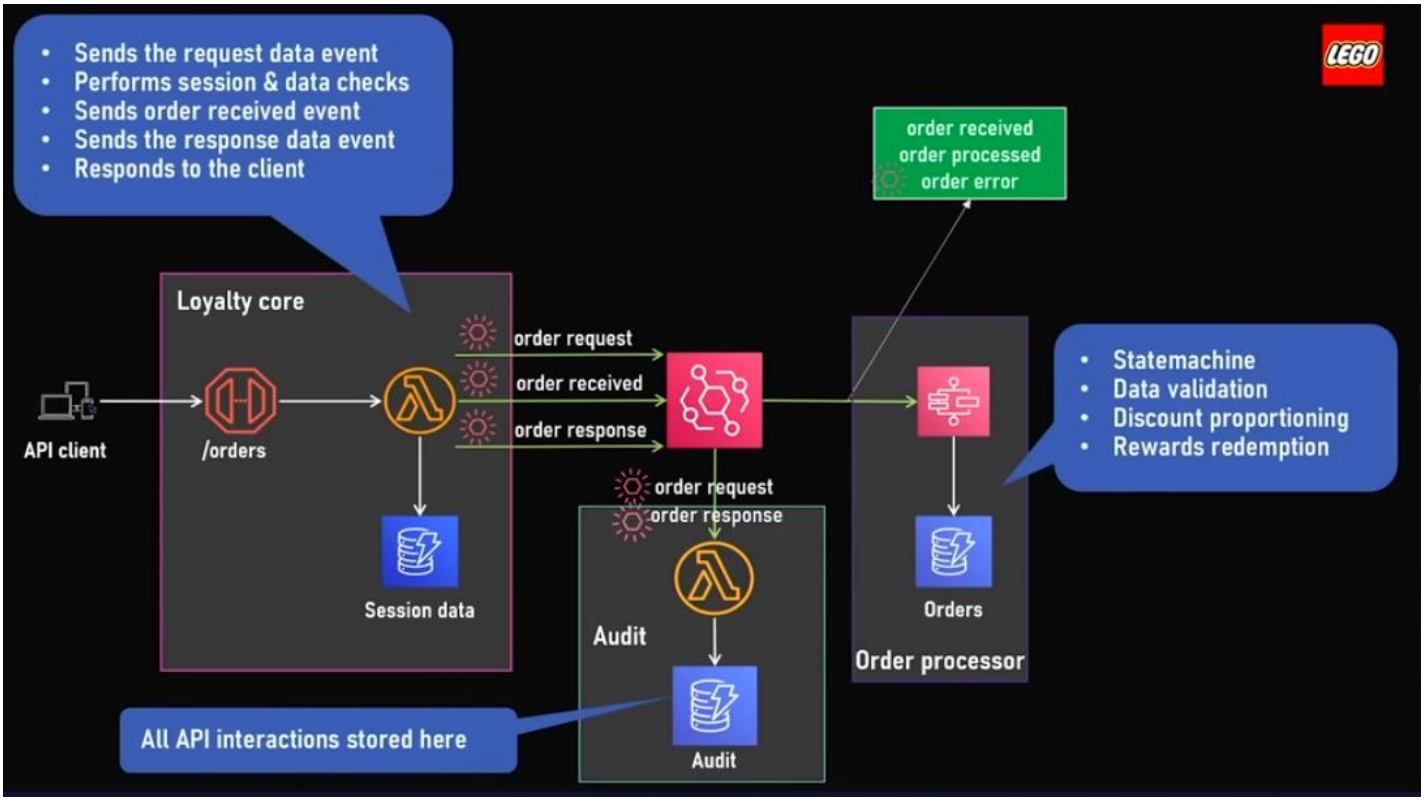
Domain	3rd party isolation	Service intelligence
Subdomain	Common resources	Management tasks
Bounded context	Support services	Notifications
Ubiquitous language	Async activities	New features
API landing zone	Event source audit	

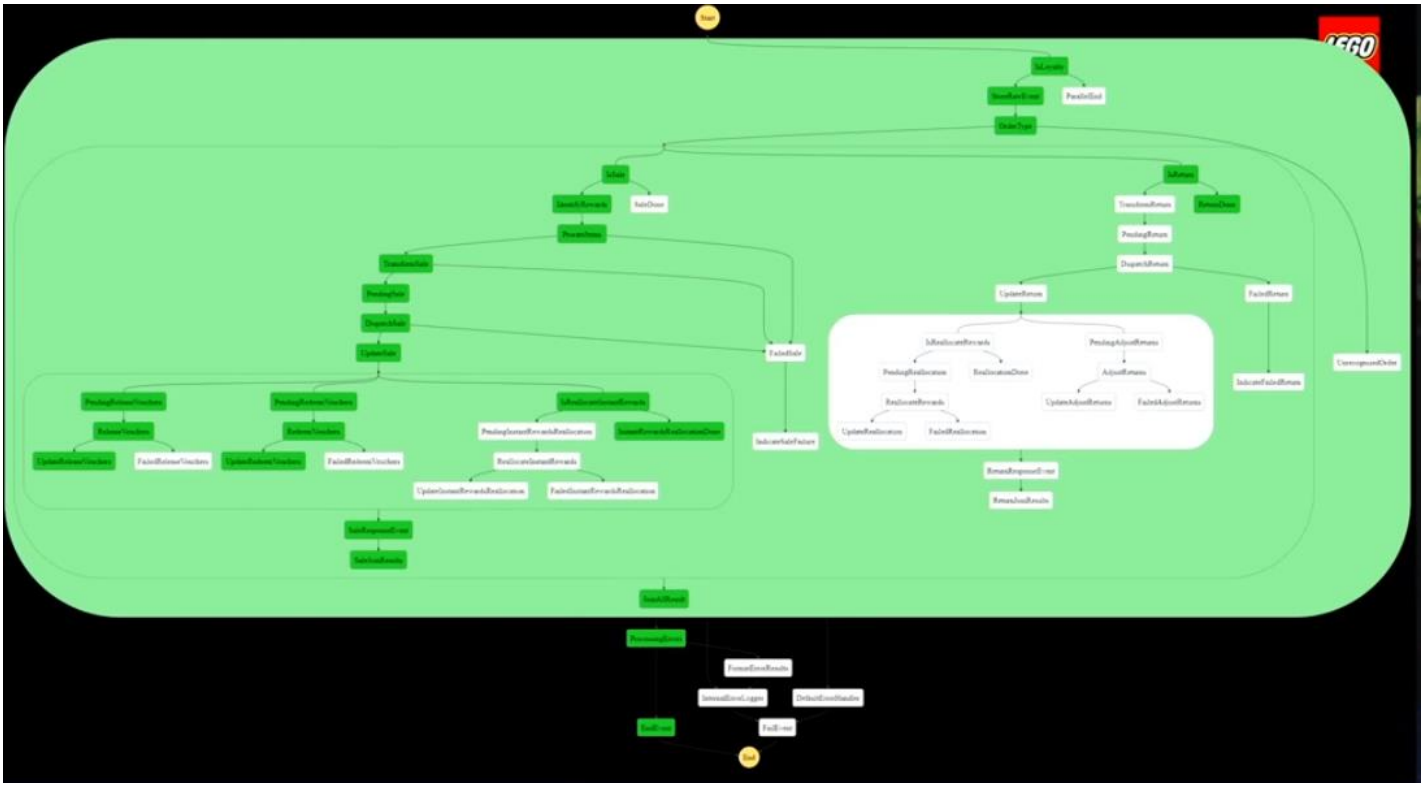


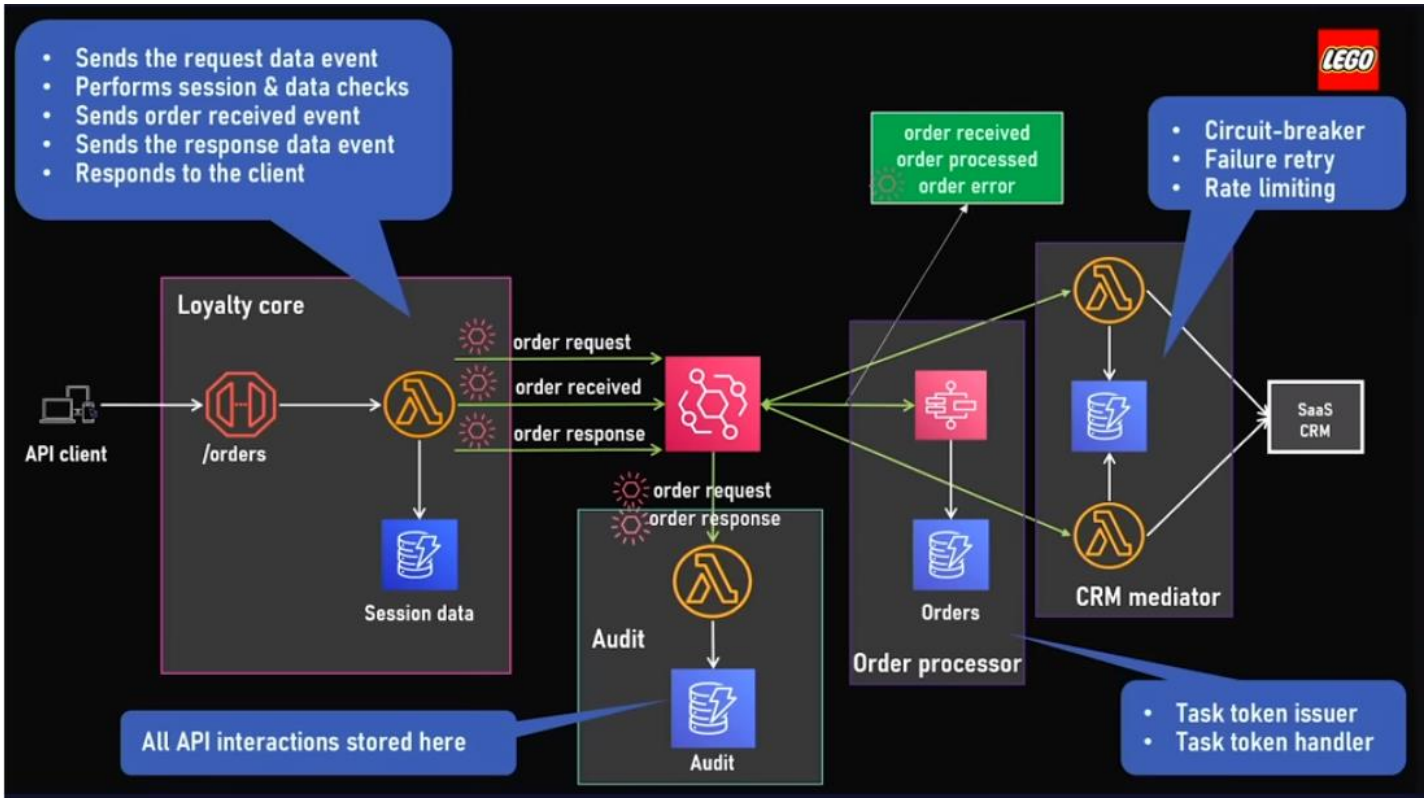
Handling a loyalty order

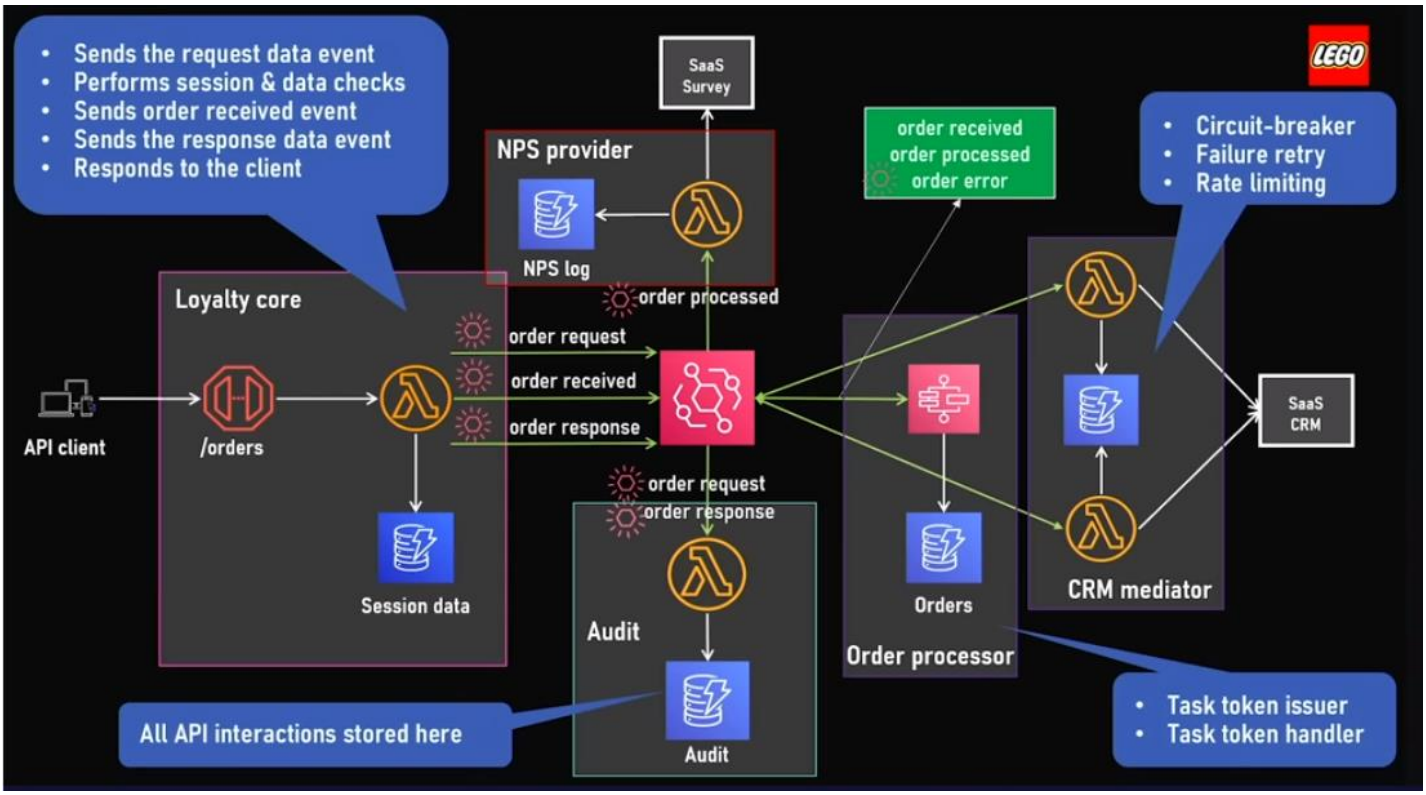



Sends the request data event
Performs session & data checks
Sends order received event
Sends the response data event
Responds to the client
Loyalty core
order request
order received
order response
API client
/orders
order request
order response
Session data
Audit
All API interactions stored here
Audit

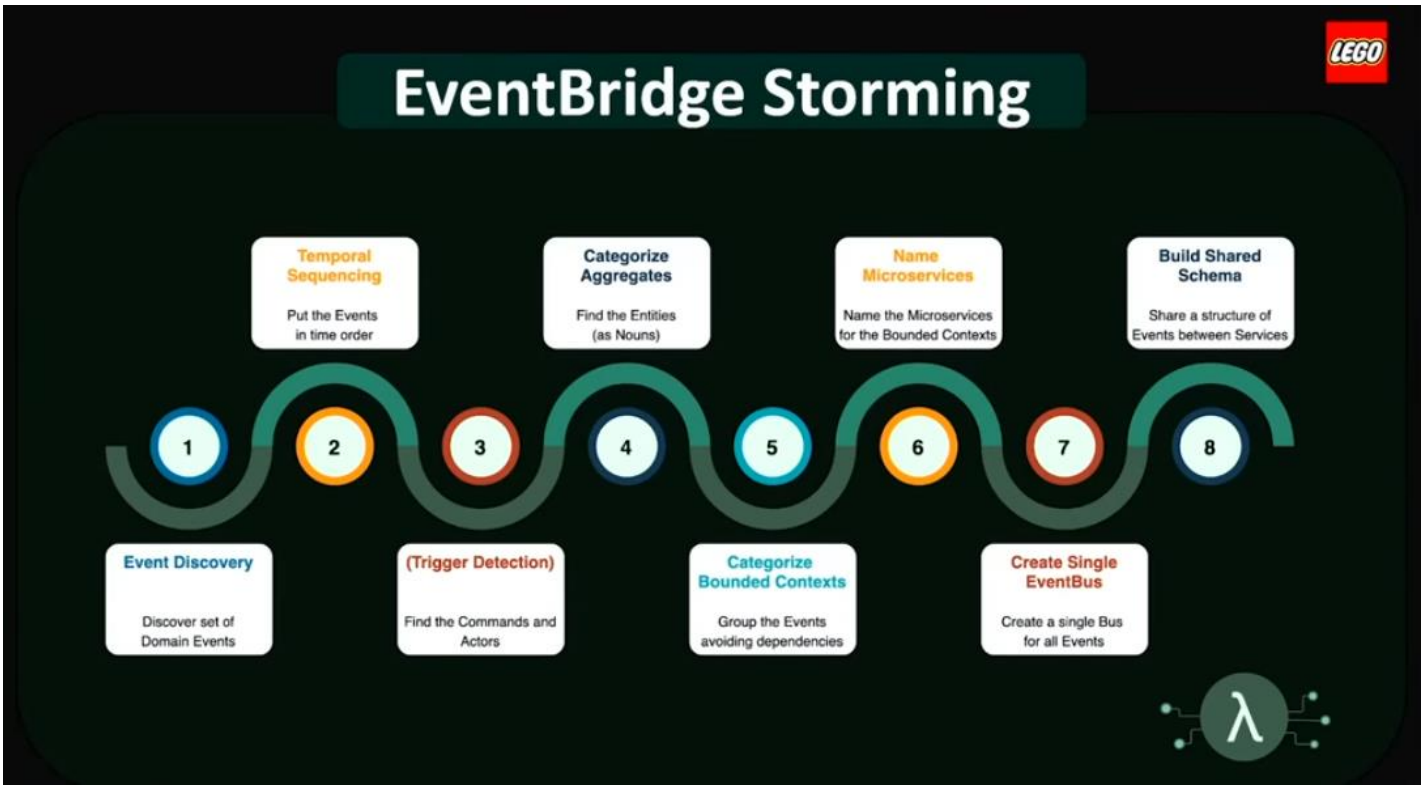

Sends the request data event
Performs session & data checks
Sends order received event
Sends the response data event
Responds to the client
order received
order processed
order error
Loyalty core
order request
order received
order response
API client
/orders
Statemachine
Data validation
Discount proportioning
Rewards redemption
order request
order response
Session data
Audit
Orders
Order processor
All API interactions stored here
Audit



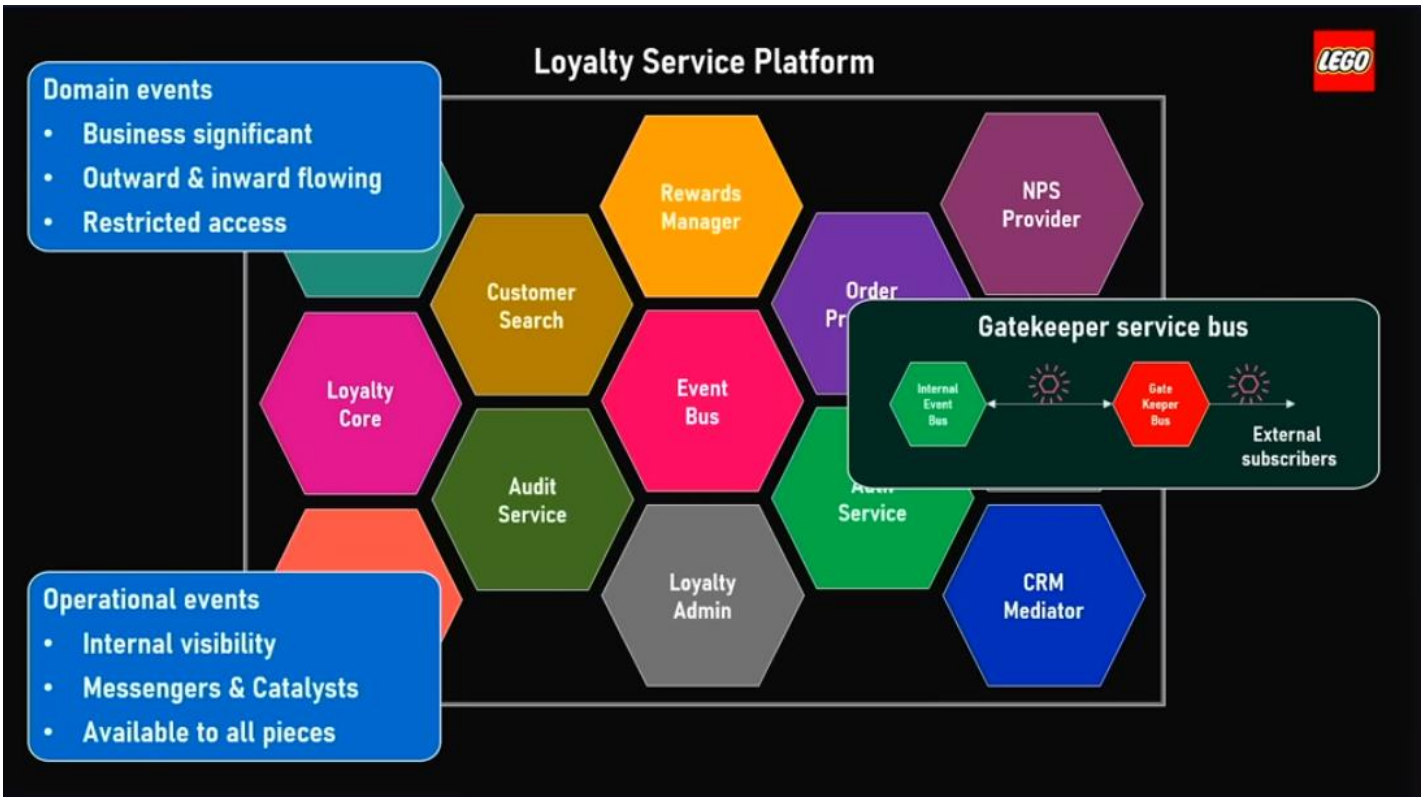

Sends the request data event
Performs session & data checks
Sends order received event
Sends the response data event
Responds to the client
Loyalty core
API client
/orders
Session data
order request
order received
order response
order request
order response
Audit
Audit
All API interactions stored here
order received
order processed
order error
Orders
Order processor
Circuit-breaker
Failure retry
Rate limiting
CRM mediator
SaaS
CRM
Task token issuer
Task token handler



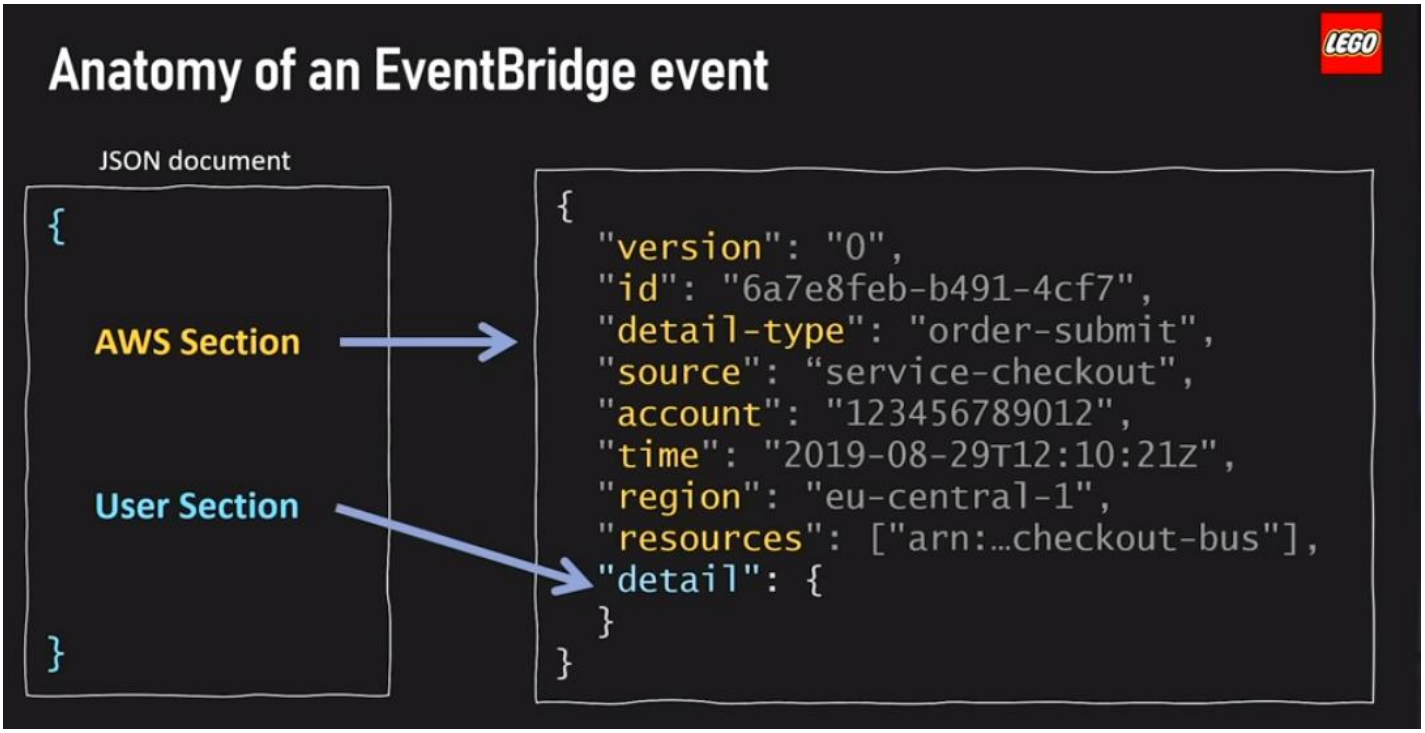
How do you find the events?

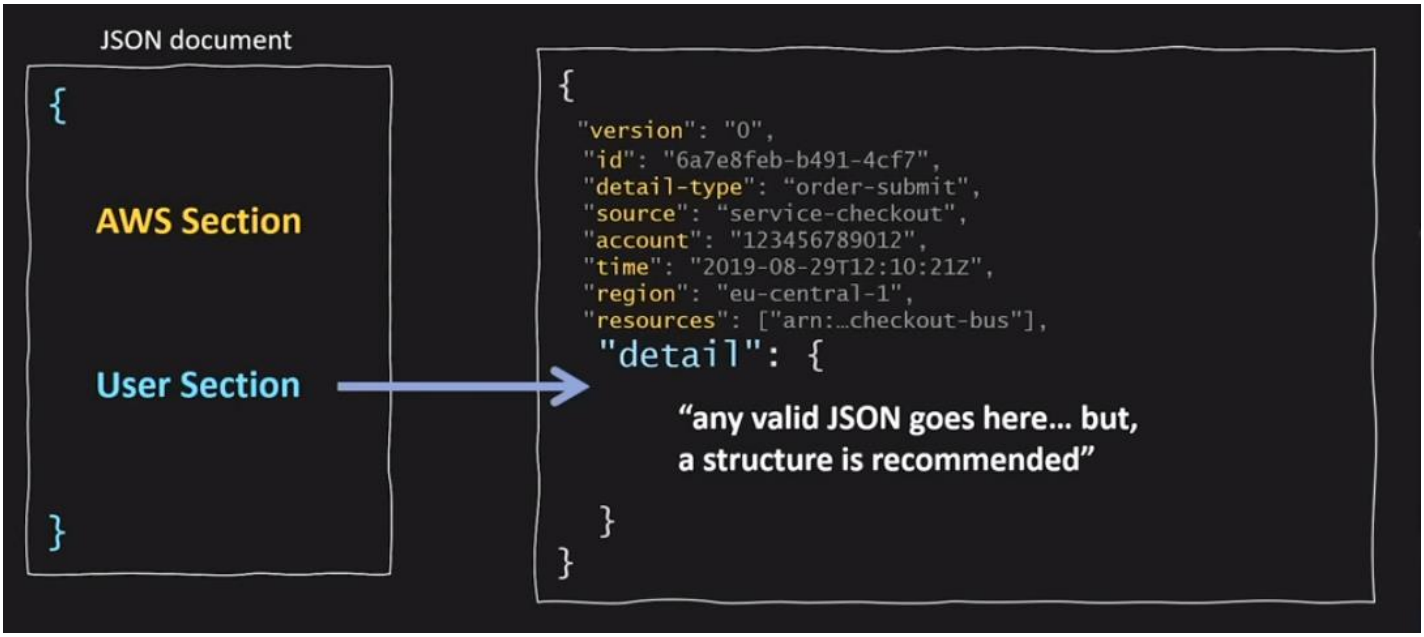

EventBridge Storming
1
Event Discovery
Discover set of Domain Events
2
Temporal Sequencing
Put the Events in time order
3
(Trigger Detection)
Find the Commands and Actors
4
Categorize Aggregates
Find the Entities (as Nouns)
5
Categorize Bounded Contexts
Group the Events avoiding dependencies
6
Name Microservices
Name the Microservices for the Bounded Contexts
7
Create Single EventBus
Create a single Bus for all Events
8
Build Shared Schema
Share a structure of Events between Services
λ

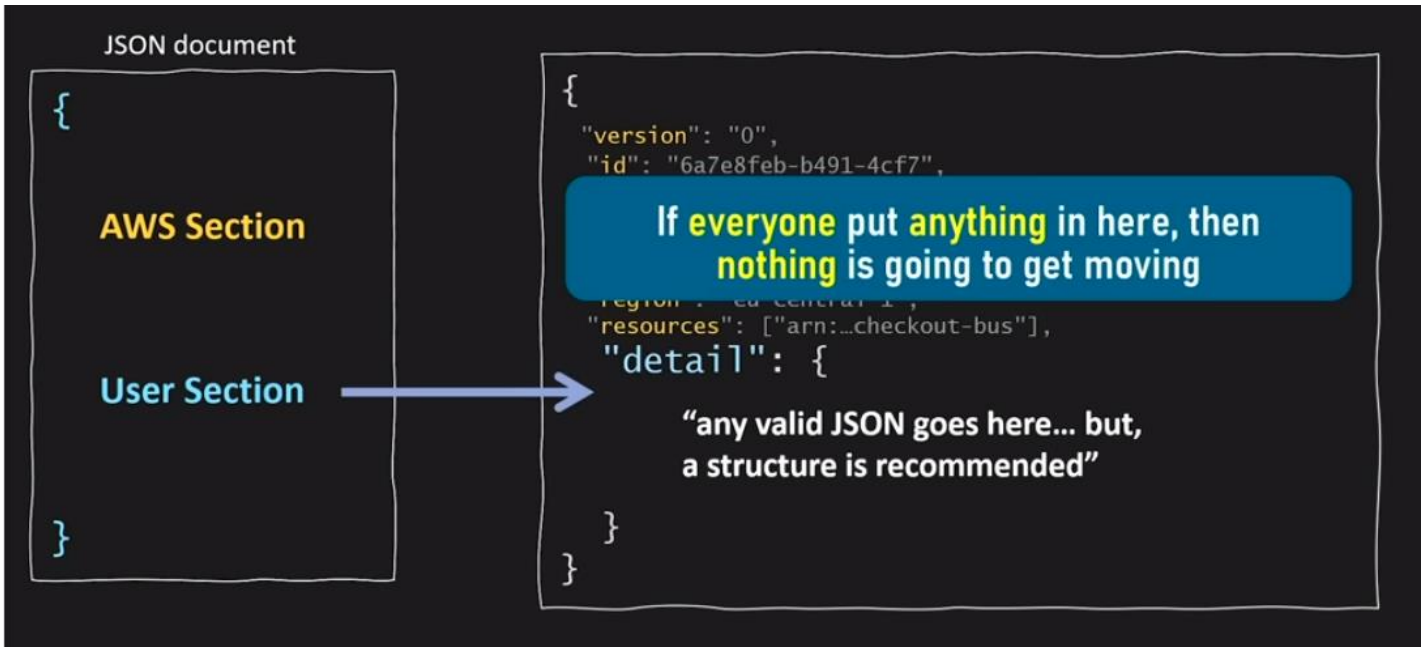

Loyalty Service Platform
Fraud Engine
Rewards Manager
NPS Provider
Customer Search
Order Processor
Loyalty Core
Event Bus
Data Feeder
Audit Service
Auth Service
Webhook
Loyalty Admin
CRM Mediator

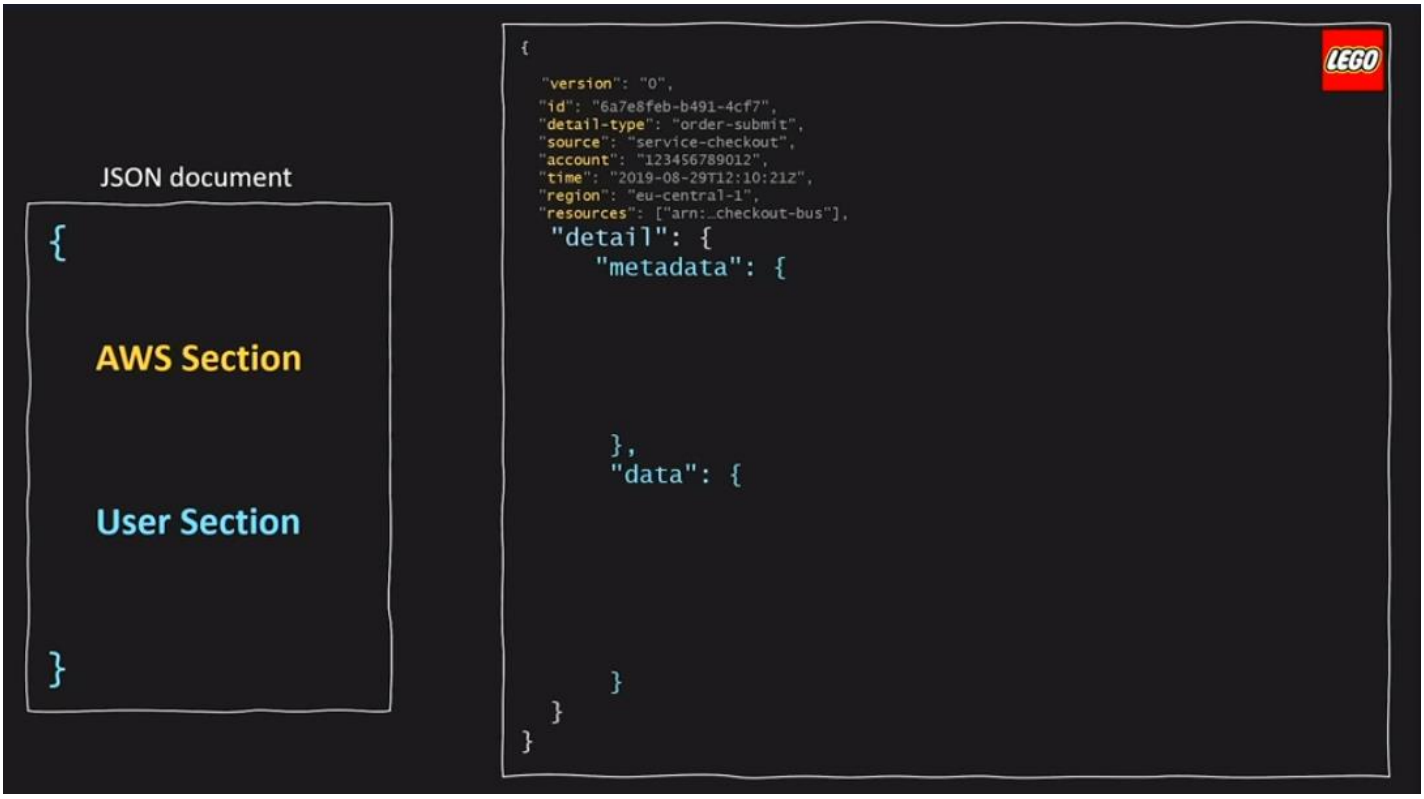


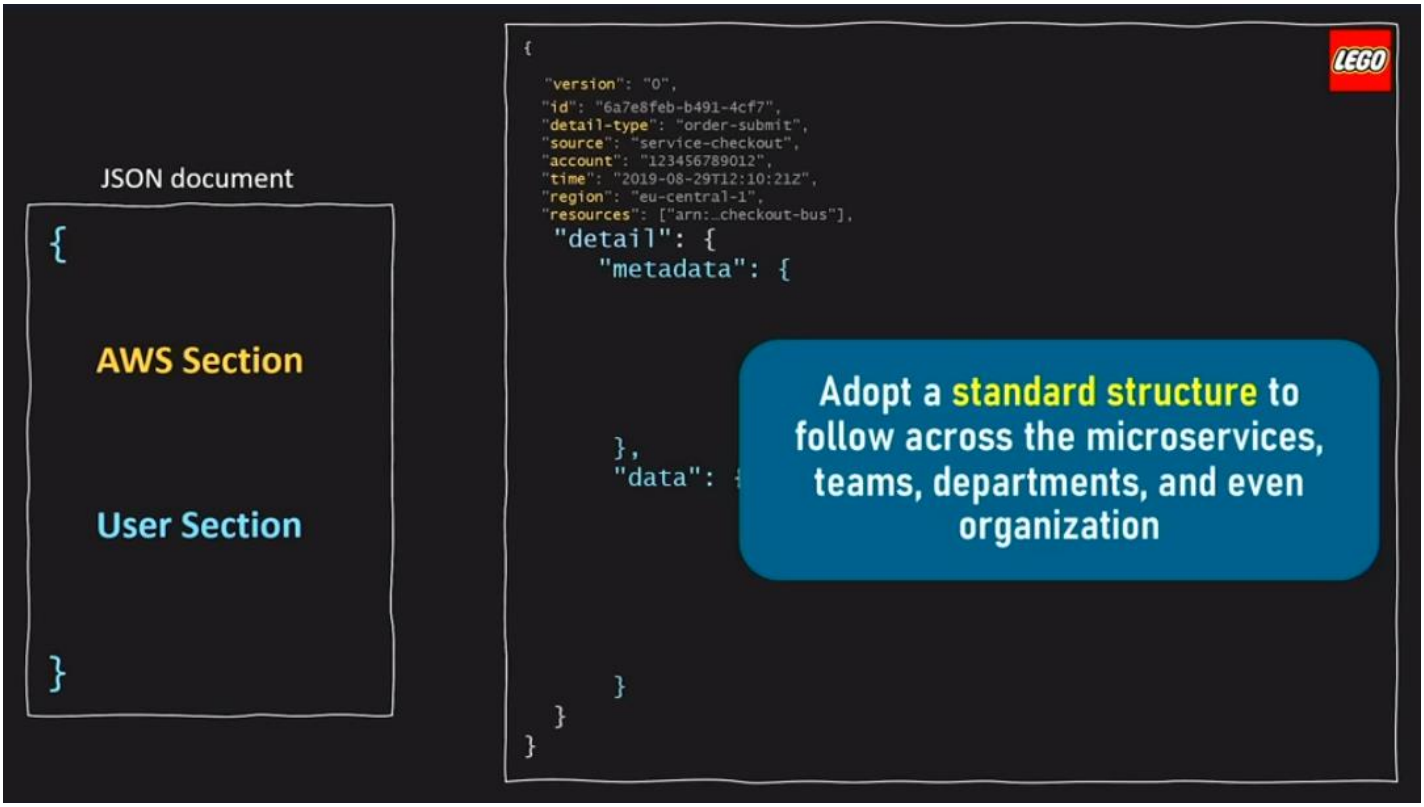
What's in an event?




JSON document
{
AWS Section
User Section
}
{
"version": "0",
"id": "6a7e8feb-b491-4cf7",
"detail-type": "order-submit",
"source": "service-checkout",
"account": "123456789012",
"time": "2019-08-29T12:10:21Z",
"region": "eu-central-1",
"resources": ["arn:…checkout-bus"],
"detail": {
"any valid JSON goes here... but,
a structure is recommended"
}
}


JSON document
{
AWS Section
User Section
}
{
"version": "0",
"id": "6a7e8feb-b491-4cf7",
If everyone put anything in here, then
nothing is going to get moving
"resources": ["arn:…checkout-bus"],
"detail": {
"any valid JSON goes here... but,
a structure is recommended"
}
}


JSON document
{
AWS Section
User Section
}
{
"version": "0",
"id": "6a7e8feb-b491-4cf7",
"detail-type": "order-submit",
"source": "service-checkout",
"account": "123456789012",
"time": "2019-08-29T12:10:21Z",
"region": "eu-central-1",
"resources": ["arn:…checkout-bus"],
"detail": {
"metadata": {
},
"data": {
}
}
}


JSON document
{
AWS Section
User Section
}
{
"version": "0",
"id": "6a7e8feb-b491-4cf7",
"detail-type": "order-submit",
"source": "service-checkout",
"account": "123456789012",
"time": "2019-08-29T12:10:21Z",
"region": "eu-central-1",
"resources": ["arn:…checkout-bus"],
"detail": {
"metadata": {
},
"data": {
}
}
}
Adopt a standard structure to follow across the microservices, teams, departments, and even organization

JSON document

{

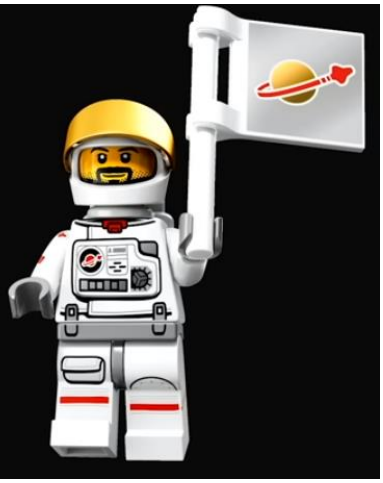
AWS Section

User Section

}

The Experience

Effect of EDA adoption



You cannot create experience,

you must undergo it

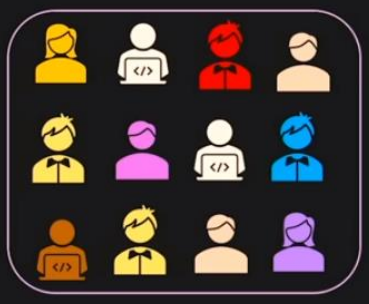
- Albert Camus

Team growth

Spring 2018

From...

- 1 team
- 10 engineers
- 1 monolith
- 8 application servers

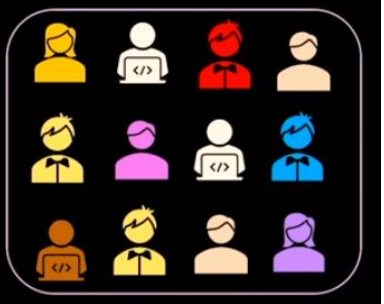


Ecommerce team

July 2019

- 2 squads
- 20 engineers
- 2 repositories
- 1 AWS prod account
- 20 microservices
- 100 lambda functions

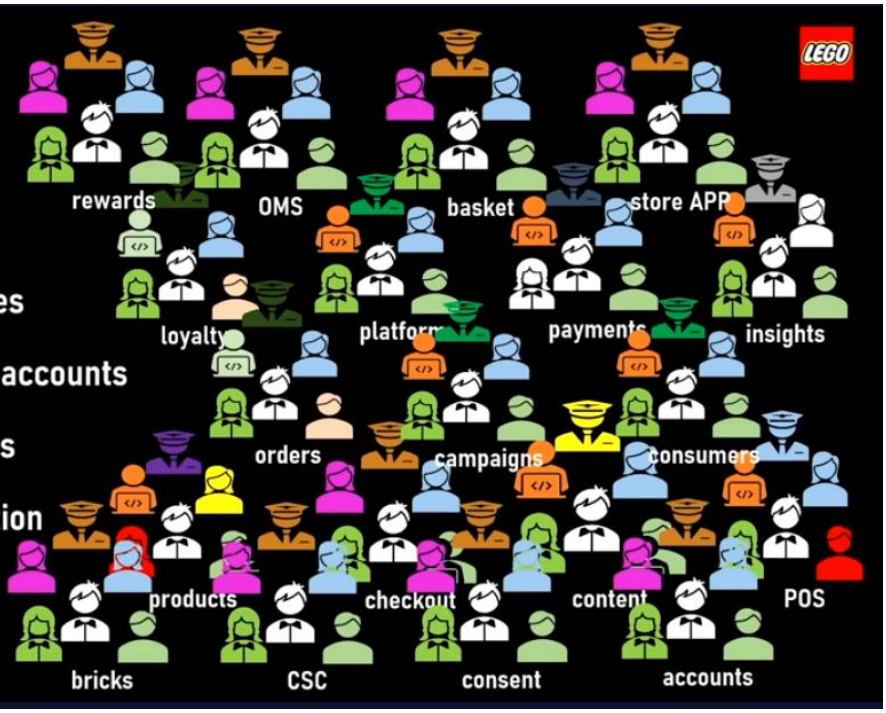
Serverless team



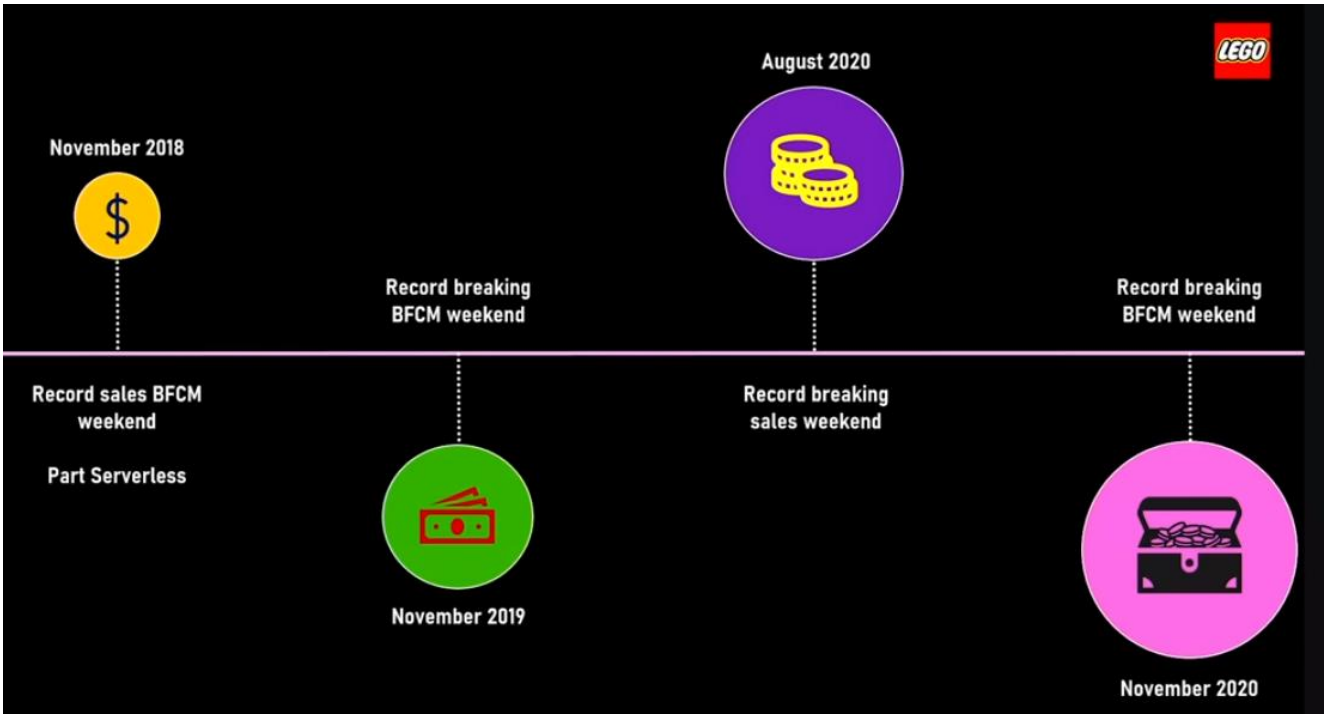
Front-end team

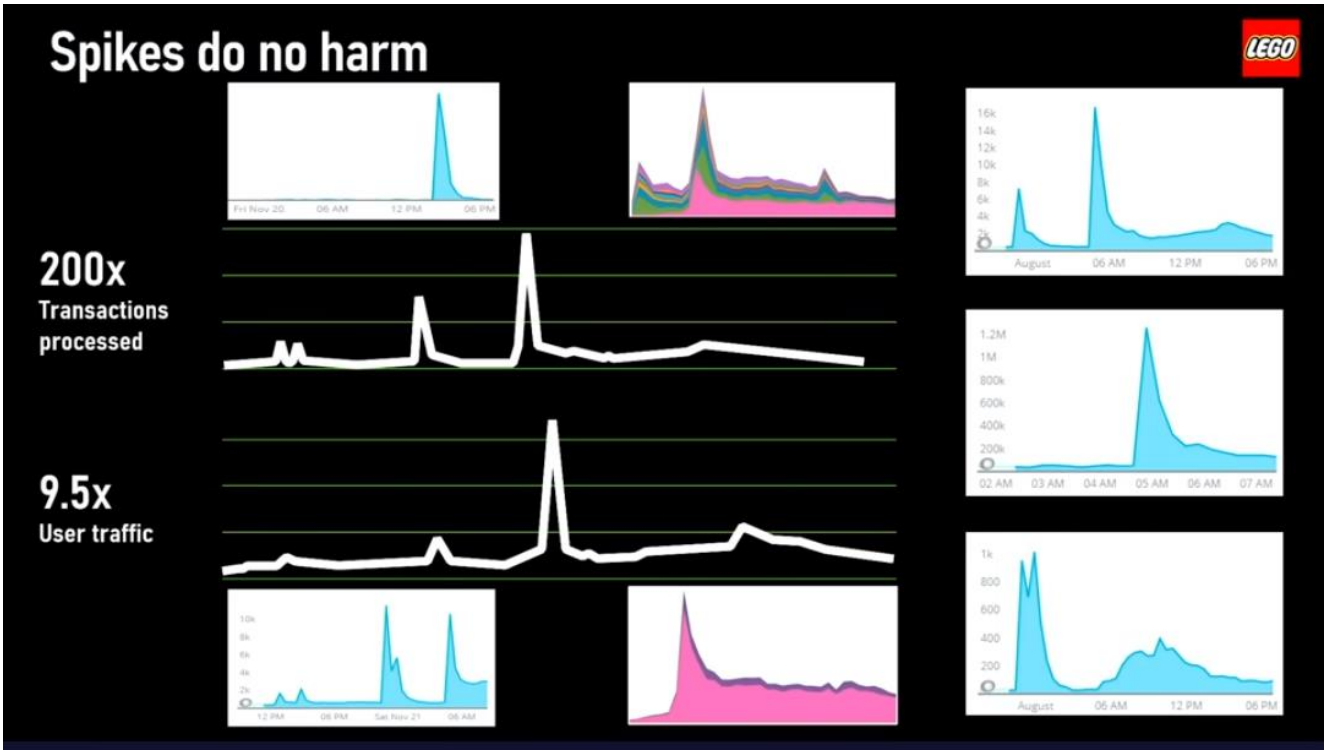
Summer 2022

- 25+ squads
- 150+ engineers
- multiple repositories
- multiple AWS prod accounts
- many microservices
- many lambda function
- and growing!



Business growth





To close...

The future is already here;
it's just not very evenly distributed.
- William Gibson

The future is already here;
it's just not very **event**ly distributed.

Let **event-driven architecture** be the enabler!