Event-Driven Architectures in Node.js

JEFF MATAYA

@jmataya

@thebigmatay

jeff@jeffmataya.com

Microsoft

FoxCommerce

$HOP

Let's talk events.

THE BASICS

- What are event-driven architectures?
- Why are they useful?
- How do they work?

BUILDING EVENT-DRIVEN SYSTEMS

- Selecting an event bus
- Consumers and producers in Node
- Fun use cases

WRAPPING UP

- Lessons learned
- Serverless and the future
- Resources
- Questions

Let's talk events.

THE BASICS

- What are event-driven architectures?
- Why are they useful?
- How do they work?

BUILDING EVENT-DRIVEN SYSTEMS

- Selecting an event bus
- Consumers and producers in Go
- Fun use cases

WRAPPING UP

- Lessons learned
- Serverless and the future
- Resources
- Questions

What are event-driven architectures?

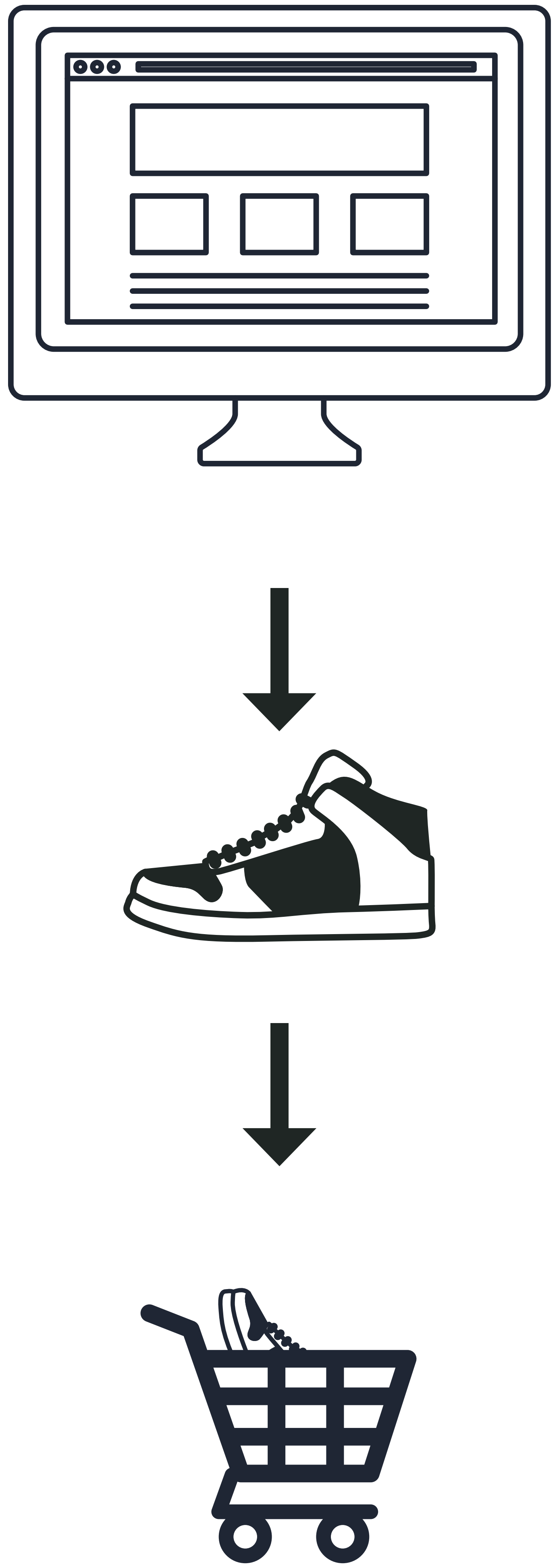
Systems that store all of actions that occur in a system, rather than storing the current state of the application.

Application state is a first-level derivative of all actions that have occurred.

A traditional system.

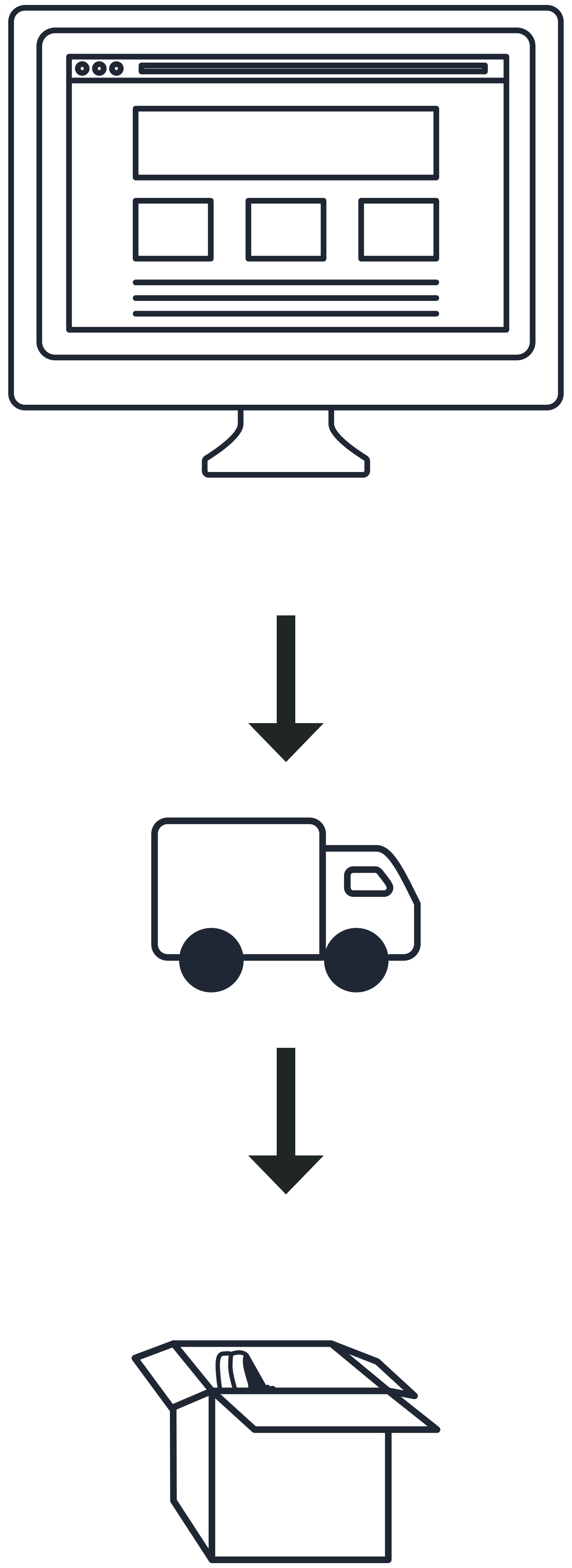
```
“order”: {
  “orderRef”: “ABC”,
  “customerId”: 1,
  “lineItems”: [],
  “shippingAddress”: {},
  “payment”: {},
  “orderTotal”: 0
}
```

A traditional system.



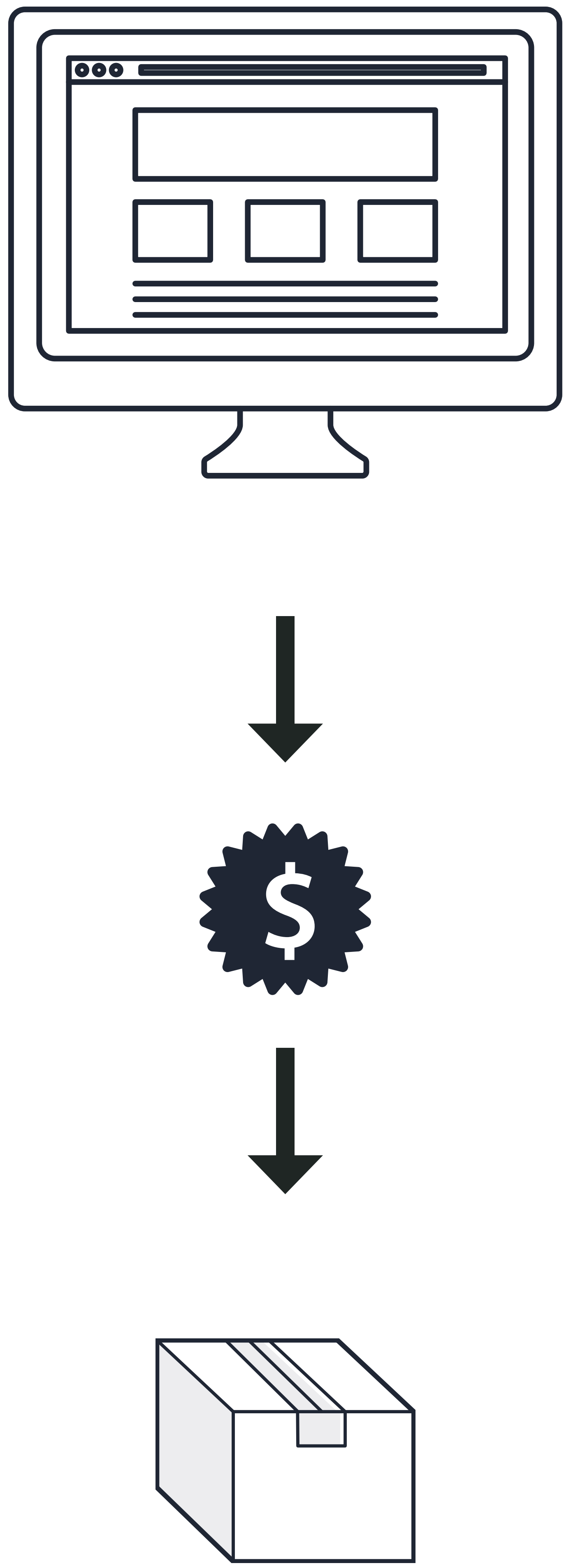
```
“order”: {
  “orderRef”: “ABC”,
  “state”: “cart”,
  “customerId”: 1,
  “lineItems”: [{
    “sku”: “basketball-shoe”,
    “qty”: 1,
    “price”: 100.00
  }],
  “shippingAddress”: {},
  “payment”: {},
  “orderTotal”: 100.00
}
```

A traditional system.



```
“order”: {
  “orderRef”: “ABC”,
  “state”: “cart”,
  “customerId”: 1,
  “lineItems”: [{
    “sku”: “basketball-shoe”,
    “qty”: 1,
    “price”: 100.00
  }],
  “shippingAddress”: {
    “city”: “Detroit”,
    …
  },
  “payment”: {},
  “orderTotal”: 100.00
}
```

A traditional system.



```
“order”: {
  “orderRef”: “ABC”,
  “state”: “order”,
  “customerId”: 1,
  “lineItems”: [{
    “sku”: “basketball-shoe”,
    “qty”: 1,
    “price”: 100.00
  }],
  “shippingAddress”: {
    “city”: “Detroit”,
    …
  },
  “payment”: {
    “creditCard”: “xxxx”,
    …
  },
  “orderTotal”: 100.00
}
```

An event-based system.

```
“activity”: {
  “action”: “add_line_items”,
  “orderRef”: “ABC”,
  “lineItems”: [{
    “sku”: “basketball-shoe”,
    “qty”: 1,
    “price”: 100.00
  }]
}
```

An event-based system.

```
“activity”: {
  “action”: “add_shipping_address”,
  “orderRef”: “ABC”,
  “shippingAddress”: {
    “city”: “Detroit”,
    …
  }
}
```

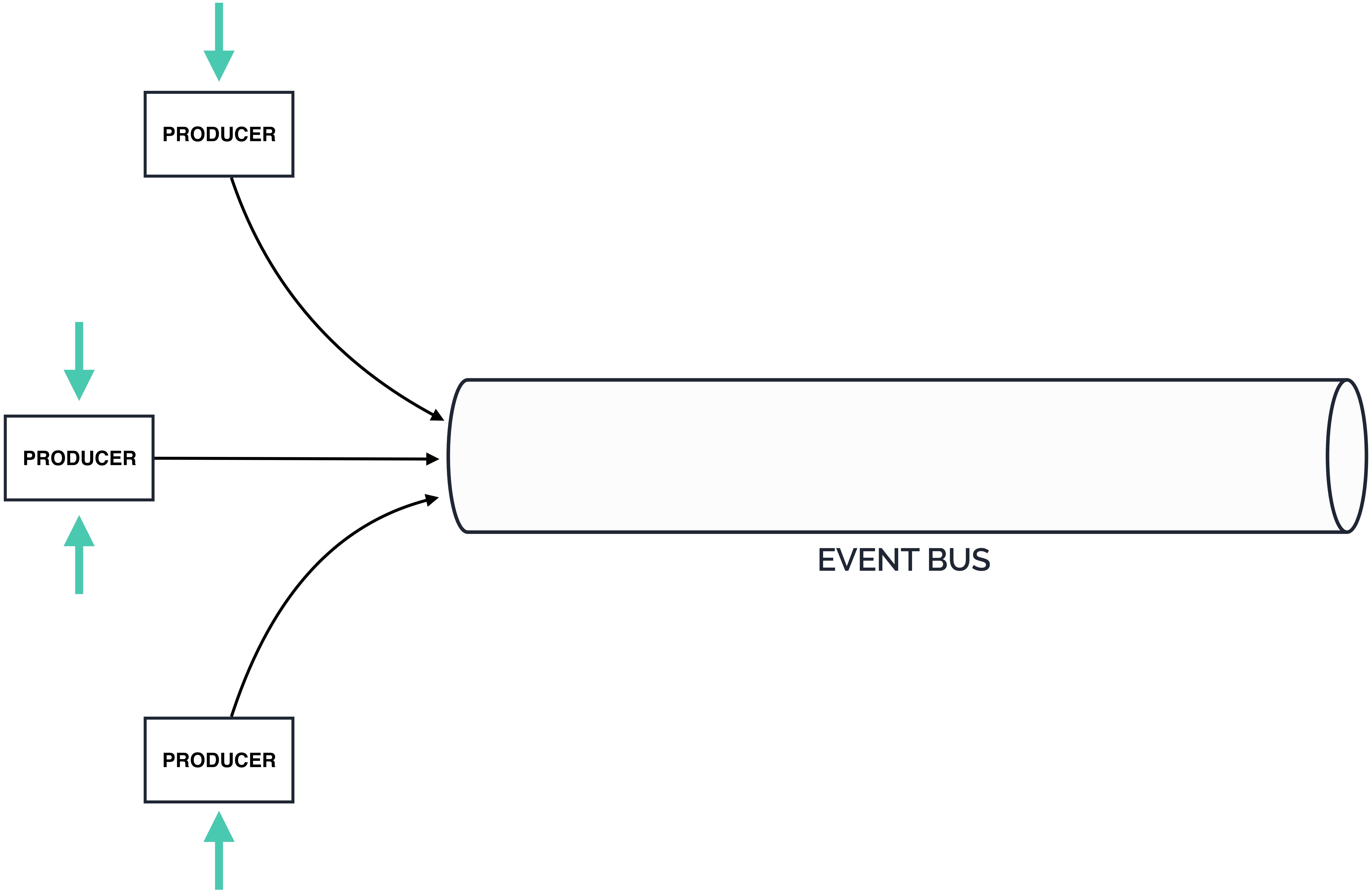
An event-based system.

```
“activity”: {
  “action”: “add_payment”,
  “orderRef”: “ABC”,
  “payment”: {
    “creditCard”: “xxxx”,
    …
  }
}
```

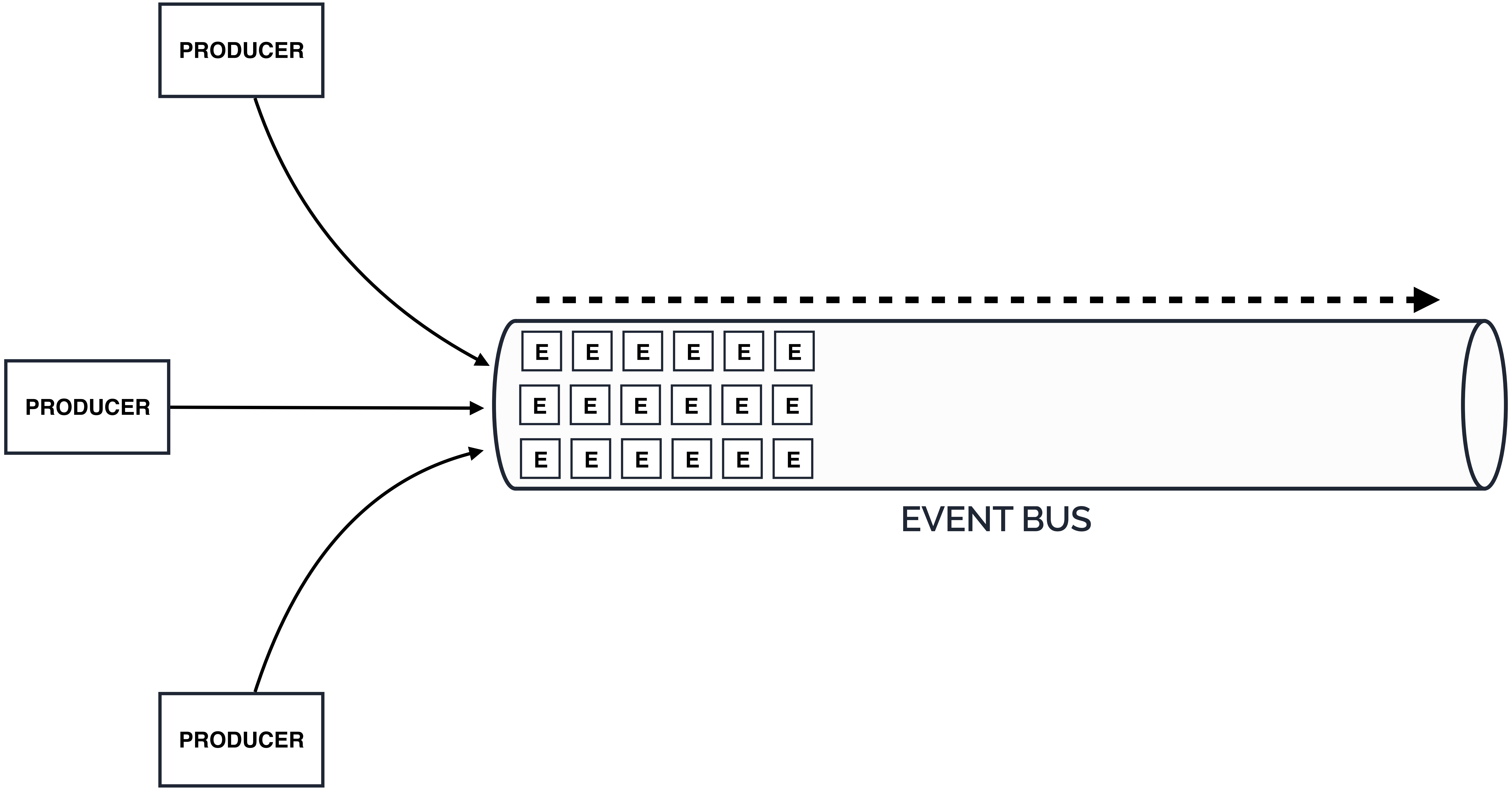
Why use event-based systems?

- It often maps better to the real world than traditional relational or document systems;
- It decouples the creation of an action from how it gets used;
- Useful for asynchronous and eventually-consistent tasks like caching;
- Aids in failure and retry-logic when building distributed systems;
- Mixing programming languages is a breeze;
- Audit logging for free!

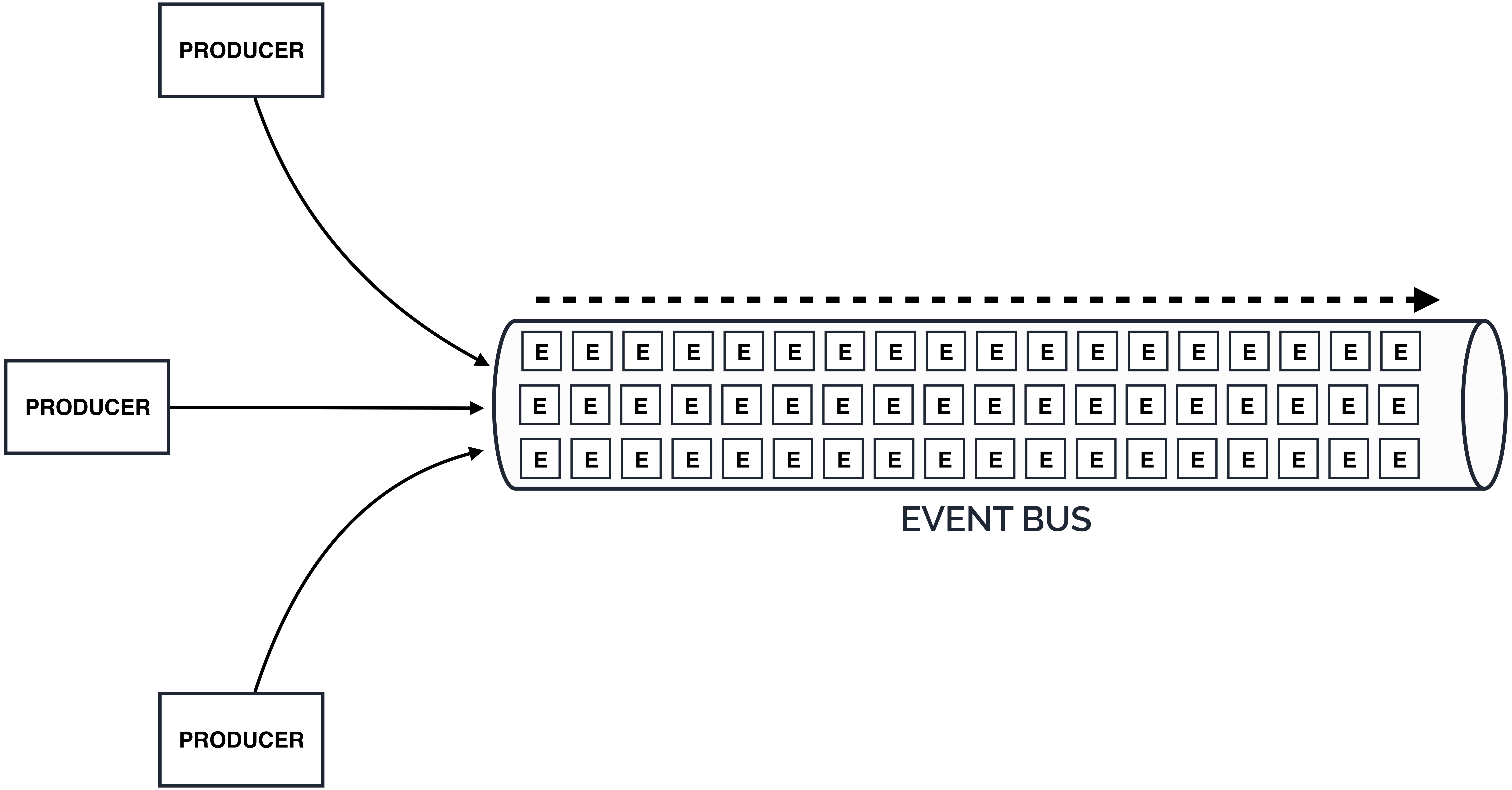
Event-based architecture overview.



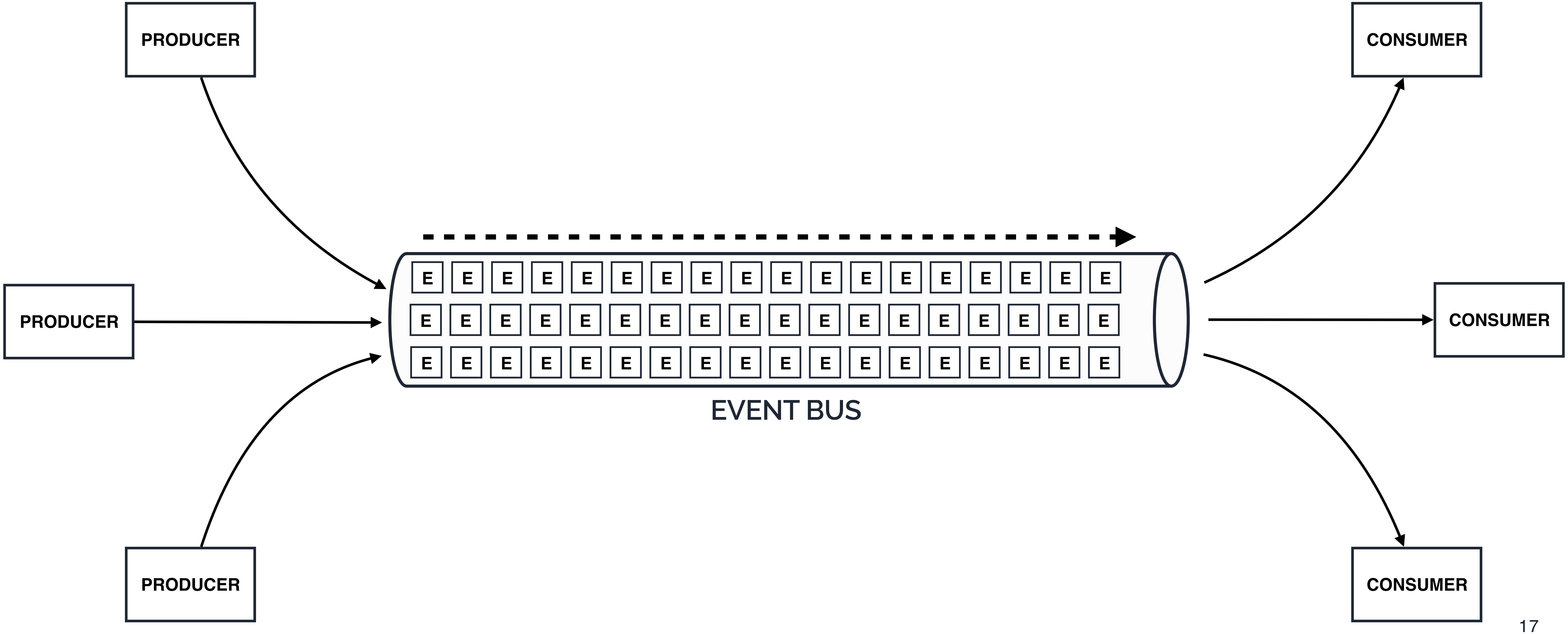
Event-based architecture overview.



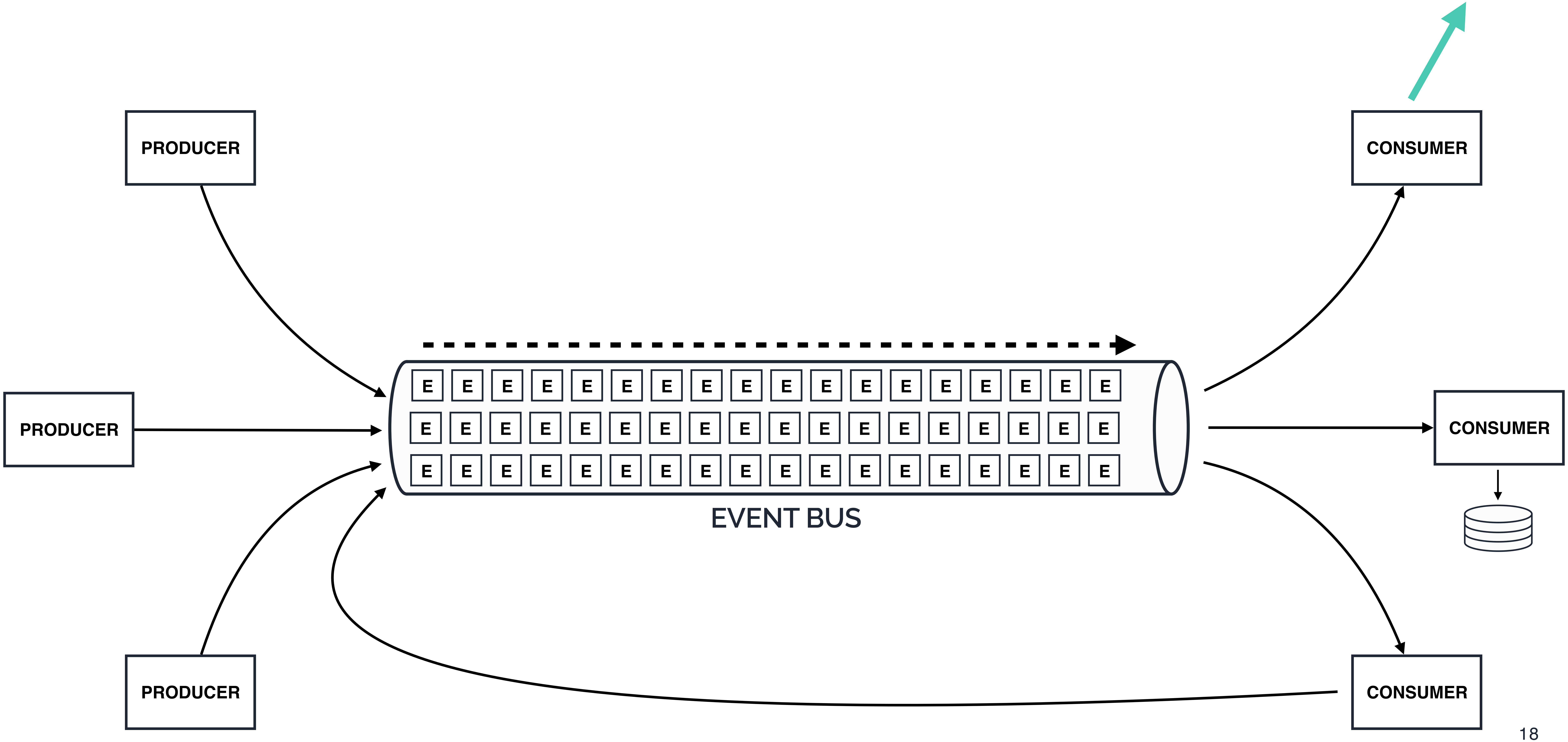
Event-based architecture overview.


PRODUCER
PRODUCER
PRODUCER
E E E E E E E E E E E E E E E E E E
E E E E E E E E E E E E E E E E E E
E E E E E E E E E E E E E E E E E E
EVENT BUS

Event-based architecture overview.



Event-based architecture overview.



Let's talk events.

THE BASICS

- What are event-driven architectures?
- Why are they useful?
- How do they work?

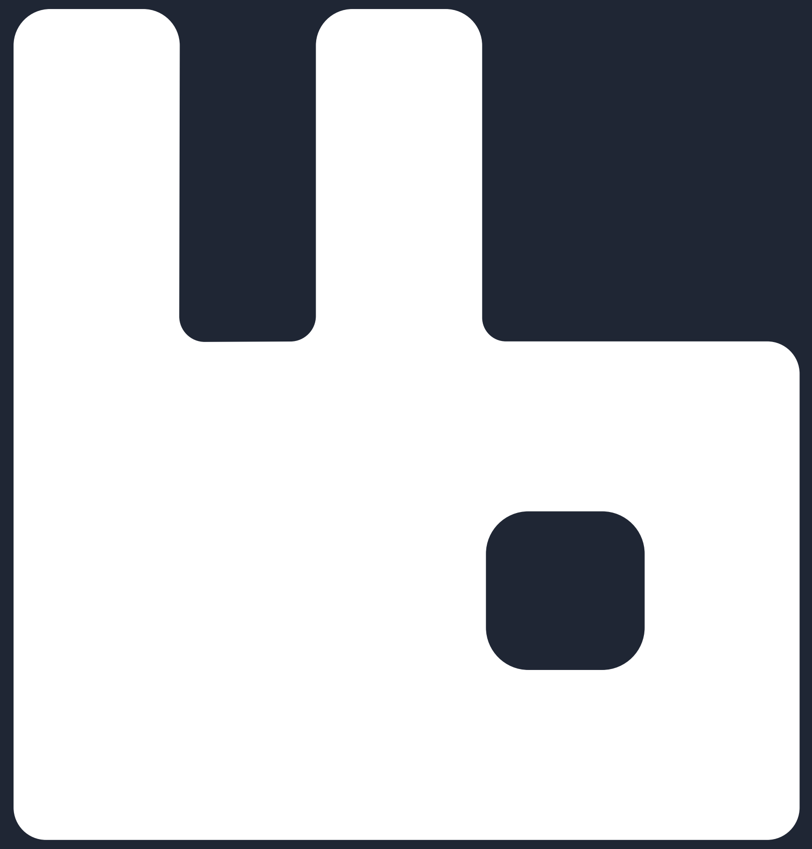
BUILDING EVENT-DRIVEN SYSTEMS

- Selecting an event bus
- Consumers and producers in Node
- Fun use cases

WRAPPING UP

- Lessons learned
- Serverless and the future
- Resources
- Questions

A couple options



Time to code.

github.com/jmataya/event-based-architectures-in-node

github.com/confluentinc/cp-docker-images

github.com/SOHU-Co/kafka-node

Let's talk events.

THE BASICS

- What are event-driven architectures?
- Why are they useful?
- How do they work?

BUILDING EVENT-DRIVEN SYSTEMS

- Selecting an event bus
- Consumers and producers in Node
- Fun use cases

WRAPPING UP

- Lessons learned
- Serverless and the future
- Resources
- Questions

Lessons learned (the hard way).

- Standardize your activity types and share definitions across producers and consumers (Avro)
- Consider fast-failure with automatic replay logic - see: foxcomm/metamorphosis
- Don’t be afraid to partition
- Keep consumers light and focused
- Produce events big enough to be acted upon effectively, but no bigger
- Be careful with services that both produce and consume
- Event-based and relational systems can be mixed!
- Understand your team’s DevOps capacity

To the future.

- Confluent continues to develop the ecosystem for Kafka - no longer only a first-class Java citizen
- DevOps is getting easier - big cloud providers (Amazon, Microsoft, Google) are rolling out native support, as are orchestration providers like Mesosphere
- Consider “serverless” architecture - you’ll trade off flexibility with way less DevOps effort.

Resources

- Getting started locally: https://docs.confluent.io/current/installation/docker/docs/quickstart.html
- Playing with Avro: https://github.com/foxcomm/metamorphosis
- Source Code: https://github.com/jmataya/event-based-architectures-in-node
- More examples: https://github.com/foxcomm/highlander

Questions?