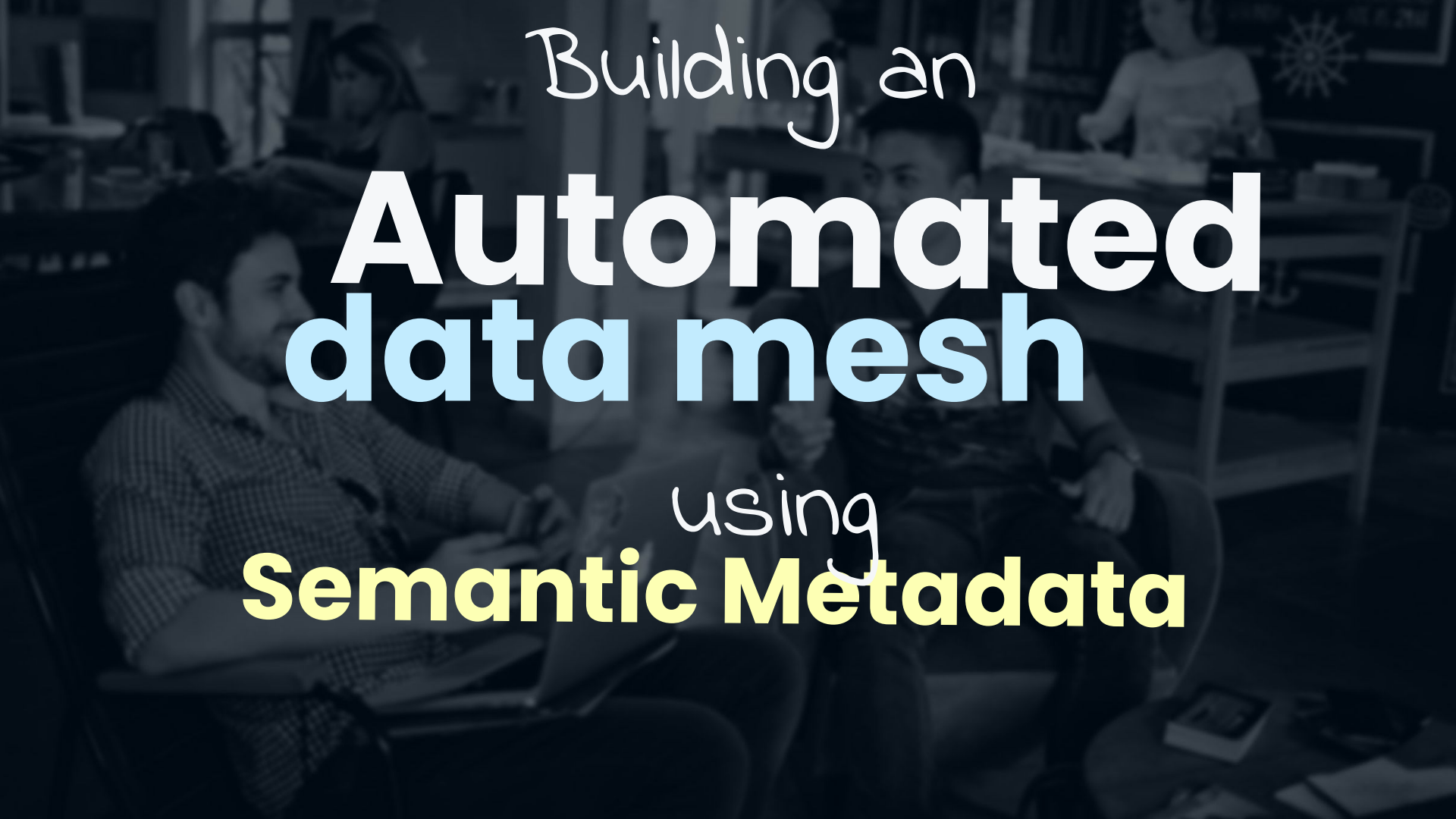
SOFTWARE DEVELOPMENT CONFERENCE

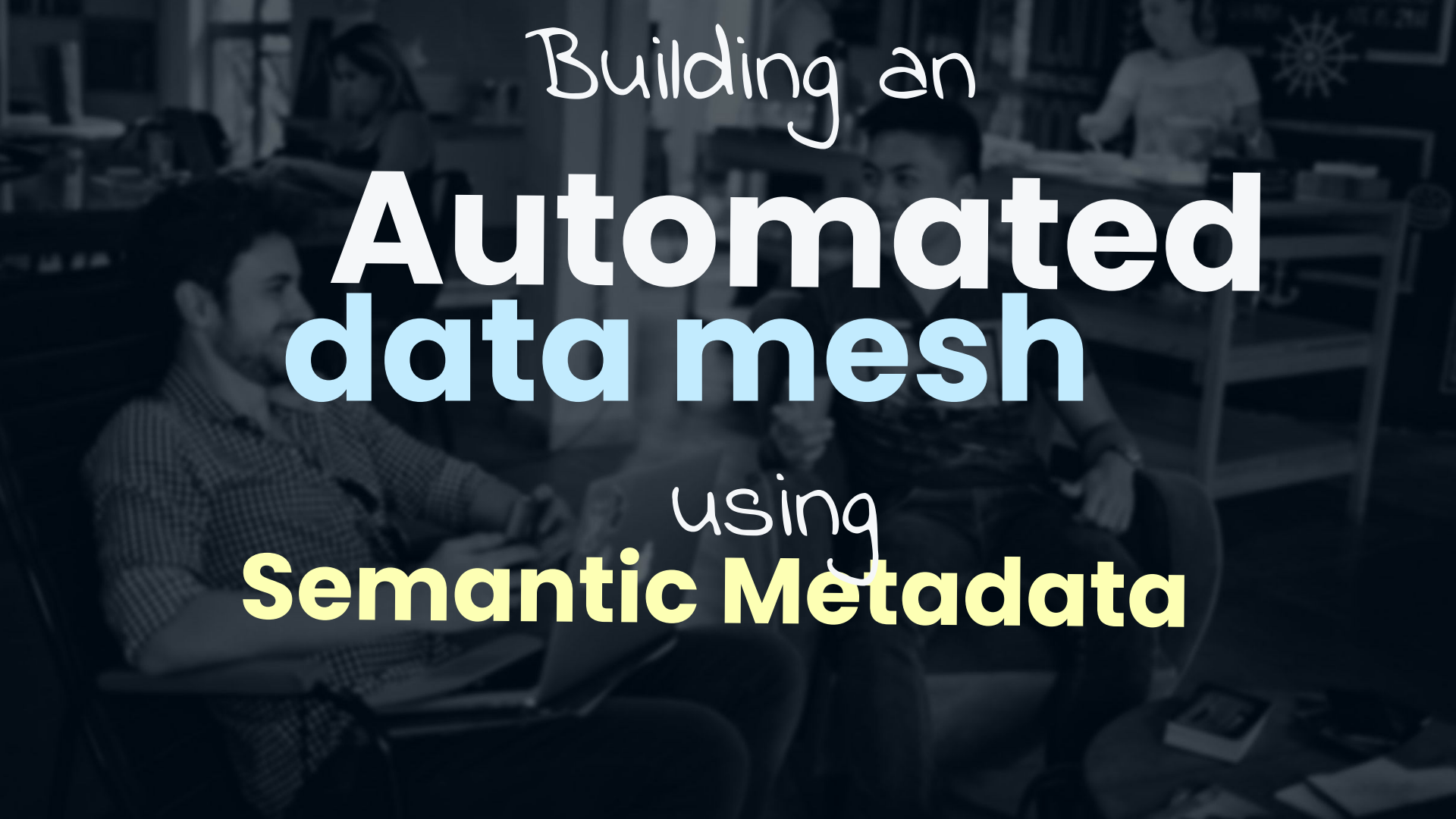
YOW! LONDON 2022

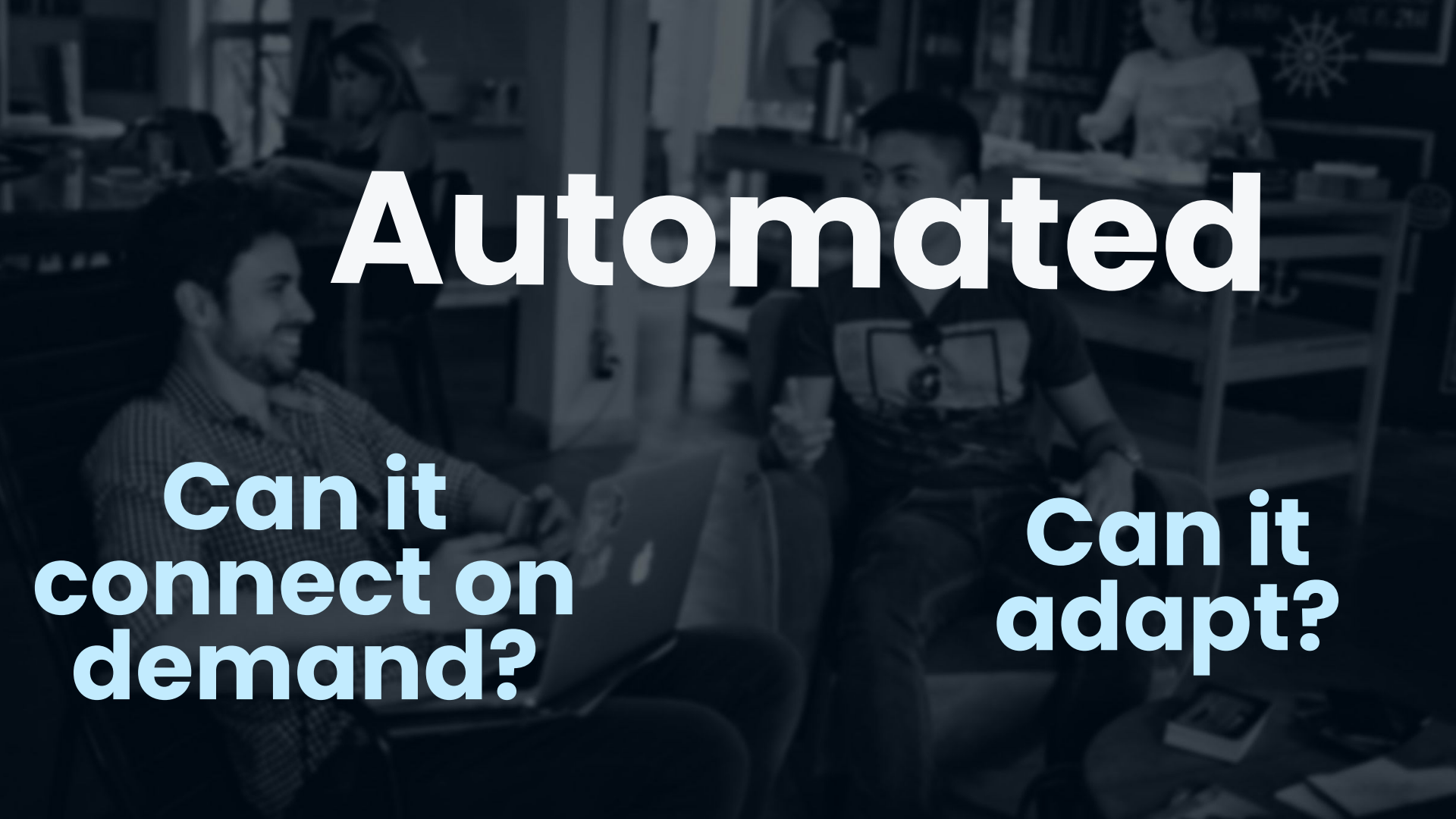
Using **Semantic Metadata** To create an **Automated Microservice Data Mesh**

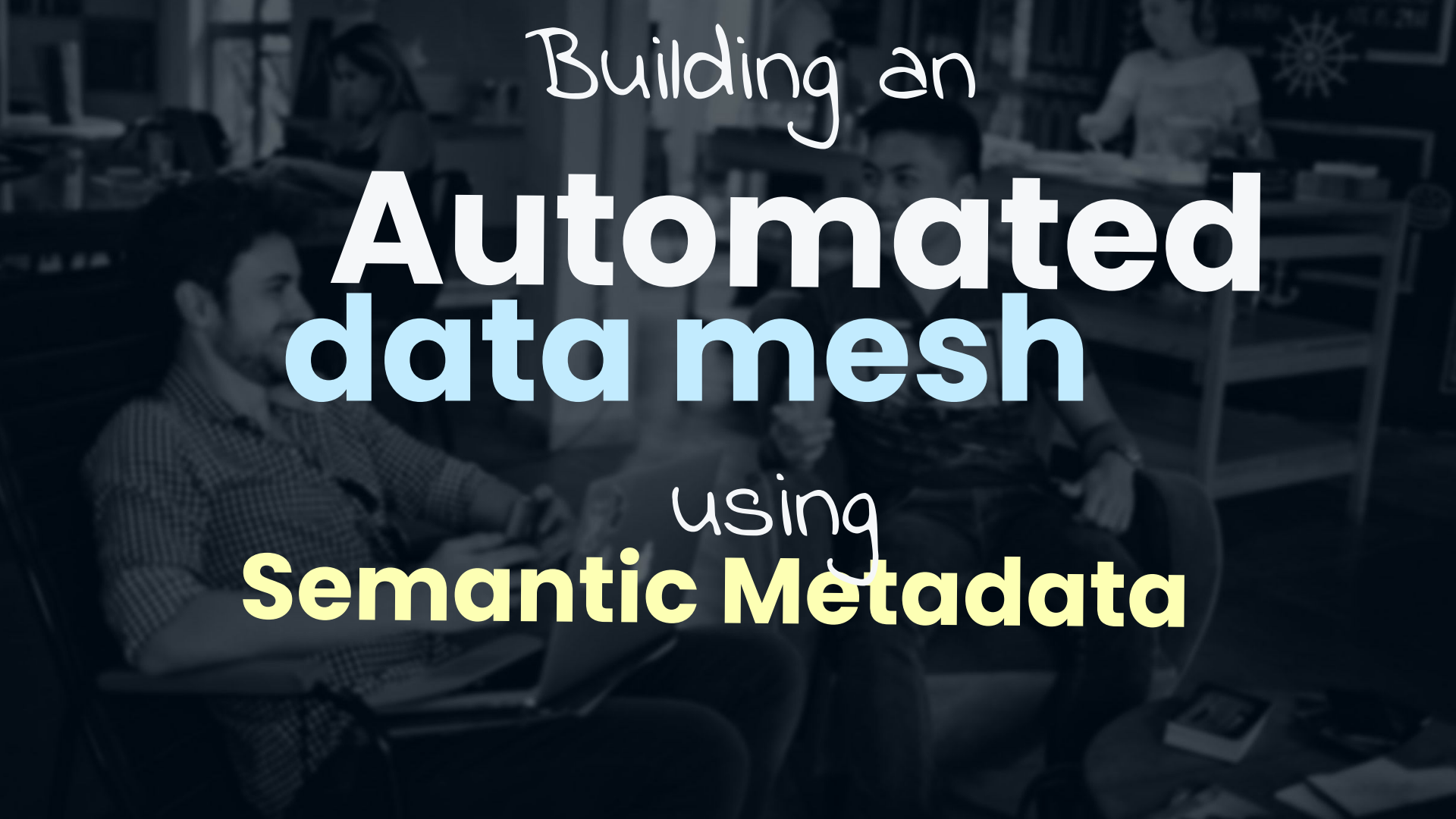
Marty Pitt
Founder - Vyne / Taxi


Building an
Automated
data mesh
using
Semantic Metadata

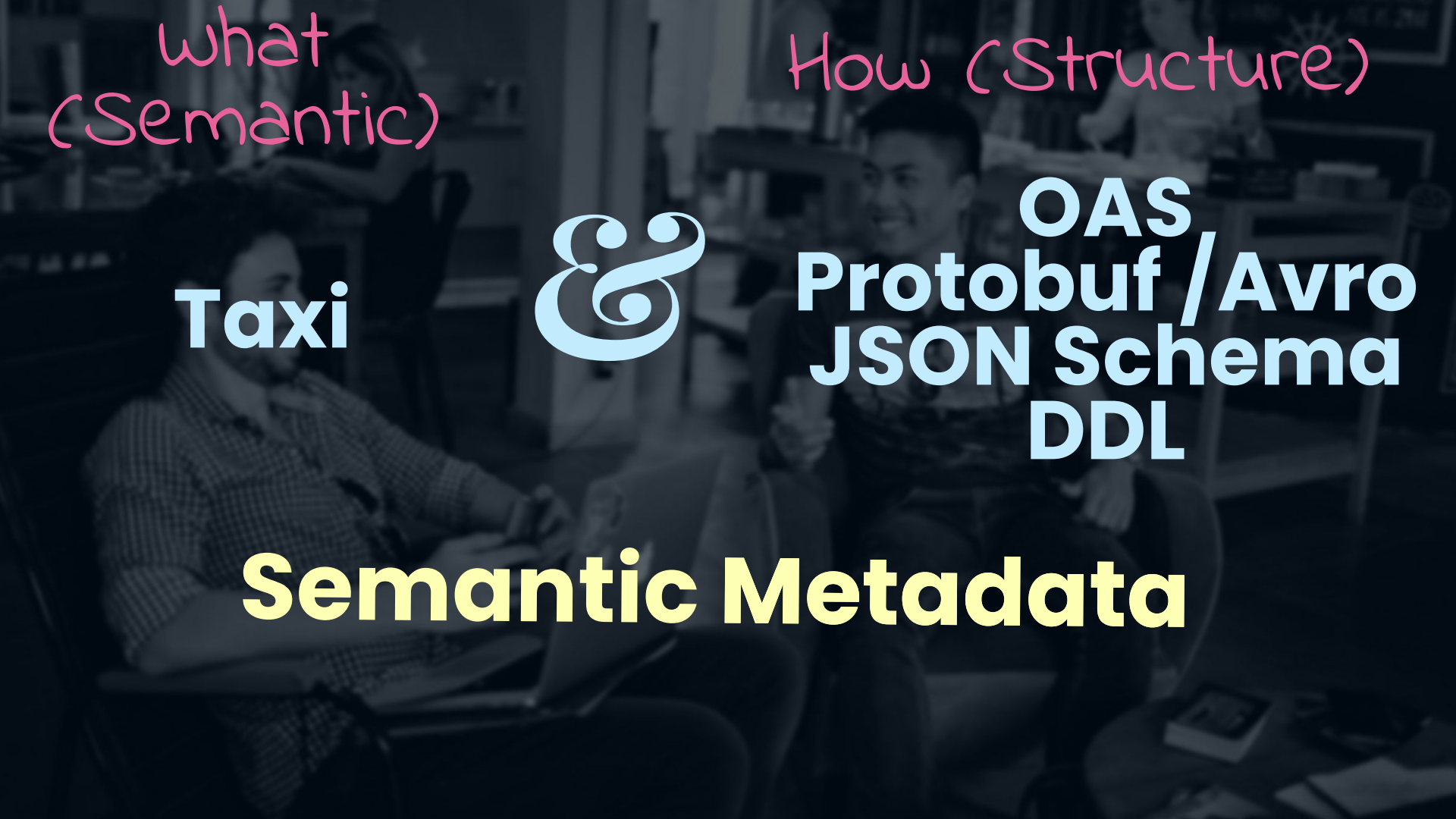

data mesh
Services DBs Queues Lambdas
On Demand / No ETL

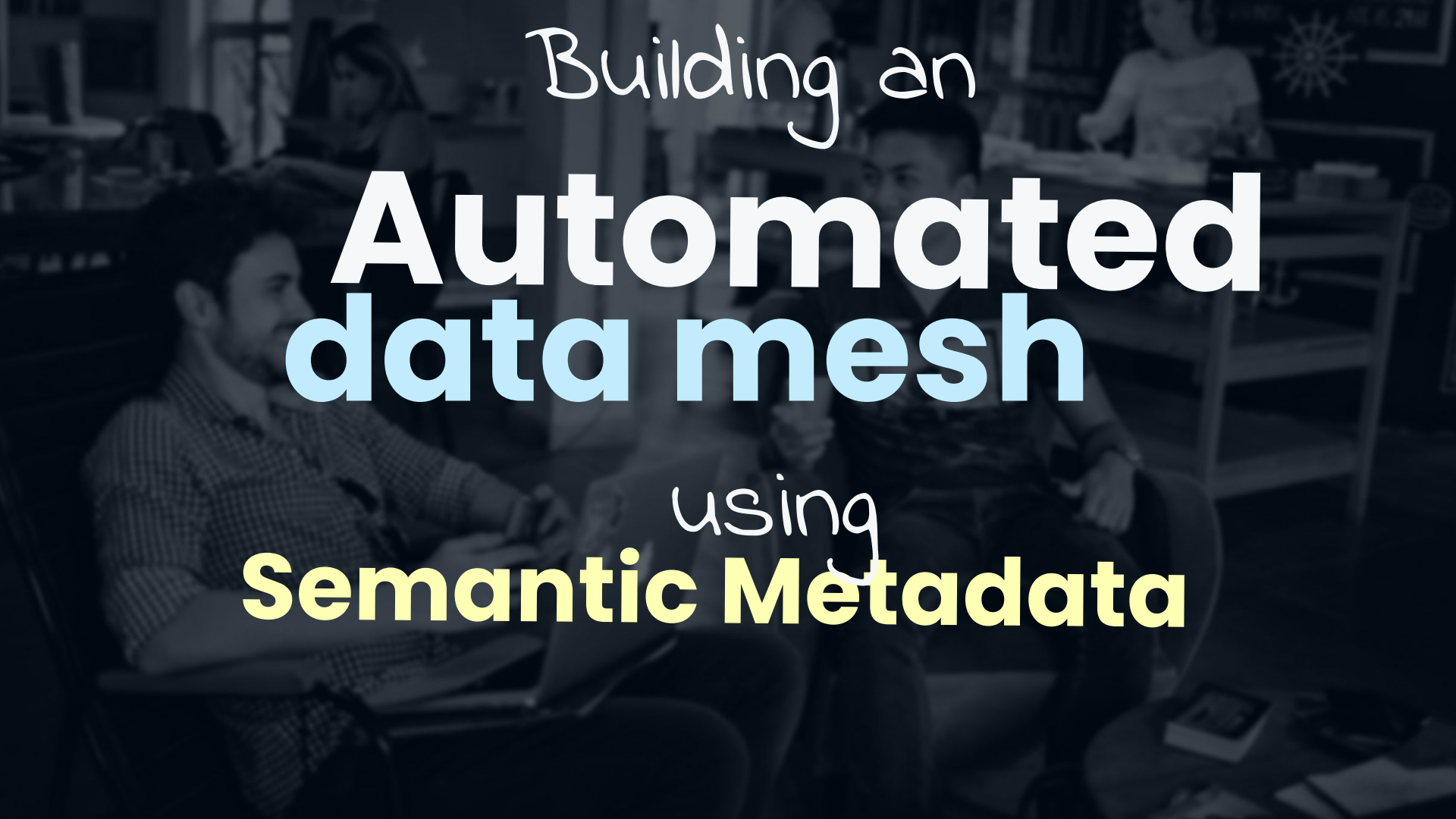

Building an
Automated
data mesh
using
Semantic Metadata

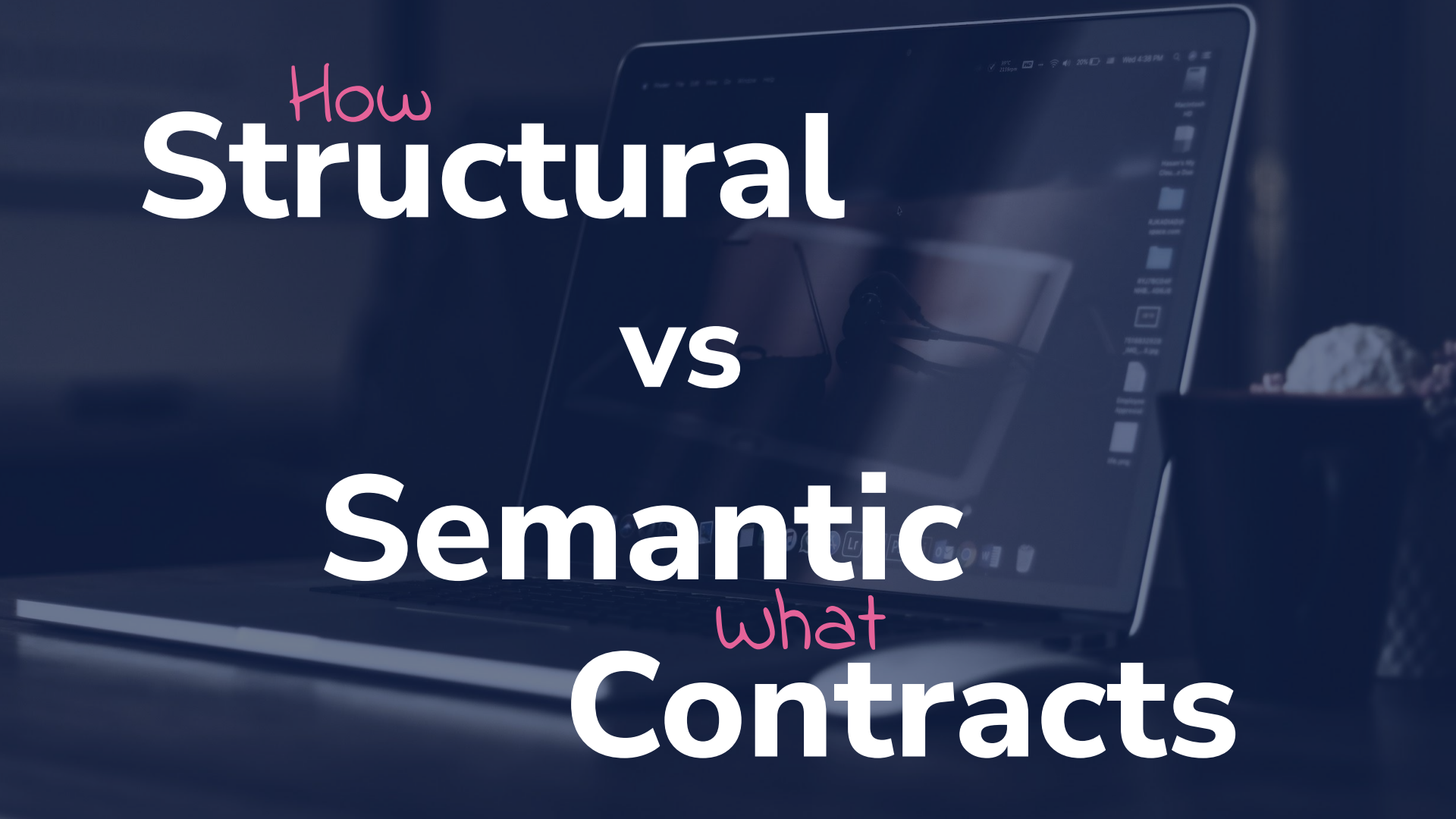

Automated
Can it connect on demand?
Can it adapt?

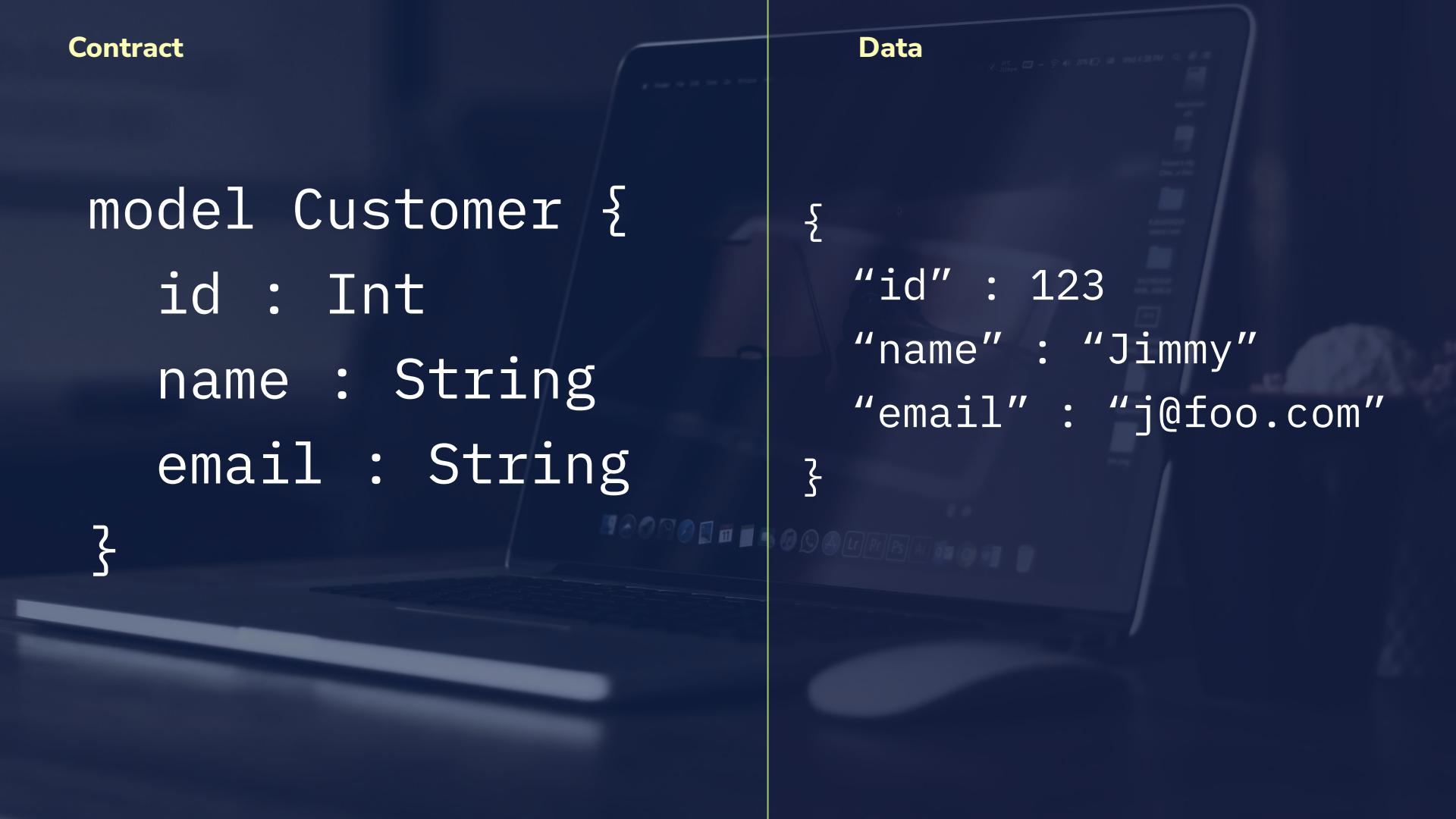

Building an
Automated
data mesh
using
Semantic Metadata

Semantic Metadata


What
(Semantic)
How (Structure)
Taxi
&
OAS
Protobuf / Avro
JSON Schema
DDL
Semantic Metadata


Building an
Automated
data mesh
using
Semantic Metadata


How
Structural
vs
Semantic
what
Contracts



Contract

```
model Customer {
   id : Int
   name : String
   email : String
}
```

Data

```
{
   “id” : 123
   “name” : “Jimmy”
   “email” : “j@foo.com”
}
```

Contract

```
model Customer {
   id : Int
   name : String
   email : String
}
```

Data

```
{
  “id” : 123
  “name” : “Jimmy”
  “email” : “j@foo.com”
}
```

Structure

“How to navigate a map, to find the good stuff?”

Contract

```
model Customer {
   id : Int
givenName : String
   email : String
}
```

Structure

Data

```
{
  “id” : 123
  “givenName” : “Jimmy”
  “email” : “j@foo.com”
}
```

Contract

```
model Customer {
   id : Int
givenName : String
  contacts : {
    email: String
    }
}
```

Structure

Data

```
{
  “id” : 123
  “givenName” : “Jimmy”
  “contacts” : {
    “email”: “j@foo.com”
    }
}
```

Contract

```
model Customer {
   id : Int
givenName  : String
  contacts  : {
    email: String
    }
}
```

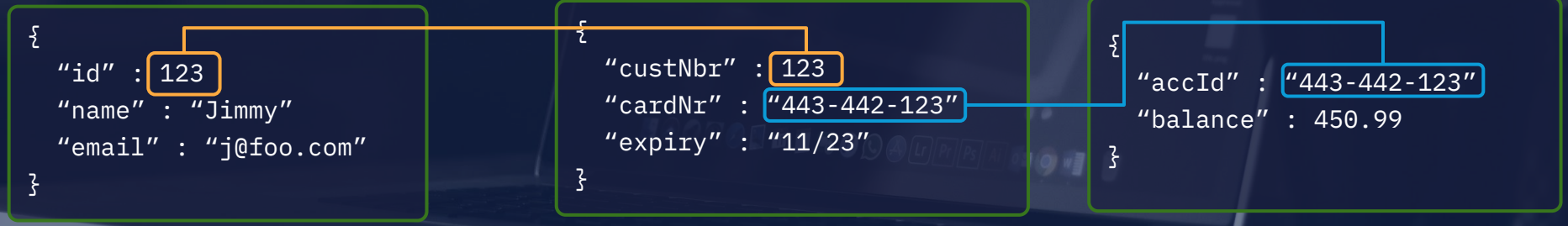
Data

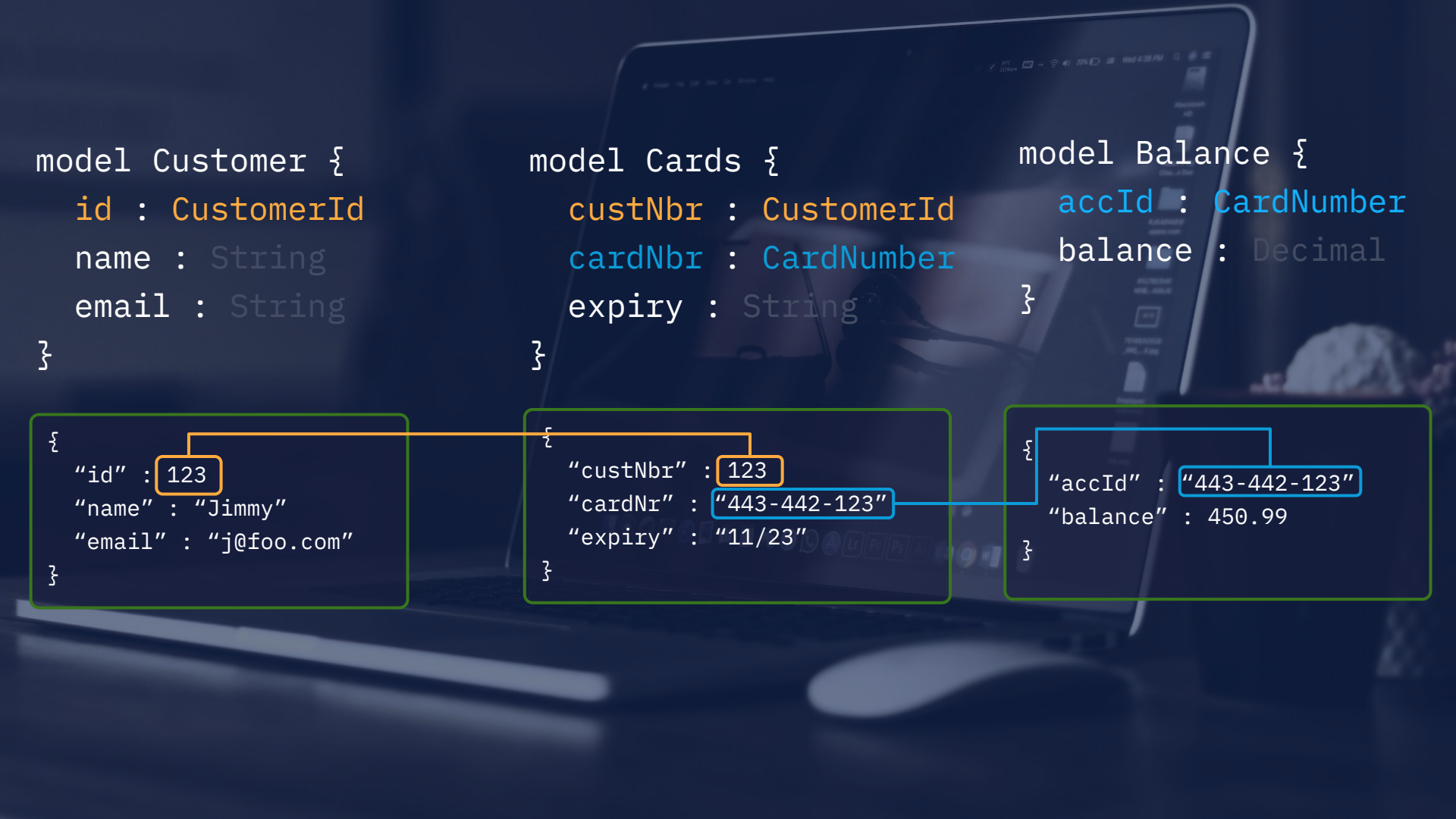
```
{
  “id” : 123
  “givenName” : “Jimmy”
  “contacts” : {
    “email”: “SW18 9EN”
    }
}
```

```
model Customer {
  id : Int
  name : String
  email : String
}
```

```
model Cards {
  custNbr : Int
  cardNbr : String
  expiry : String
}
```

```
model Balance {
  accId : String
  balance : Decimal
}
```

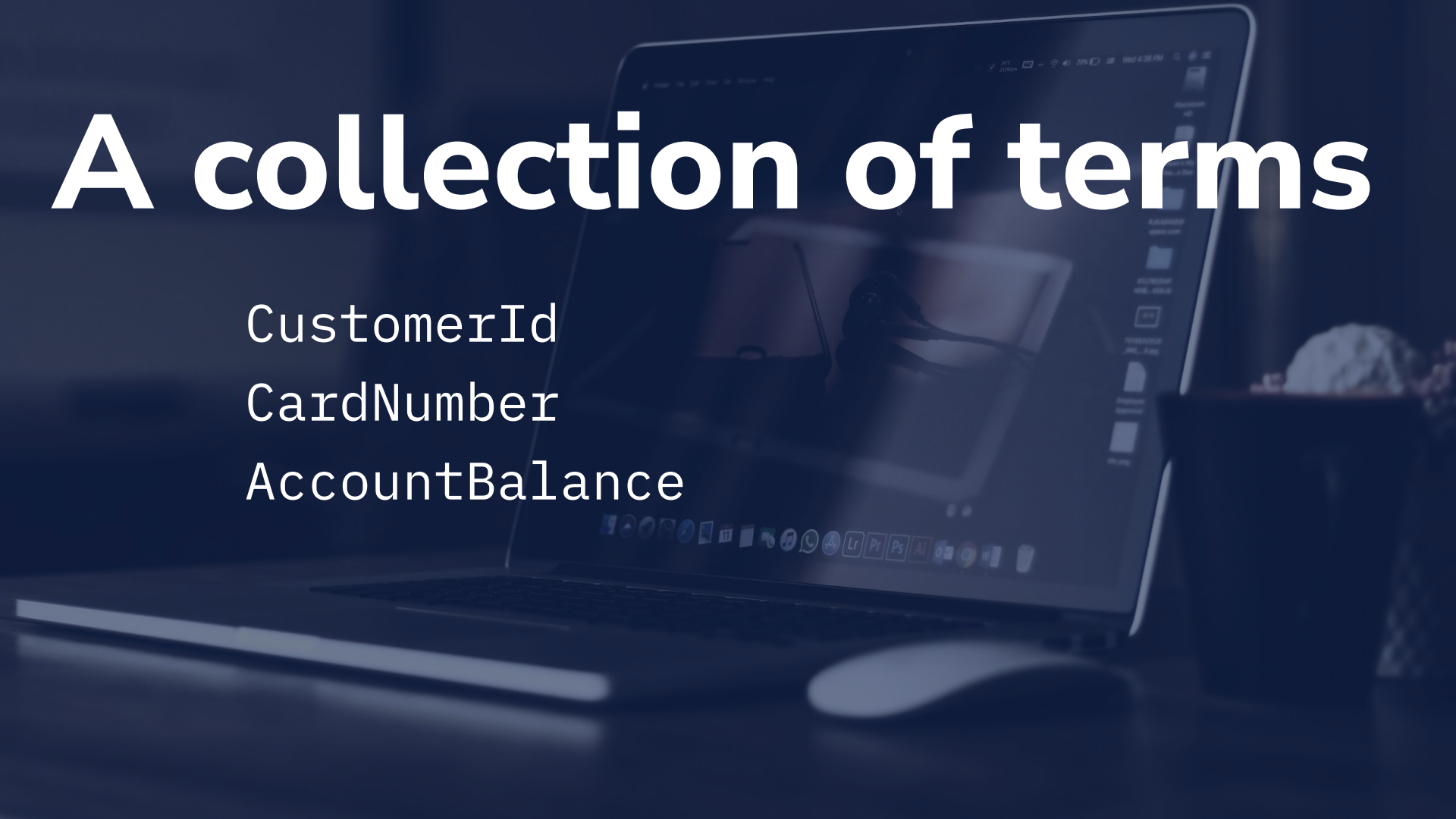



model Customer {
id : CustomerId
name : String
email : String
}
model Cards {
custNbr : CustomerId
cardNbr : CardNumber
expiry : String
}
model Balance {
accId : CardNumber
balance : Decimal
}
{
“id” : 123
“name” : “Jimmy”
“email” : “j@foo.com”
}
{
“custNbr” : 123
“cardNr” : “443-442-123”
“expiry” : “11/23”
}
{
“accId” : “443-442-123”
“balance” : 450.99
}


model Customer {
id : CustomerId
name : String
email : String
}
model Cards {
custNbr : CustomerId
cardNbr : CardNumber
expiry : String
}
model Balance {
accId : CardNumber
balance : Decimal
}
// OpenAPI Spec:
Customer:
properties:
id:
type: string
x-taxi-type: CustomerId
// Protobuf
message Card {
string custNbr = 1 [(DataType)="CustomerId"];
string cardNbr = 2 [(DataType)="CardNumber"];
}
// Kotlin
@DataType("CardNumber")
typealias CardNumber = String
data class Balance(
val accId: CardNumber
)


Semantic Metadata
A collection of terms
which
form a semantic contract
and are
shared between systems

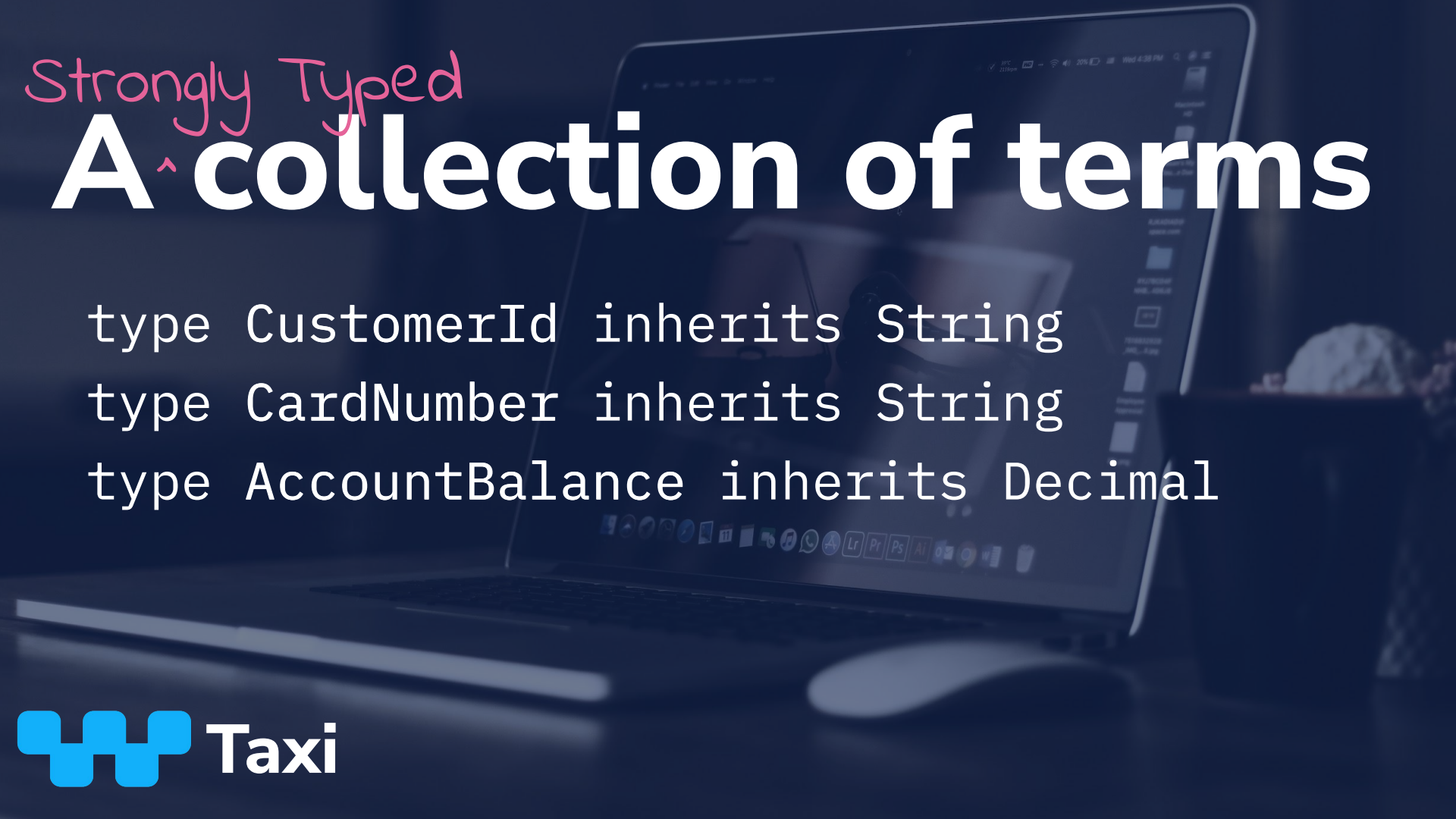
A collection of terms



CustomerId

CardNumber

AccountBalance


Strongly Typed
A collection of terms
type CustomerId inherits String
type CardNumber inherits String
type AccountBalance inherits Decimal
Taxi

Strongly Typed

A collection of terms

```
type CustomerId inherits String
type CardNumber inherits String
type AccountBalance inherits Decimal
type FirstName inherits Name
type BuyerId inherits CustomerId
```

Taxi

Semantic Contracts

"Single" / No structure

Scalar data elements that have the same semantic meaning

Semantic Contracts

```
model Customer {
  id : CustomerId
  name : String
  email : String
}
```

```
model Cards {
  custNbr : CustomerId
  cardNbr : CardNumber
  expiry : String
}
```

```
model Balance {
  accId : CardNumber
  balance : Decimal
}
```

Semantic Contracts

```
model ImdbReview {
  filmId : FilmId
  …
}
```

```
{
 "filmId" : "imdb-123"
}
```

```
model RottenTomatoesReview {
  filmId : FilmId
  …
}
```

```
{
 "filmId" : "rotten-123"
}
```

Semantic Contracts

```
model ImdbReview {
  filmId : FilmId
  …
}
```

```
{
 "filmId" : "imdb-123"
}
```

Semantically Different

```
model RottenTomatoesReview {
  filmId : FilmId
  …
}
```

```
{
 "filmId" : "rotten-123"
}
```

Semantic Contracts

```
model ImdbReview {
  filmId : imdb.FilmId
  …
}
```

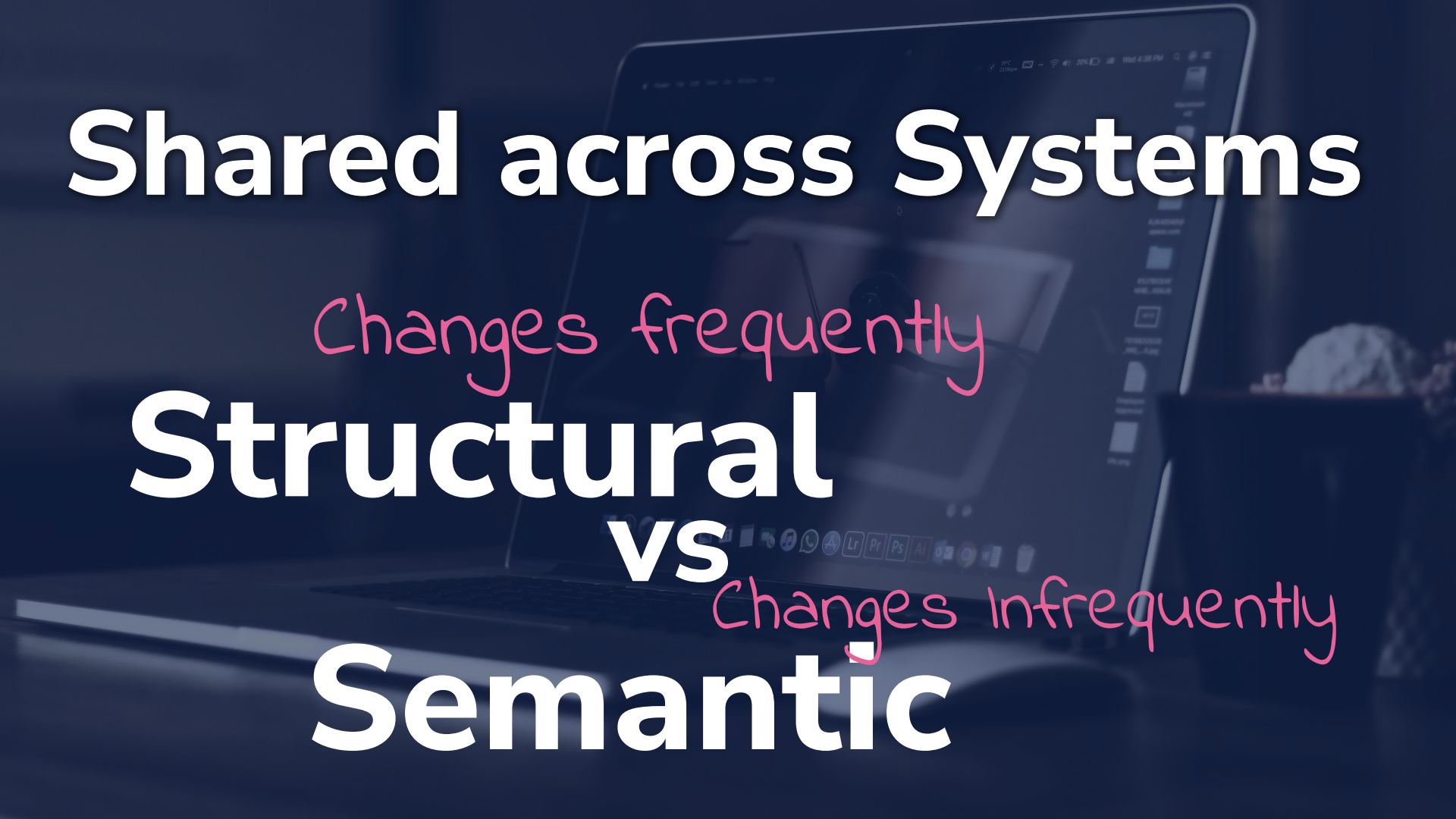
```
model RottenTomatoesReview {
  filmId : tomato.FilmId
  …
}
```

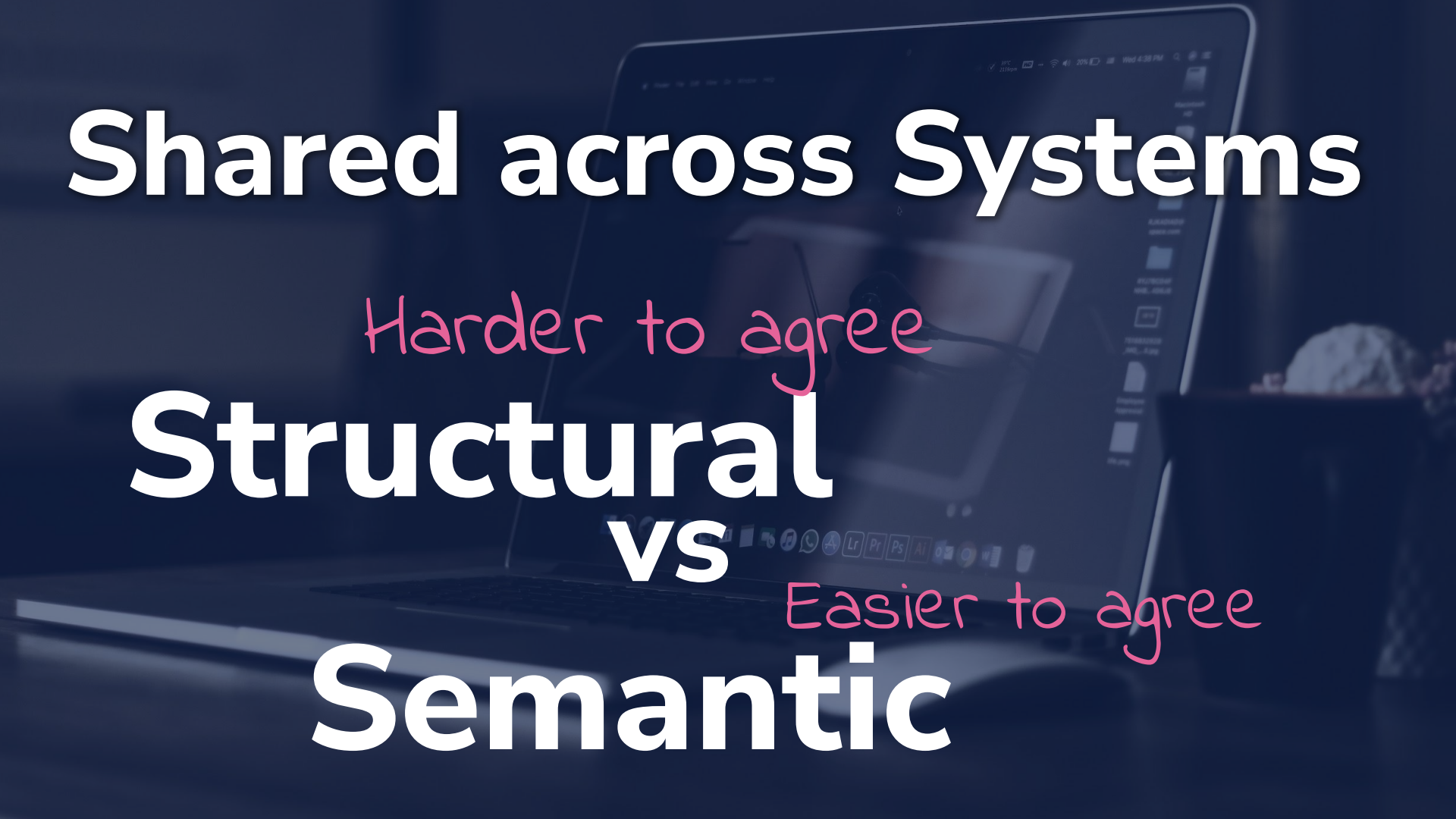
```
{
 "filmId" : "imdb-123"
}
```

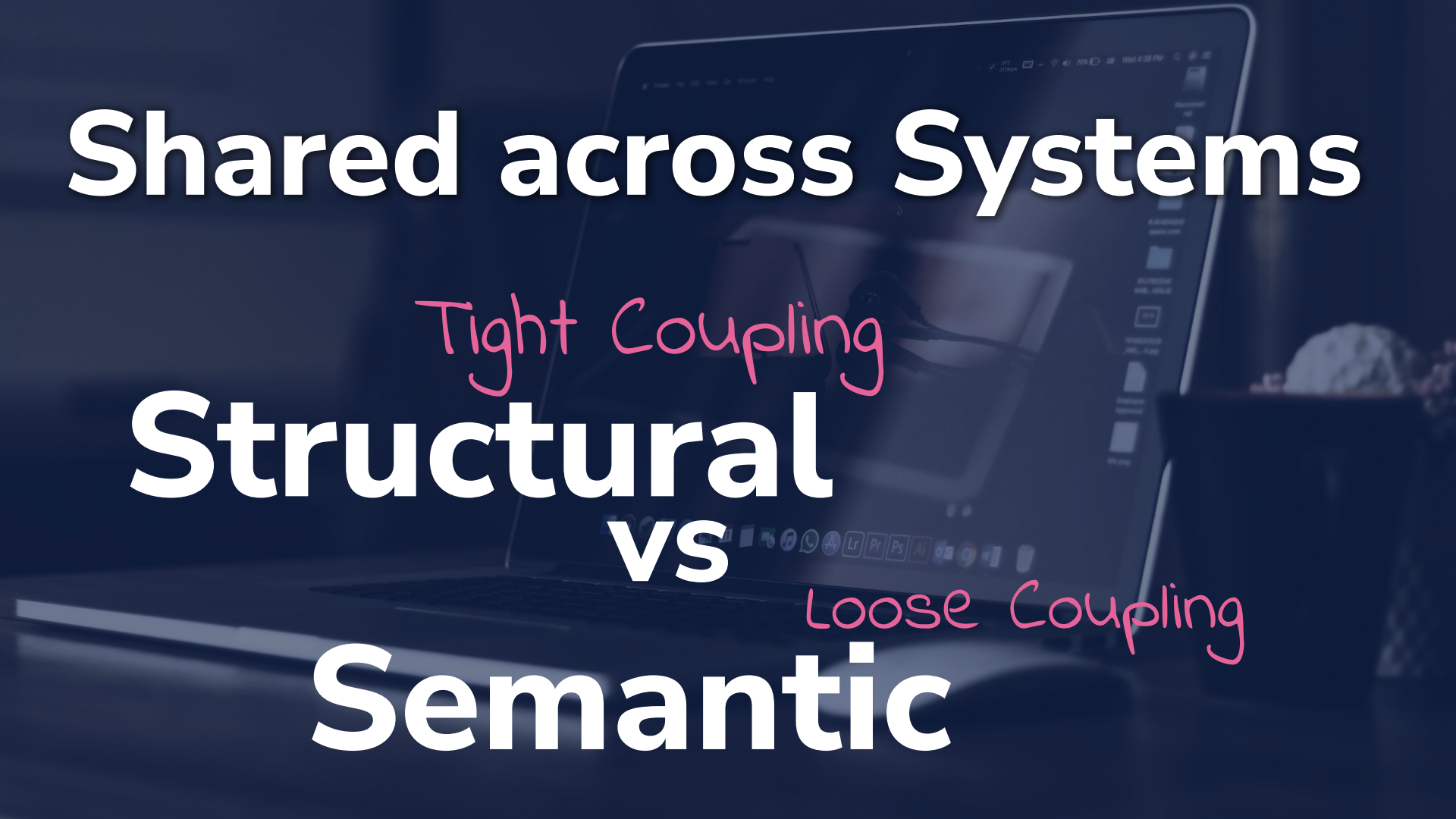
```
type imdb.FilmId inherits FilmId
type tomato.FilmId inherits FilmId
```

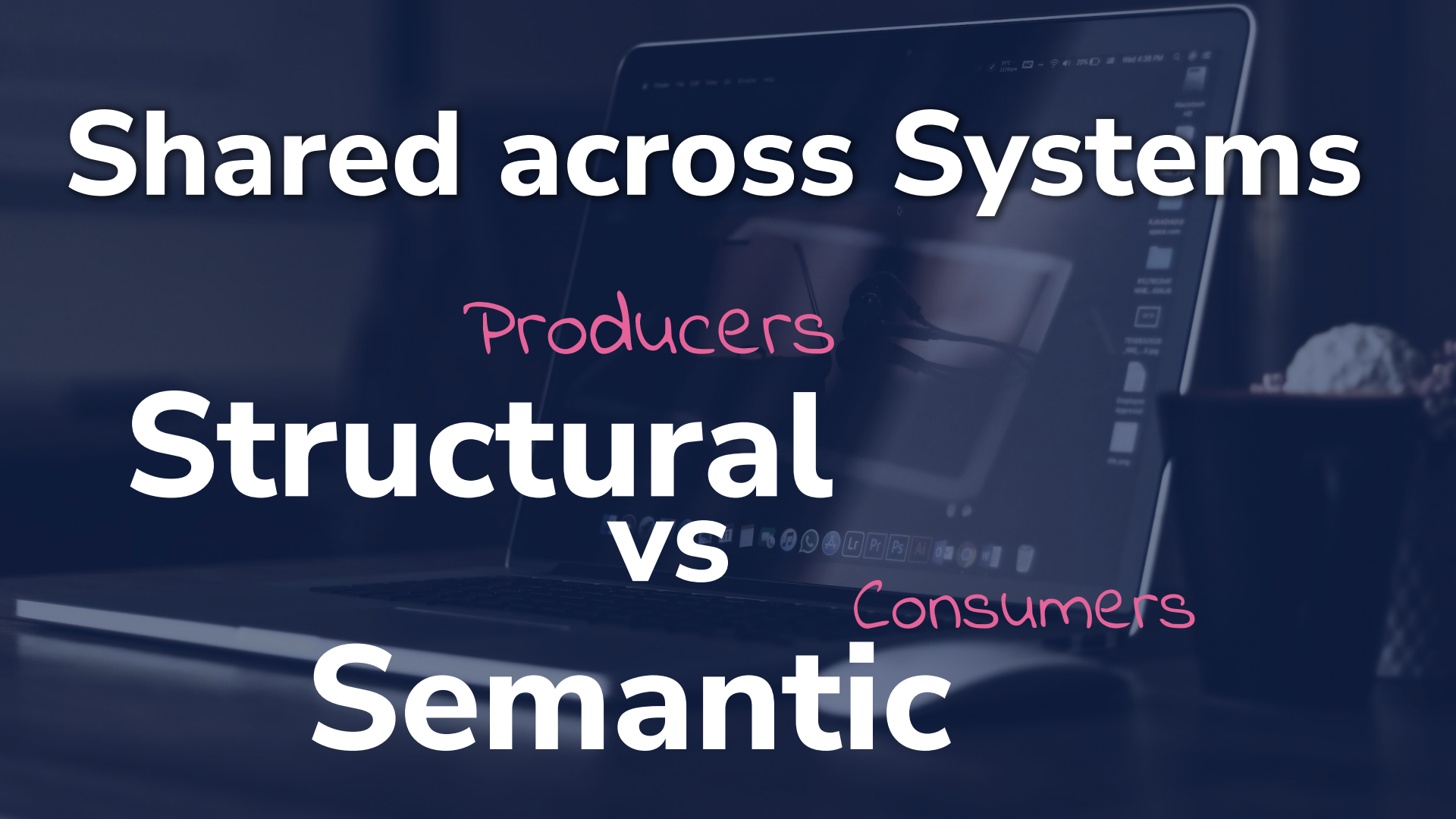
```
{
 "filmId" : "rotten-123"
}
```

Shared across Systems

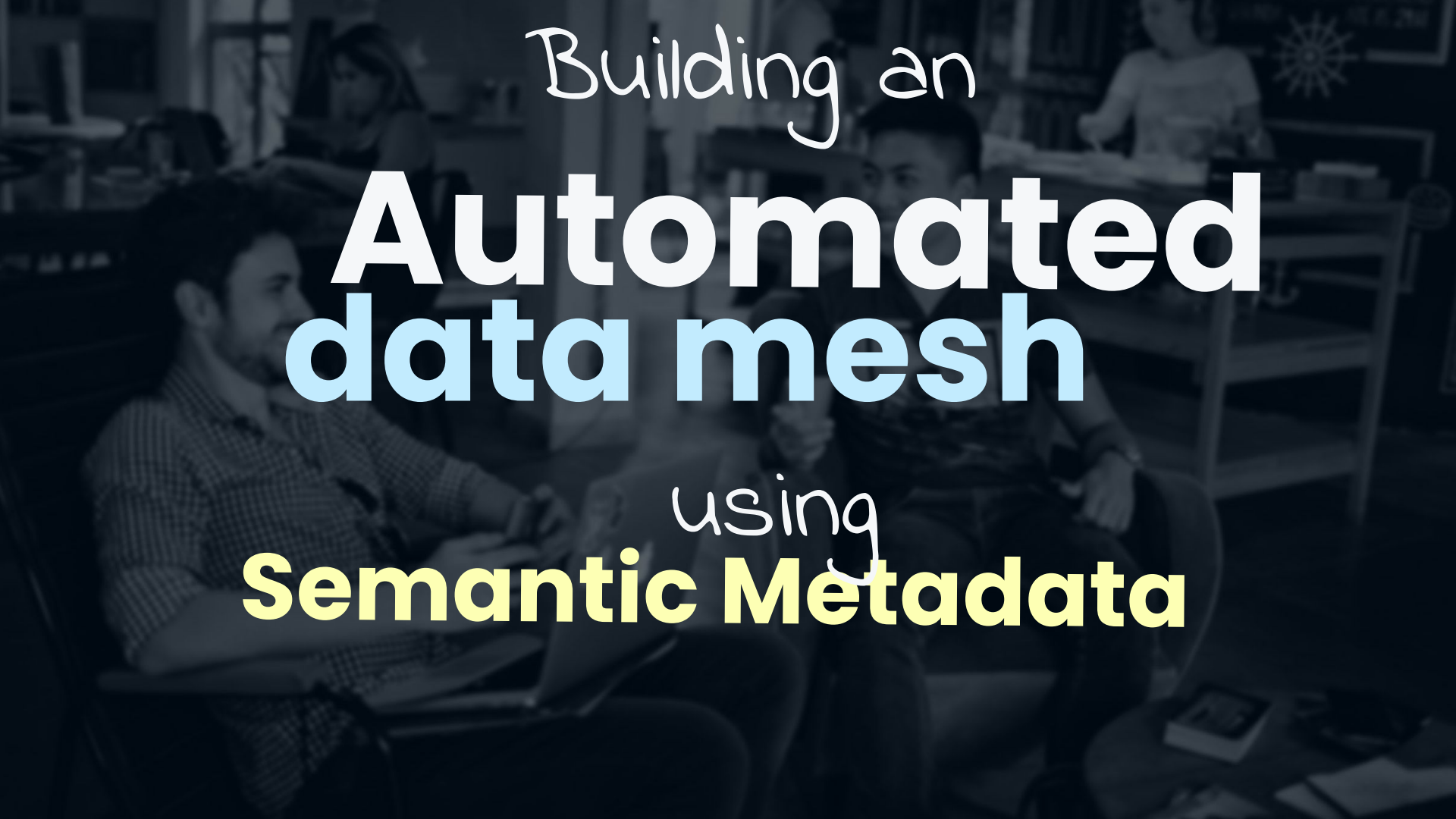

Shared across Systems
Changes frequently
Structural
vs
Changes Infrequently
Semantic

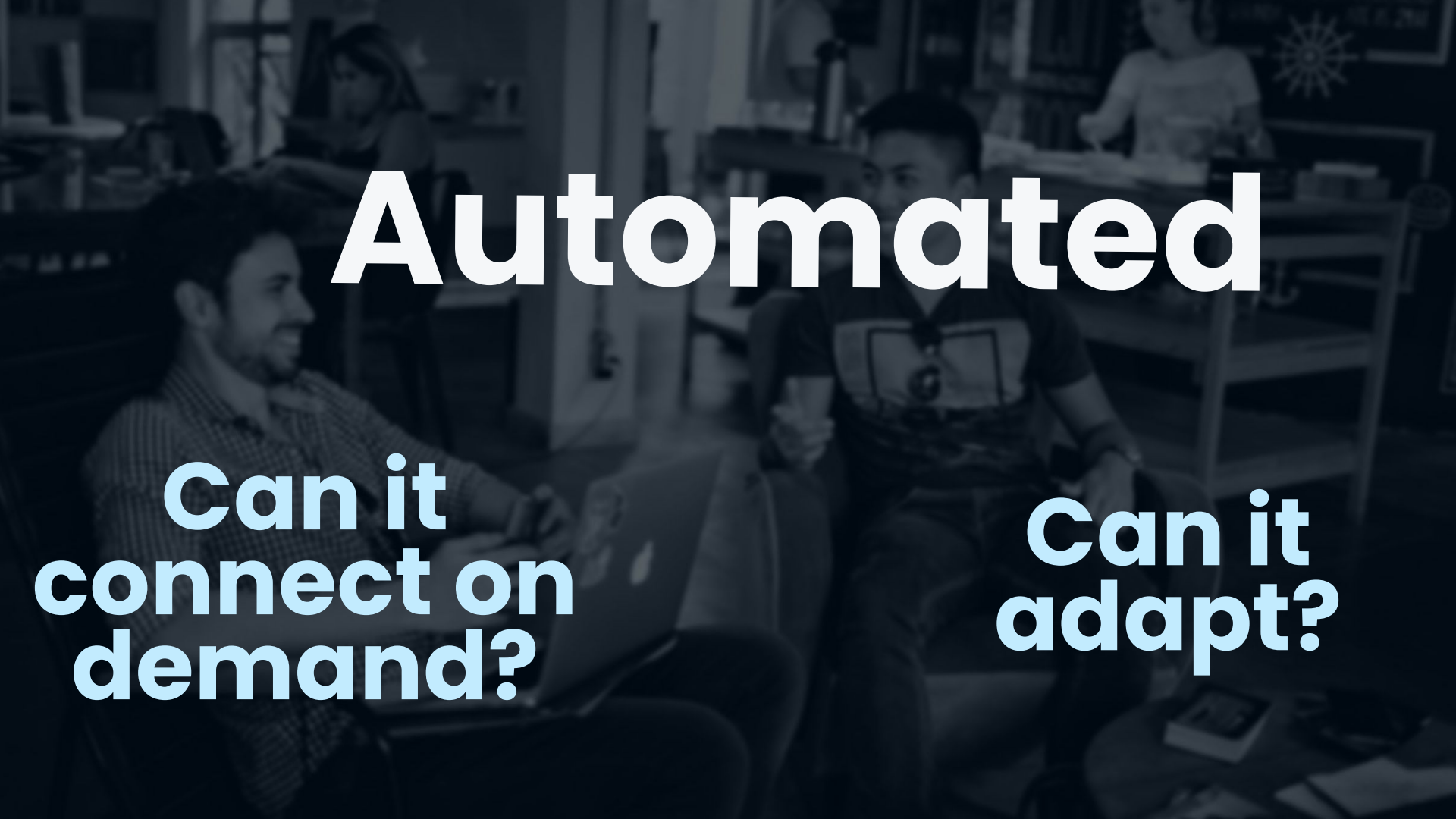

Shared across Systems
Harder to agree
Structural
vs
Easier to agree
Semantic


Shared across Systems
Tight Coupling
Structural
vs
Loose Coupling
Semantic


Shared across Systems
Producers
Structural
vs
Consumers
Semantic


Semantic Metadata
A collection of terms
which
Form a semantic contract
and are
Shared Across Systems

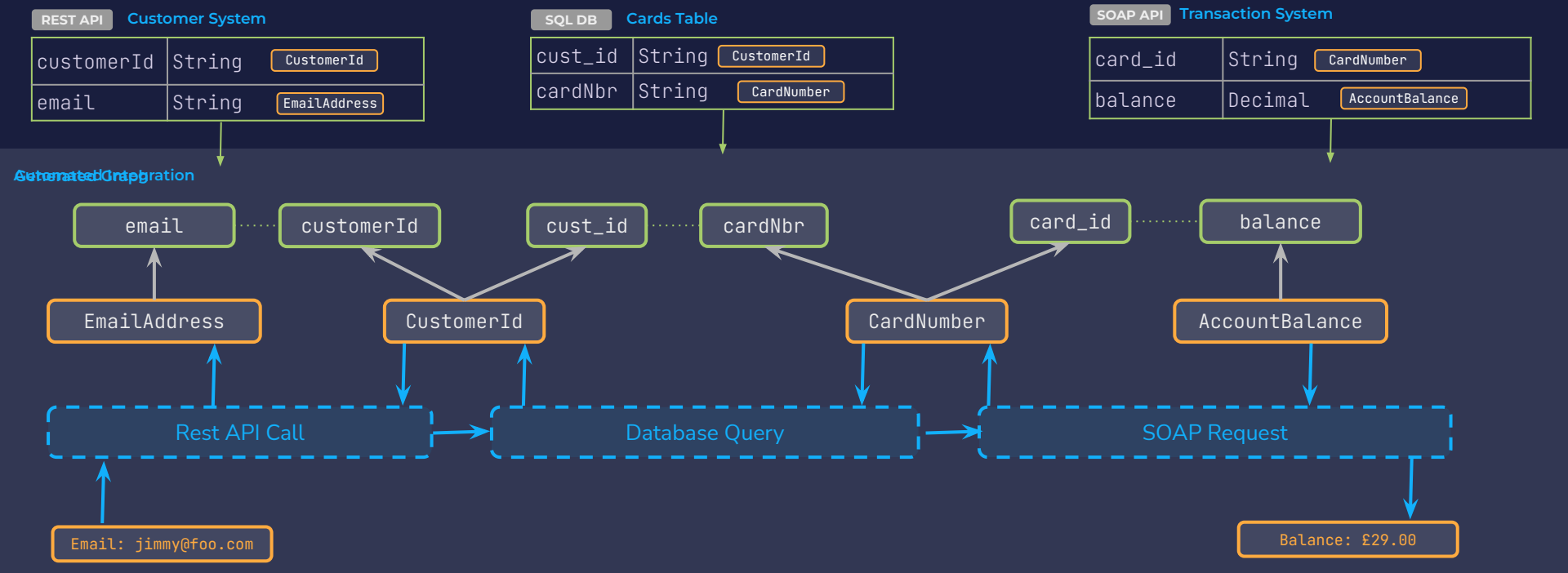

Building an
Automated
data mesh
using
Semantic Metadata


Automated
Can it connect on demand?
Can it adapt?

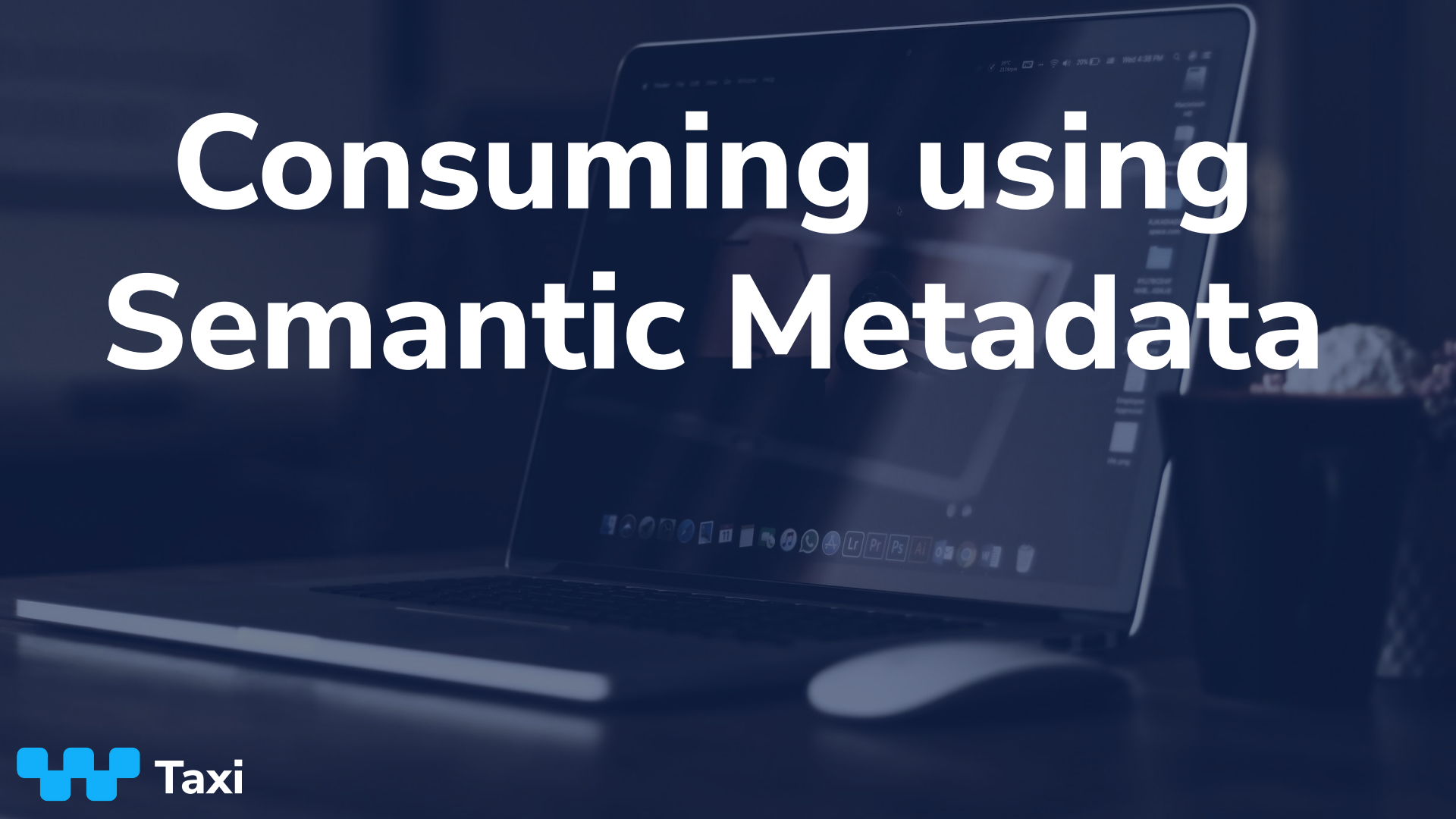
Automated Data Mesh

Vyne

"Given this **Email Address**,
What is the customer's **Account Balance?**"



Consuming using Semantic Metadata



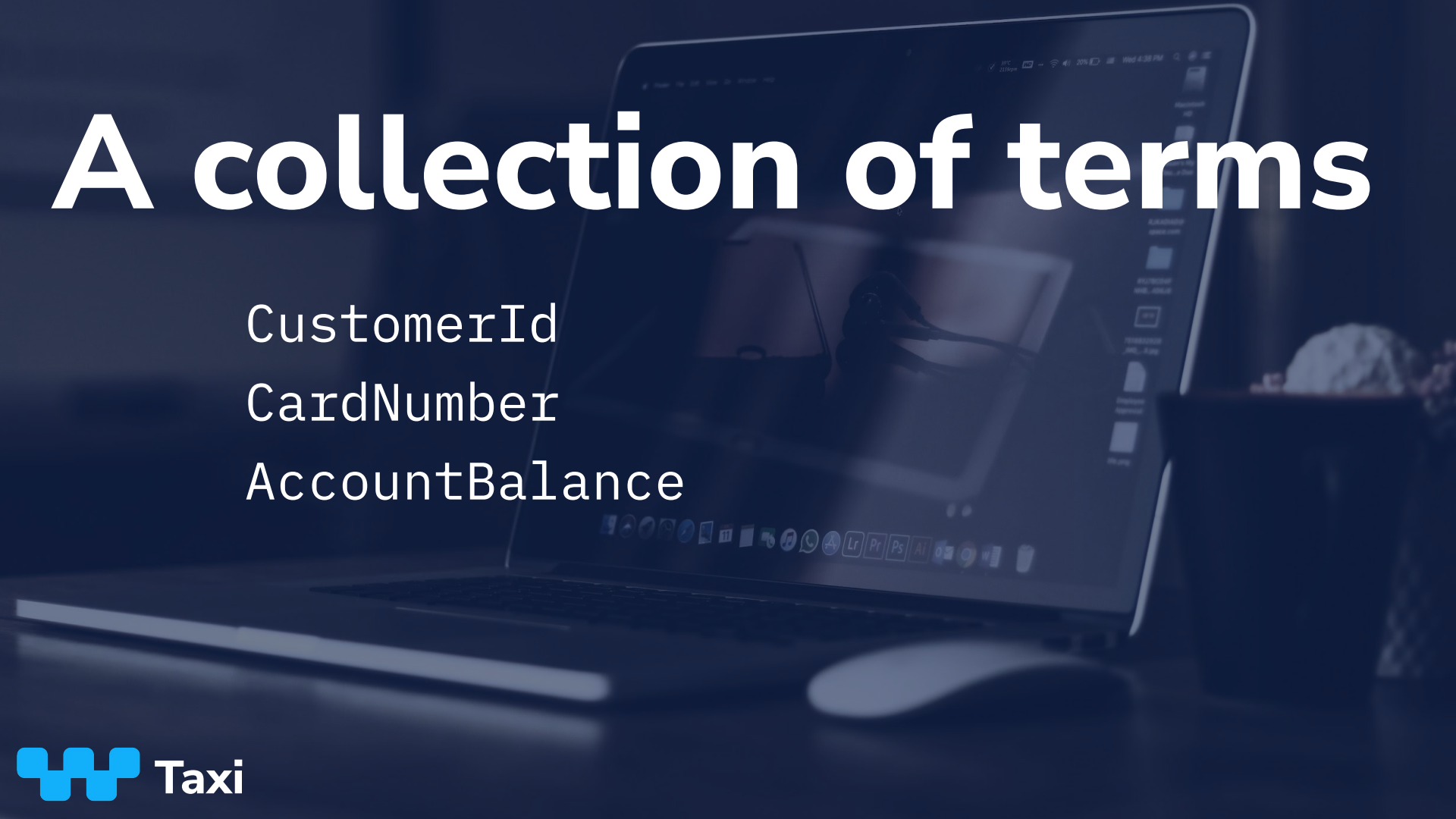
Taxi

A collection of terms

CustomerId

CardNumber

AccountBalance



Taxi

A collection of terms

```
given { CustomerId = ‘123’ }
find { AccountBalance }
```

A collection of terms

```
given { CustomerId = ‘123’ }
find {
   name : CustomerName
   balance: AccountBalance
}
```

Consumers define the contract

Taxi

Putting it all together

Taxi

taxilang.org

Semantic Metadata
& Query language

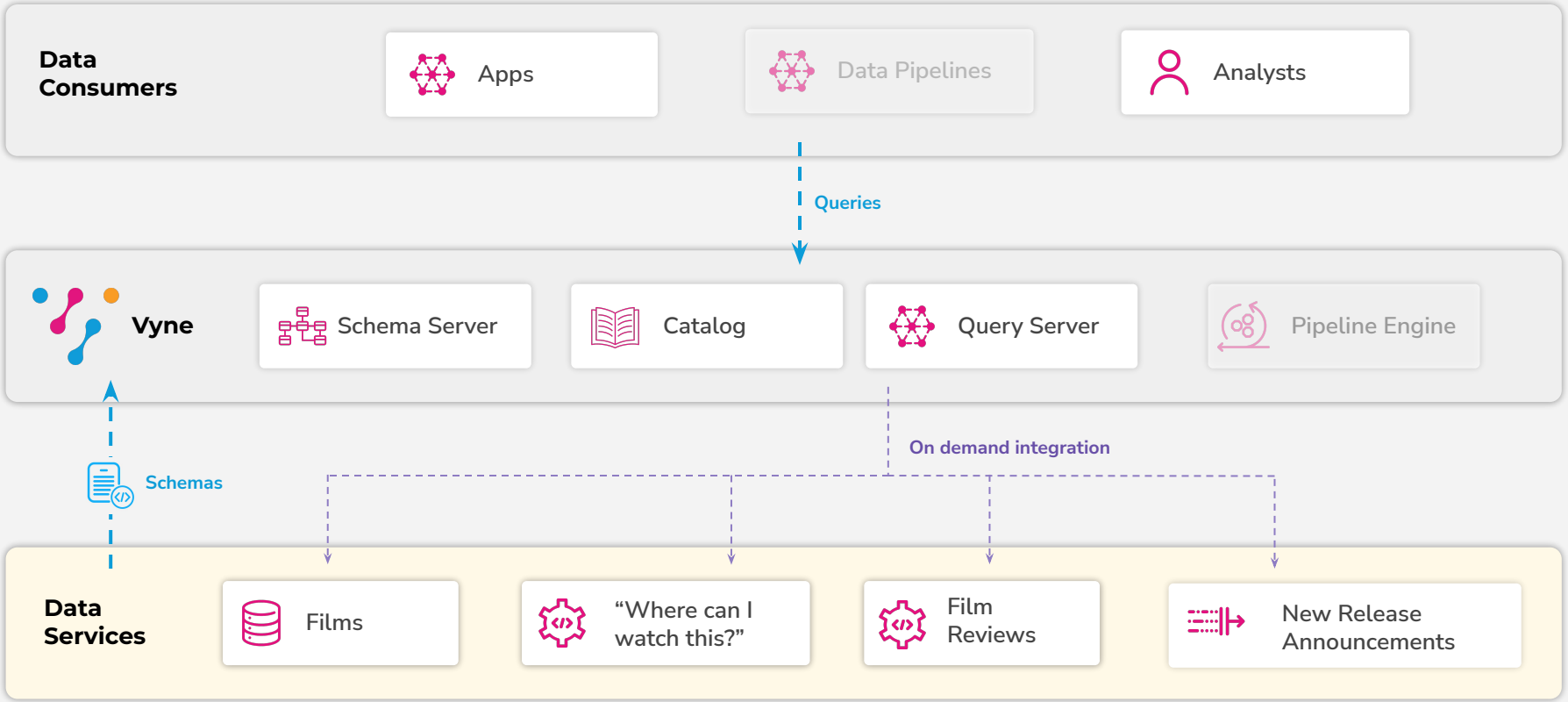
Vyne

vyne.co

Automated
Data Mesh


Demo
Time

Demo ecosystem




Taxi
taxilang.org
Semantic Metadata
& Query language
Vyne
vyne.co
Automated
Data Mesh