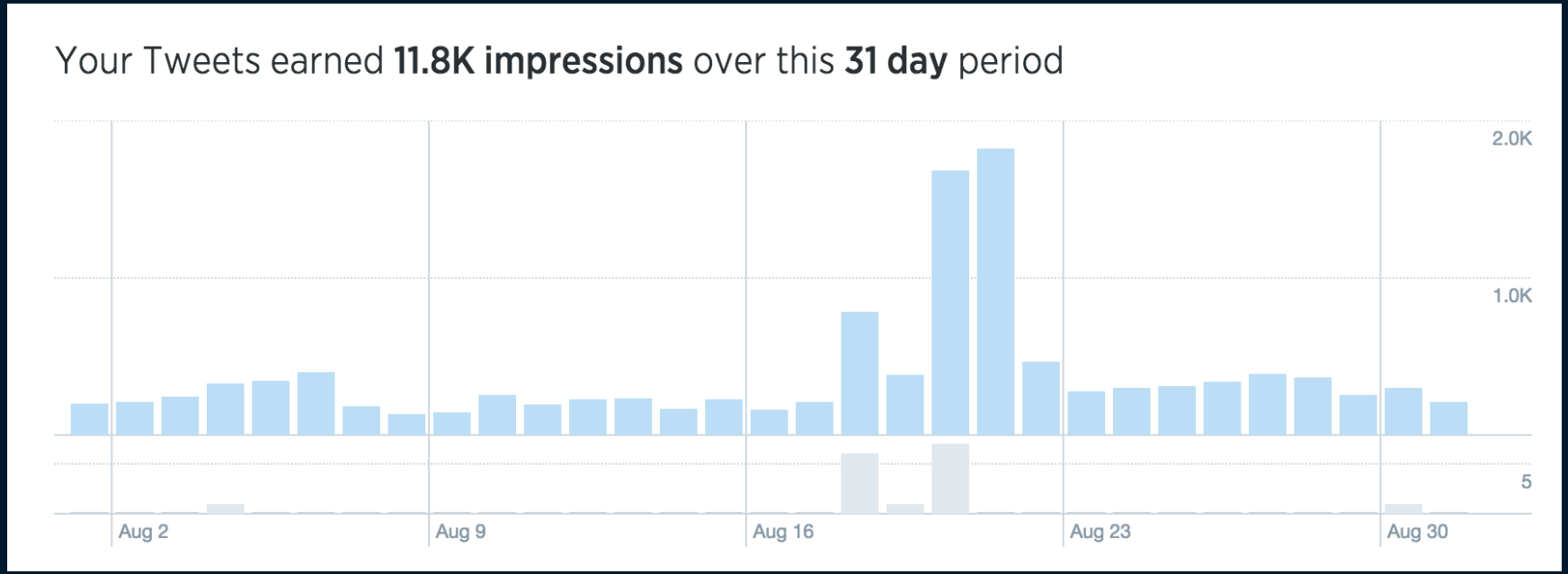
Building Real-Time Visualizations at Scale

Mike Barry @msb5014
Kevin Robinson @krob

Hello!

Hello!

analytics.twitter.com



Hello!


answers
No analysis required.
Finally, mobile app analytics you don't need to analyze.

Answers


iOS
Application 1
ios com.crashlytics.app1
Admin
Crashlytics, Inc.
7,384
ACTIVE USERS RIGHT NOW
DAILY ACTIVE USERS
211.0k
▲ 2.8%
DAILY NEW USERS
5.0k
▲ 4.2%
MONTHLY ACTIVE USERS
665.3k
▲ 3.6%
CRASH-FREE USERS
96.3%
▲ 5.0%
SESSIONS
633.6k
▲ 7.0%
PURCHASED ITEM
211.0k
▲ 2.8%
211.0k ▲ 2.8%
DAILY ACTIVE USERS
200K

Building Real-time Visualizations

Real-time

Actionable

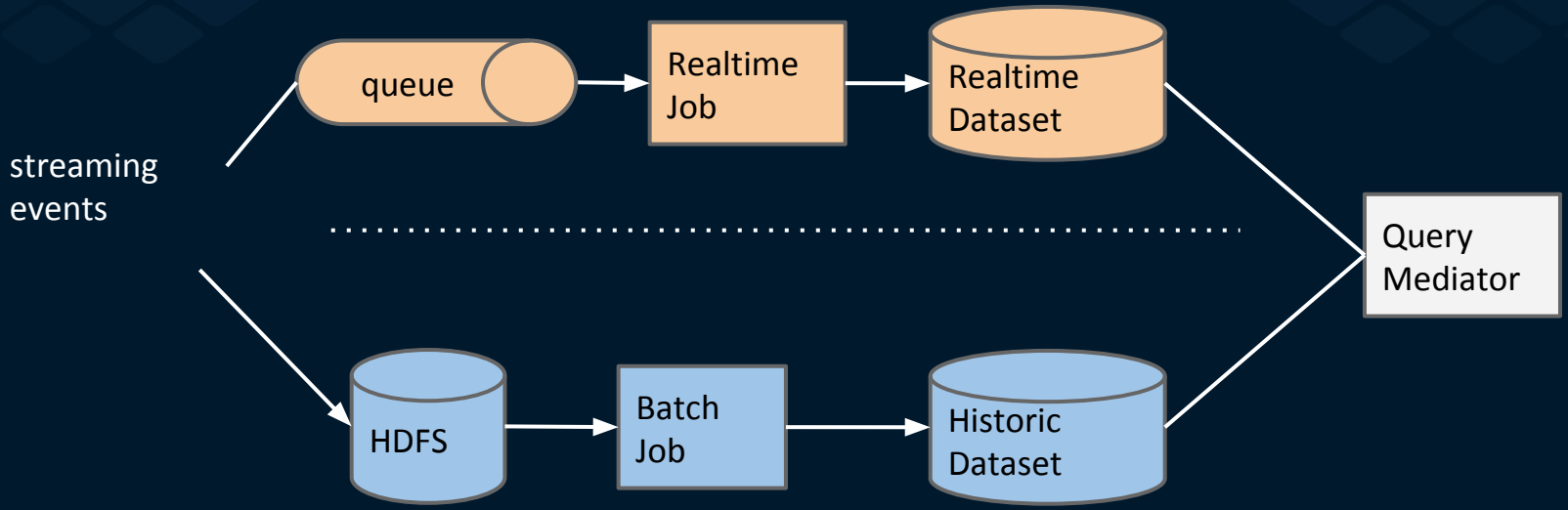
User-focused

Analytics at Twitter

Architecture

Higher-level abstractions

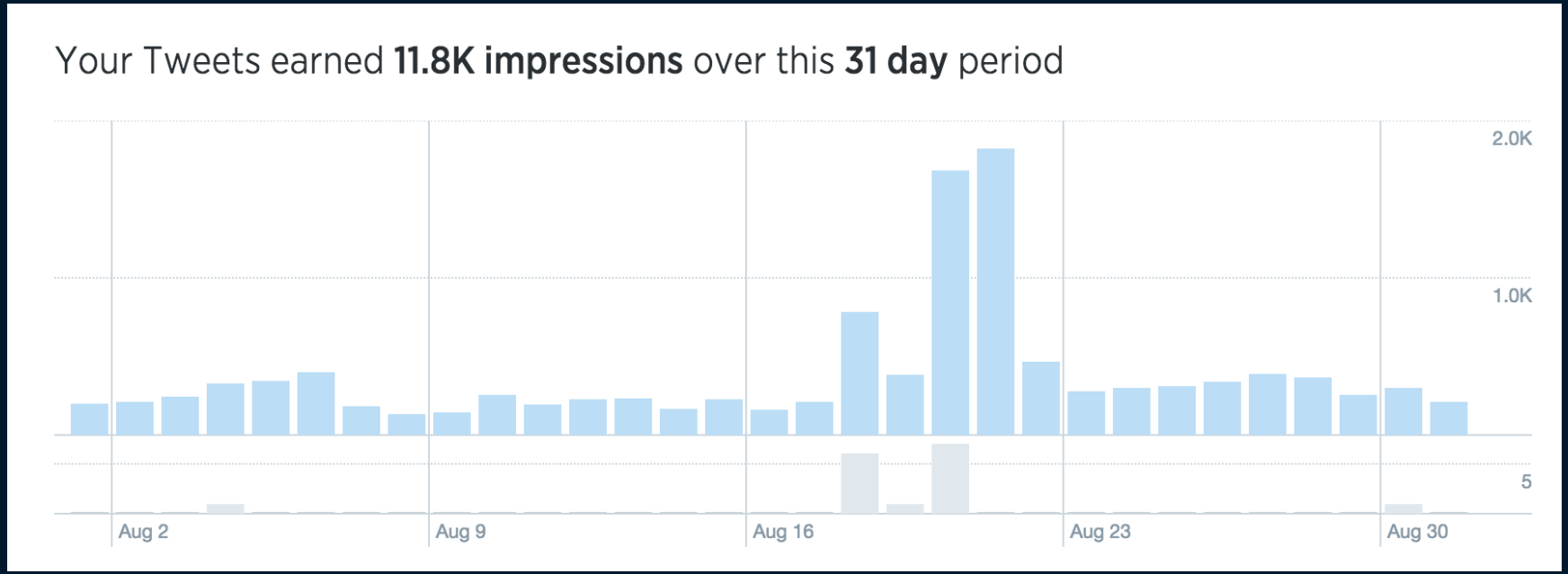
Human flexibility



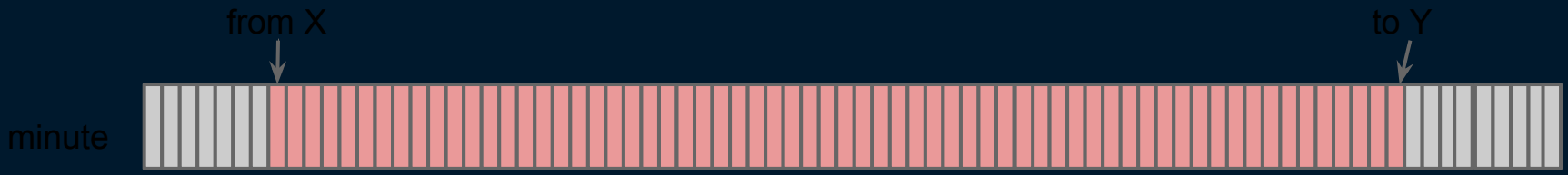
Typical Analytics Pipeline

Do more work on write so that reads are fast

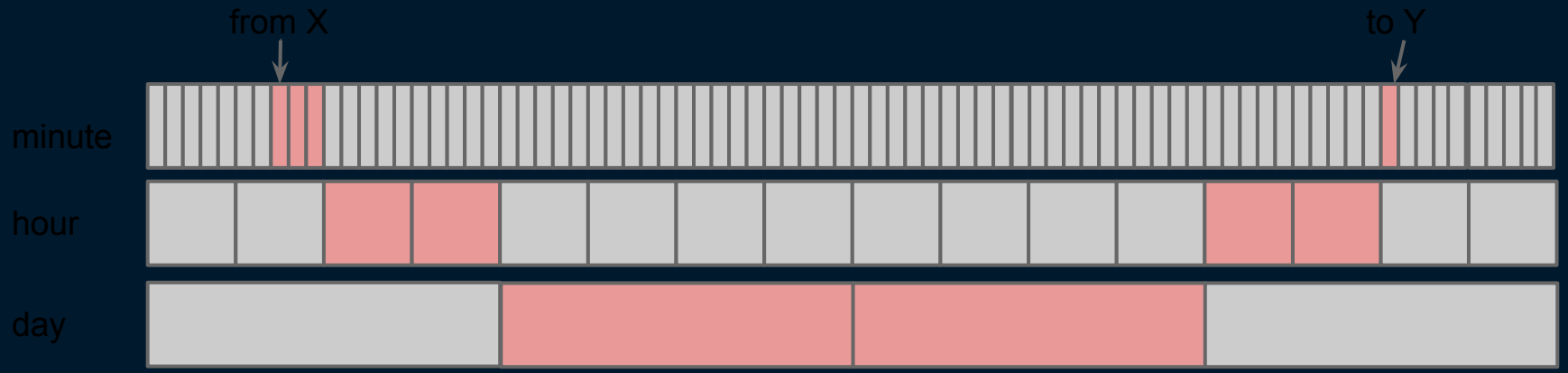
How many impressions from X to Y?



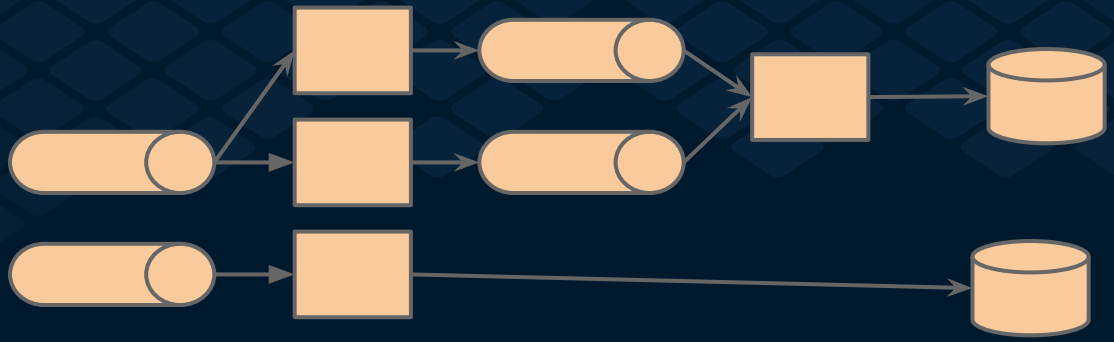
How many impressions from X to Y?

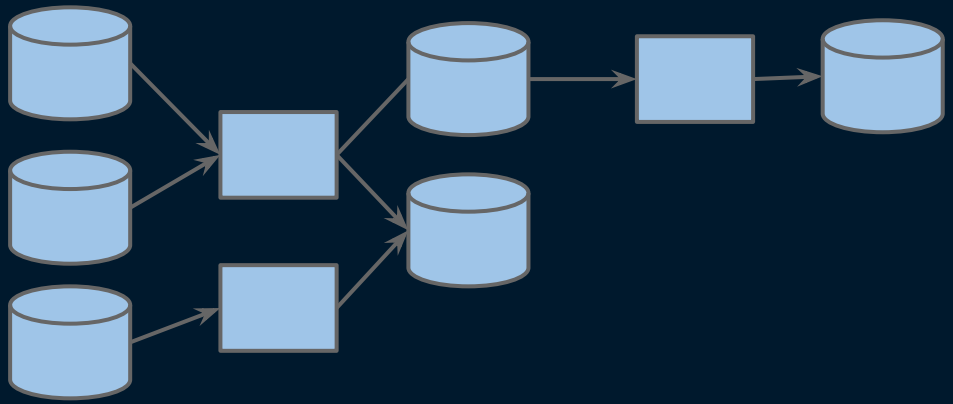


How many impressions from X to Y?



Abstractions





Scaling up

Communicate Fearlessly to Build Trust

Human Flexibility

Globally-available data

\+ Flexible individuals

\+ Hack weeks

= innovation

Analytics at Twitter

Architecture

Higher-level abstractions

Human flexibility


answers
No analysis required.
Finally, mobile app analytics you don't need to analyze.


iOS
Application 1
ios com.crashlytics.app1
Admin
Crashlytics, Inc.
7,384
ACTIVE USERS RIGHT NOW
DAILY ACTIVE USERS
211.0k
▲ 2.8%
DAILY NEW USERS
5.0k
▲ 4.2%
MONTHLY ACTIVE USERS
665.3k
▲ 3.6%
CRASH-FREE USERS
96.3%
▲ 5.0%
SESSIONS
633.6k
▲ 7.0%
PURCHASED ITEM
211.0k
▲ 2.8%
211.0k ▲ 2.8%
DAILY ACTIVE USERS
200K

answers events

answers events

Initial assumptions

Shortest path to usefulness

Real users and data change everything

answers events

Initial assumptions

Shortest path to usefulness

Real users and data change everything

No data!

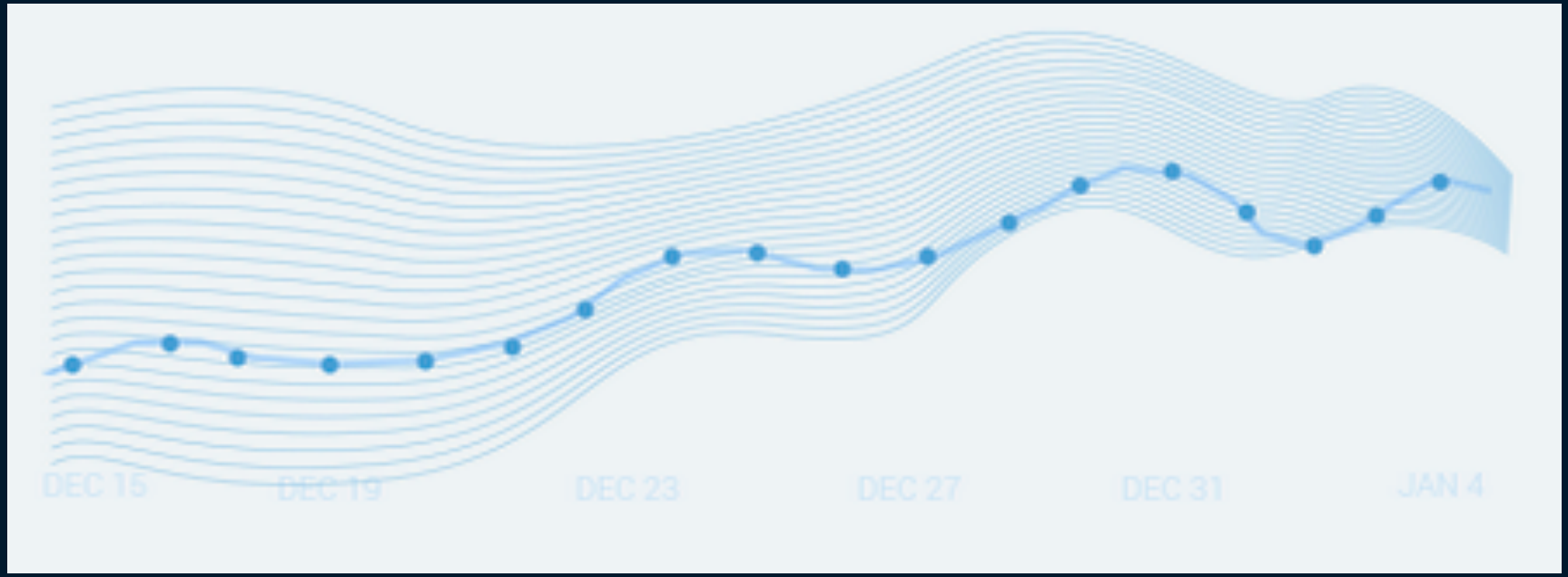
Existing data sources? nope

Predictable usage or distributions? nope

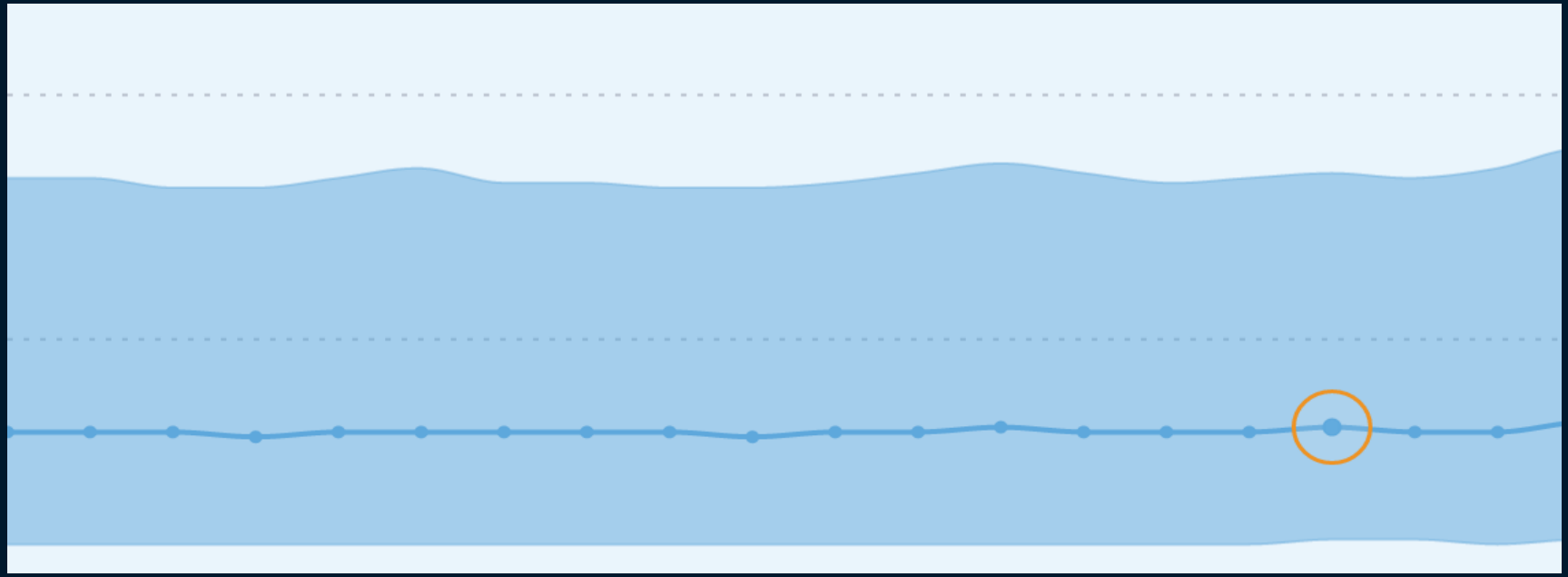
hmm...


THE DATA
MUST FLOW

Assumptions about data

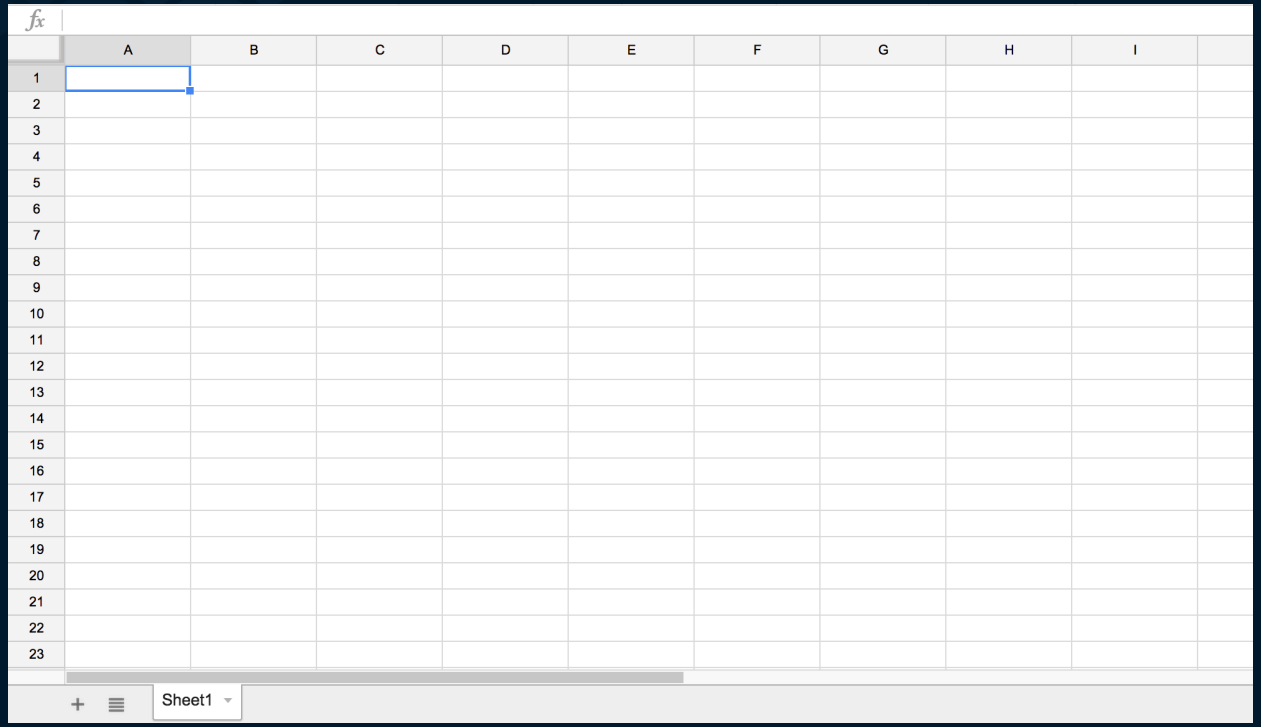


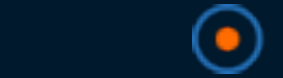
Assumptions about data



How can we make them explicit?

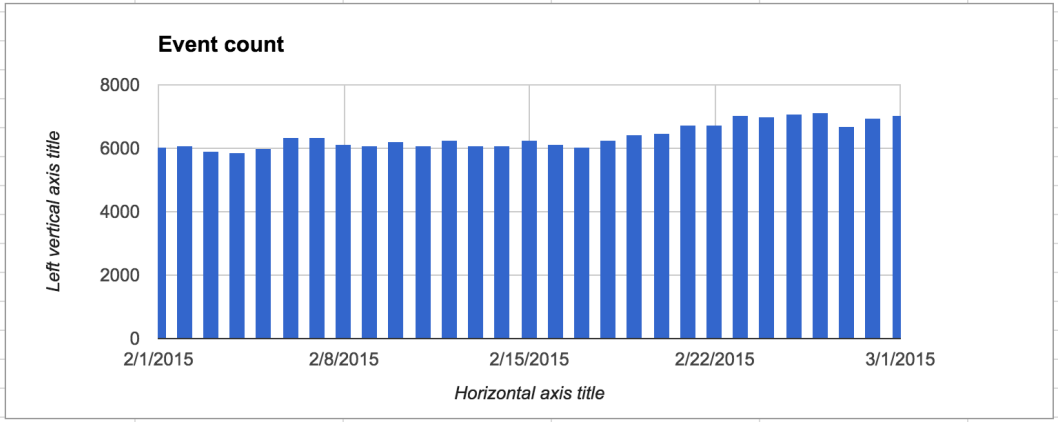
How can we make them explicit?





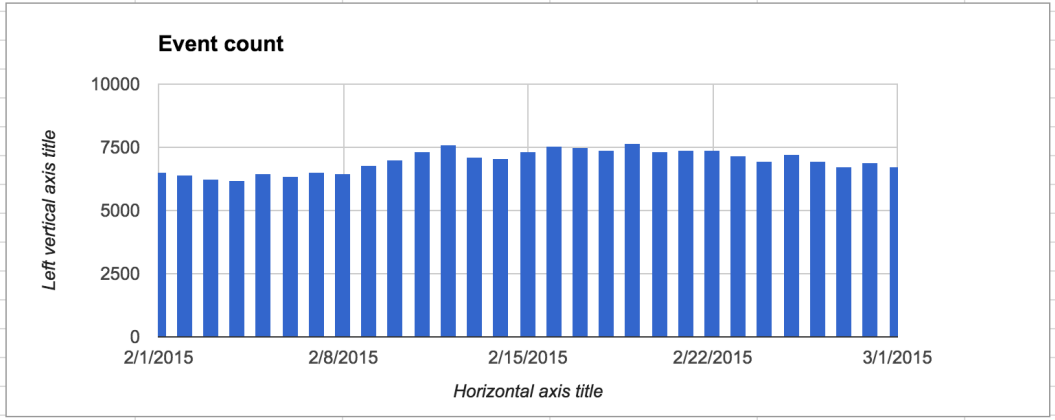
Excel prototype

	A	B	C	D	E	F
1						
2		DAU	% DAU did event	Unique users	Count per user	Count
3	Value	10978	35.36%	3882	1.8	7050
4	w/w change	5.06%	6.15%	11.52%	-6.28%	4.52%



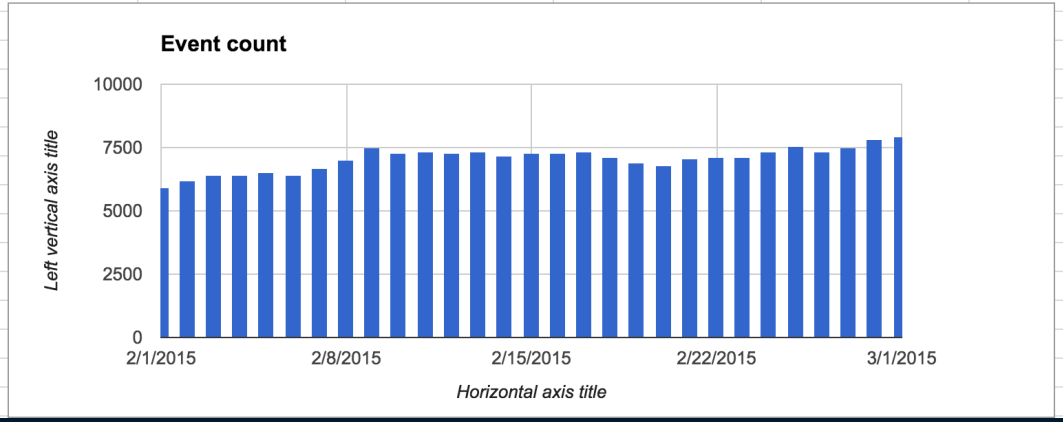
Excel prototype

	A	B	C	D	E	F
1						
2		DAU	% DAU did event	Unique users	Count per user	Count
3	Value	10652	28.37%	3022	2.2	6726
4	w/w change	-2.08%	-6.86%	-8.78%	-0.10%	-8.87%
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						



Excel prototype

	A	B	C	D	E	F
1						
2		DAU	% DAU did event	Unique users	Count per user	Count
3	Value	10233	30.45%	3116	2.5	7930
4	w/w change	8.22%	-2.00%	6.06%	5.06%	11.44%
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						



answers events

Initial assumptions about data

Shortest path to usefulness

Real users and data change everything

answers events

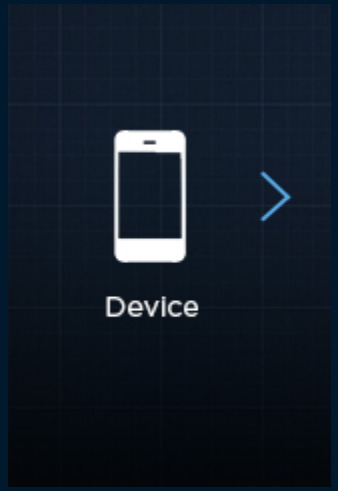
Initial assumptions about data

Shortest path to usefulness

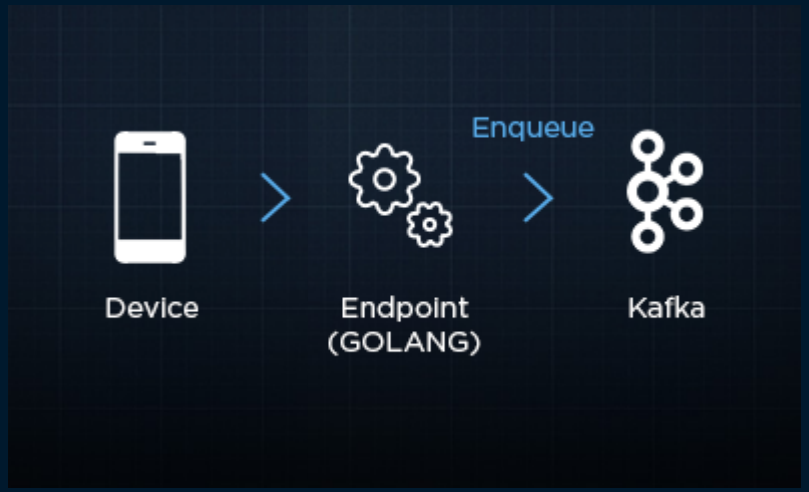
Real users and data change everything

Let’s build!

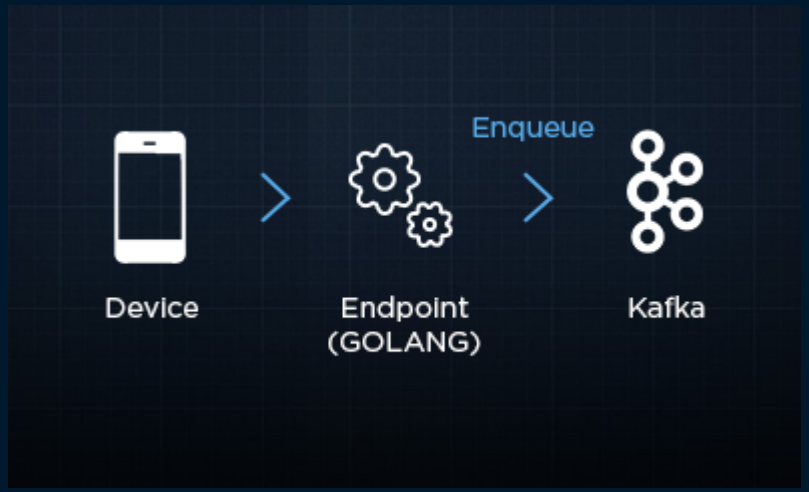
Let’s build!



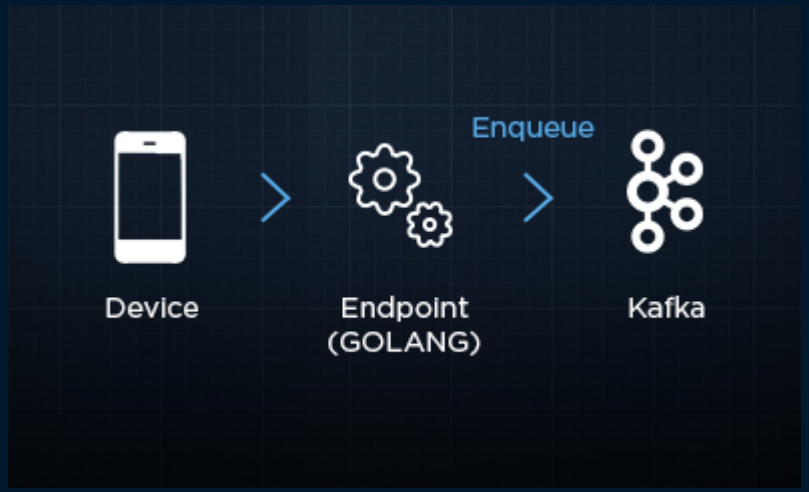
Let’s build!



Real-time computation



Real-time computation

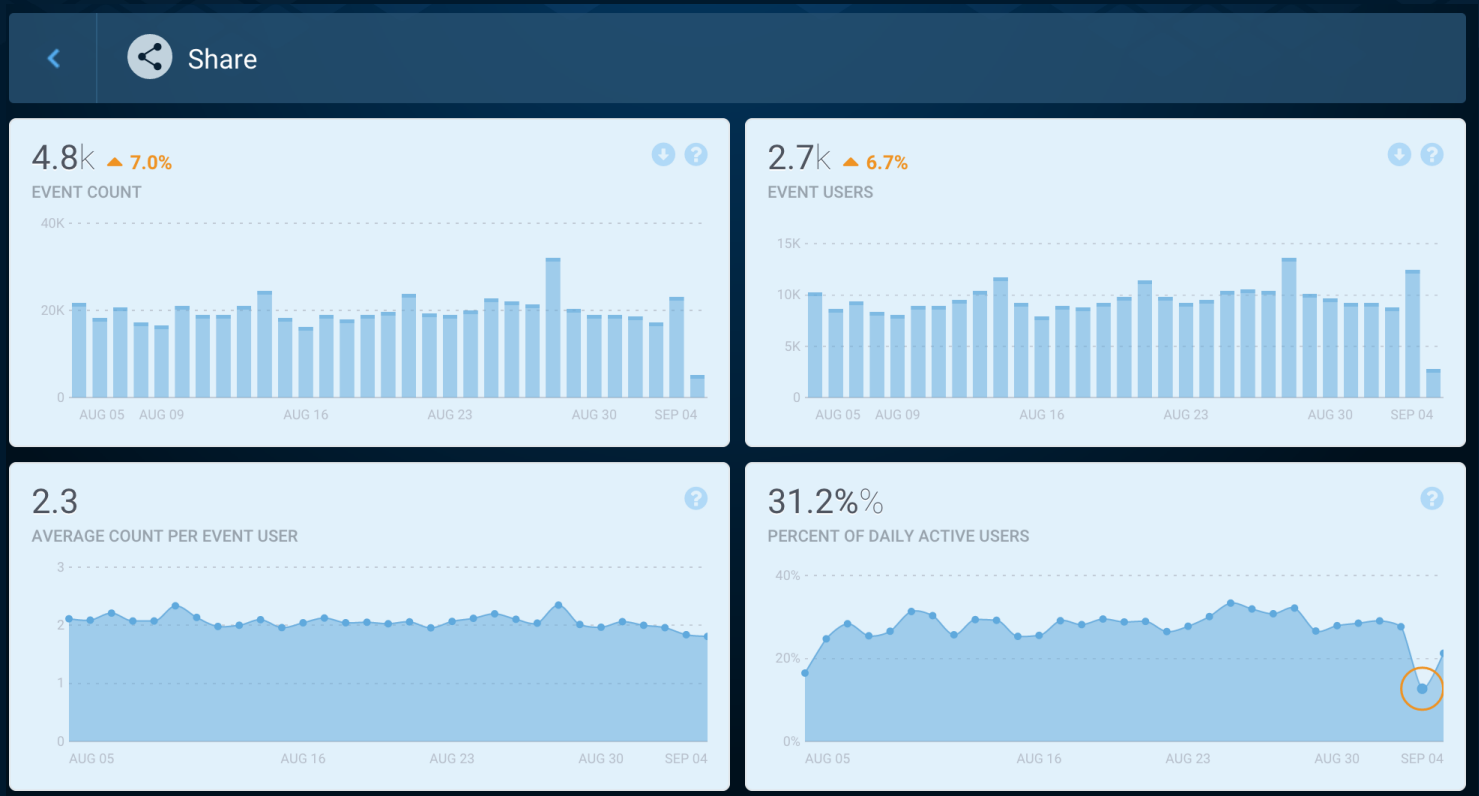


Let’s build: Prototype feature

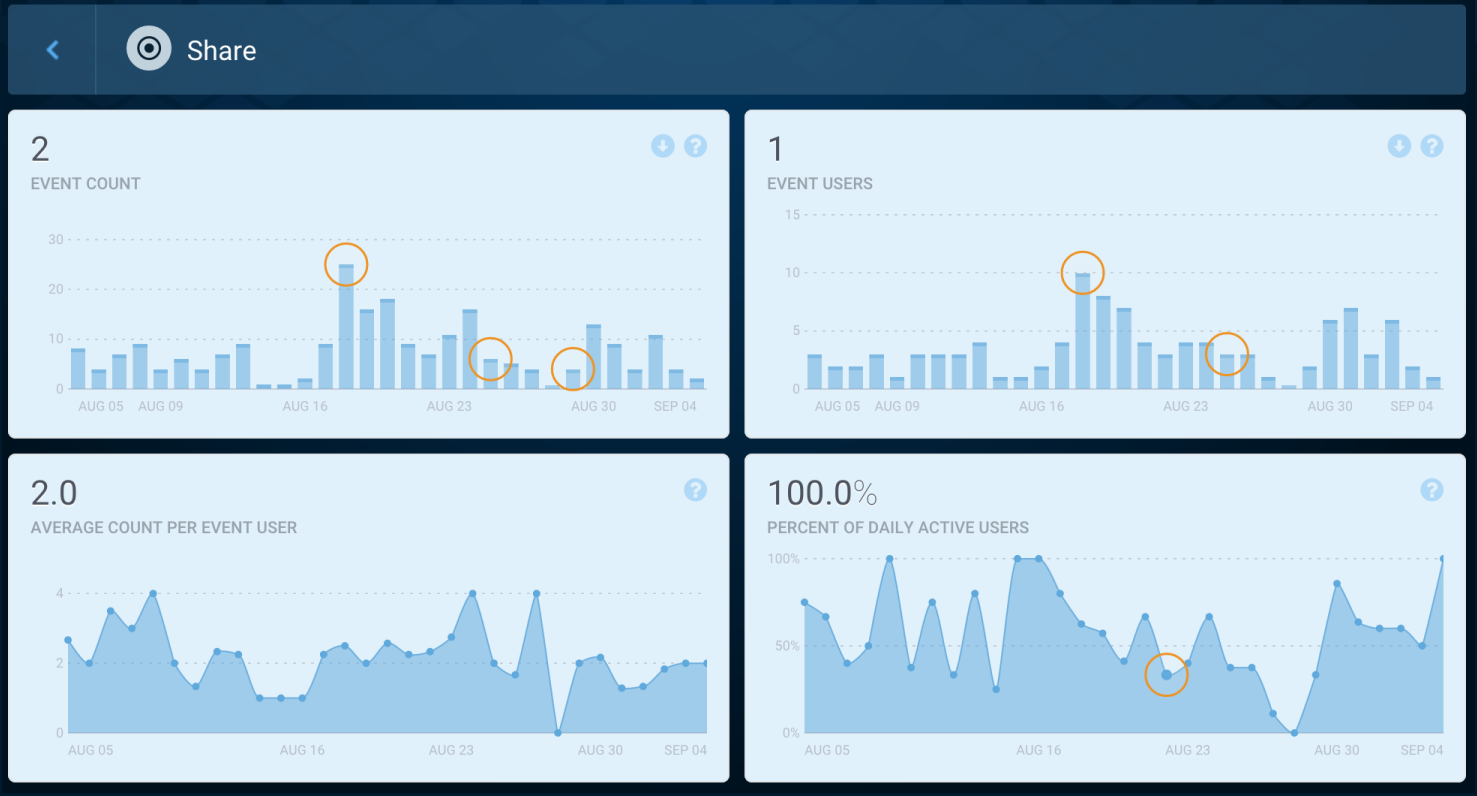
Prototype feature

Prototype feature

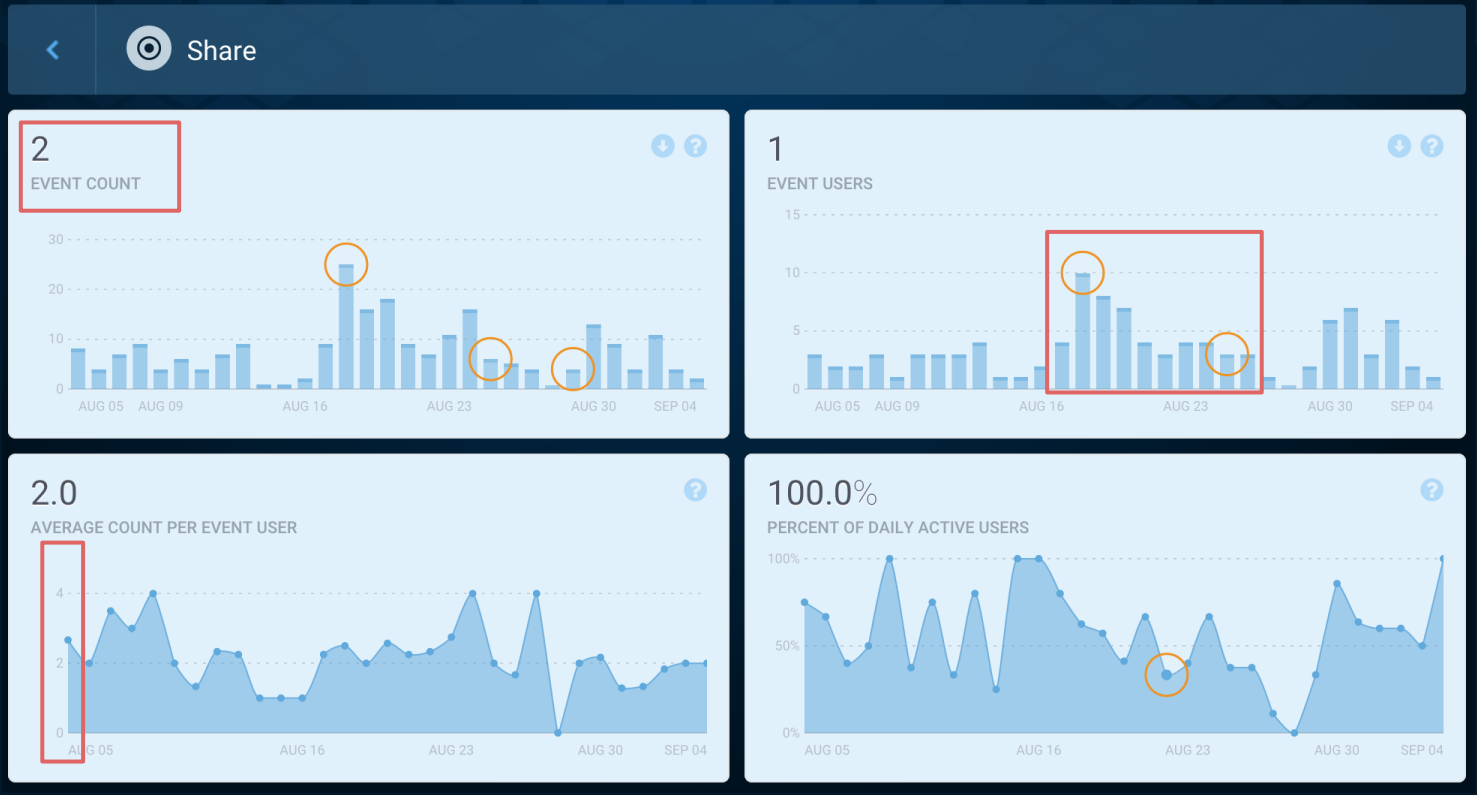
Prototype feature



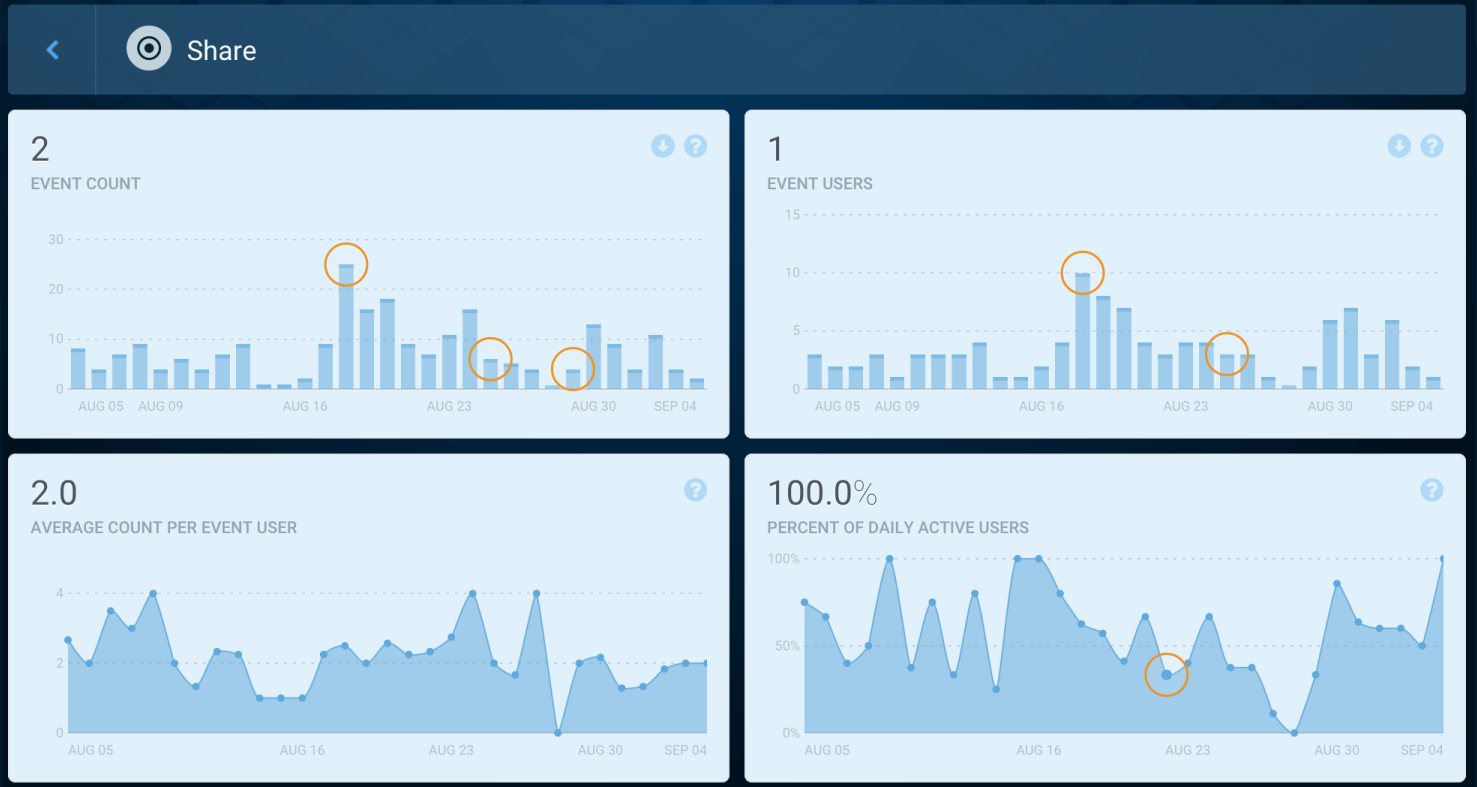
Prototype feature



Prototype feature



Production feature



More fault-tolerance

More fault-tolerance

Local Cascading jobs

Subsets or samples of real data

In-memory tests

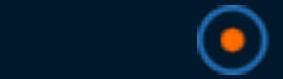
More fault-tolerance

Local Cascading jobs

Subsets or samples of real data

In-memory tests

More data only a command line away



Ready for real users!

answers events

Initial assumptions about data

Shortest path to usefulness

Real users and data change everything

answers events

Initial assumptions about data

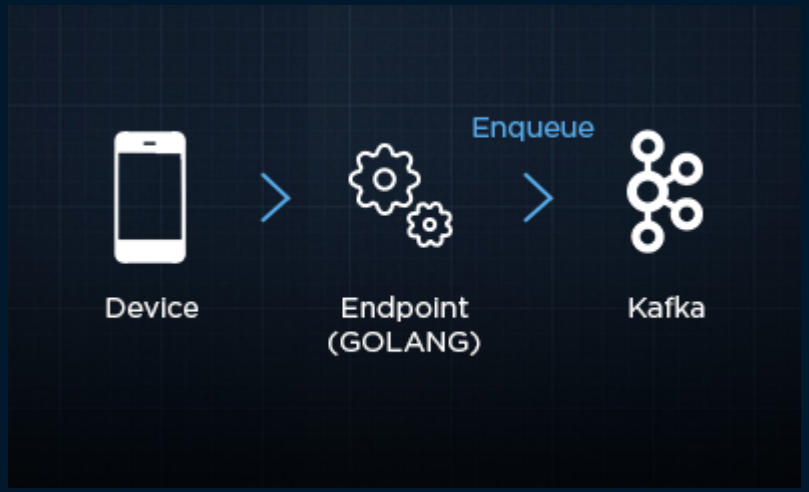
Shortest path to usefulness

Real users and data change everything

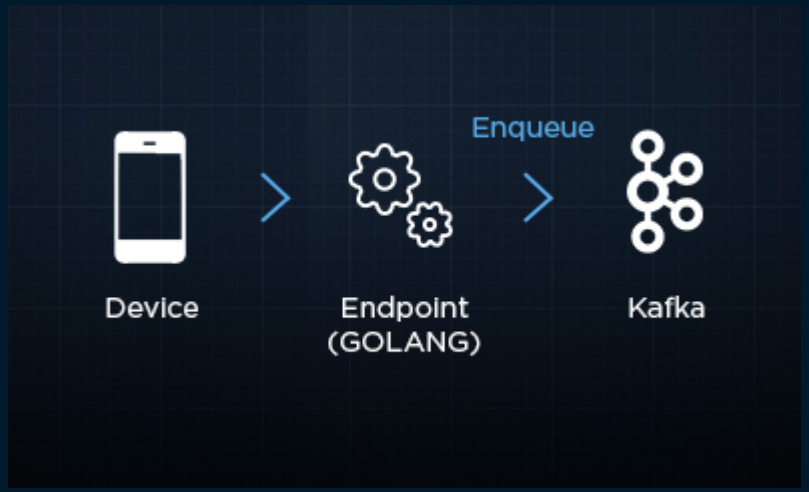
High-touch feedback

Exploring the data

Real-time prototyping

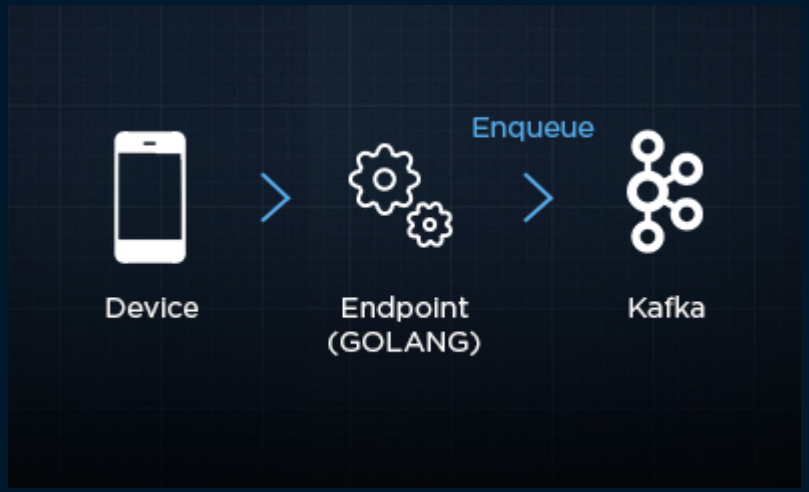


Real-time prototyping

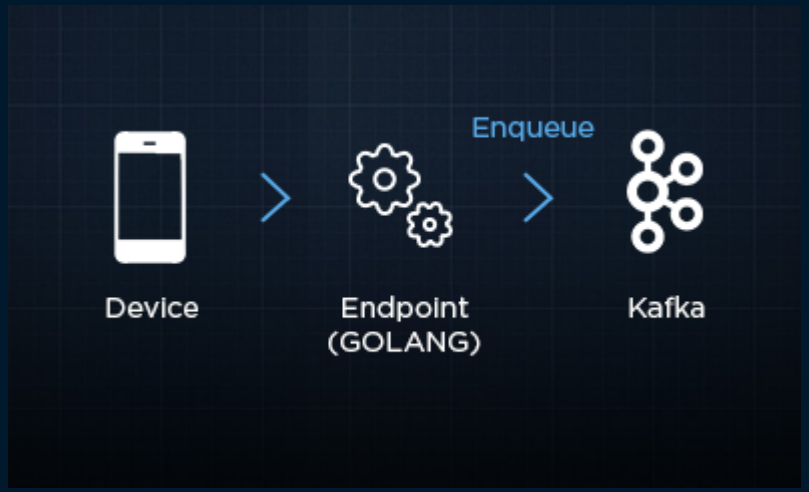


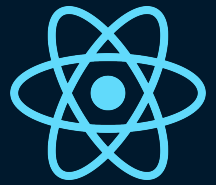
kafka logs gif

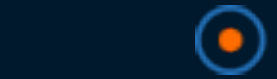
Real-time prototyping



Real-time prototyping







Real-time prototyping

```
CanvasTimeline = React.createClass
  displayName: 'CanvasTimeline'

  render: ->
    dom.canvas {
      ref: 'canvas'
      width: @props.width
      height: @props.height
      onMouseMove: @whenMouseMoveInCanvas
      onClick: @whenCanvasClicked
    }

  componentDidMount: ->
    @ctx = @refs.canvas.getDOMNode().getContext '2d'
    @redrawCanvas()

  redrawCanvas: ->
    {x, y, binHeight, minBarWidth, nowMs} = @props.chartParams

    @ctx.clearRect 0, 0, @props.width, @props.height
    @props.filteredData.forEach (d) =>
      @ctx.fillStyle = @props.blockColor d
      @ctx.fillRect [
        (x(d.startTime) + @props.xOffset) *
        y(d) * @props.yScale
        Math.max(minBarWidth, x(d.endTime) -
        binHeight * @props.yScale
      ]...

    nowX = (x(nowMs) + @props.xOffset) * @pr
    @ctx.strokeStyle = '#ccc'
    @ctx.beginPath()
    @ctx.moveTo(nowX, y.range()[0])
    @ctx.lineTo(nowX, y.range()[1])
    @ctx.stroke()
```

sublime samples

Real-time prototyping

```
CanvasTimeline = React.createClass
  displayName: 'CanvasTimeline'

  render: ->
    dom.canvas {
      ref: 'canvas'
      width: @props.width
      height: @props.height
      onMouseMove: @whenMouseMoveInCanvas
      onClick: @whenCanvasClicked
    }

  componentDidMount: ->
    @ctx = @refs.canvas.getDOMNode().getContext '2d'
    @redrawCanvas()

  redrawCanvas: ->
    {x, y, binHeight, minBarWidth, nowMs} = @props.chartParams

    @ctx.clearRect 0, 0, @props.width, @props.height
    @props.filteredData.forEach (d) =>
      @ctx.fillStyle = @props.blockColor d
      @ctx.fillRect [
        (x(d.startTime) + @props.xOffset) *
        y(d) * @props.yScale
        Math.max(minBarWidth, x(d.endTime) -
        binHeight * @props.yScale
      ]...

    nowX = (x(nowMs) + @props.xOffset) * @pr
    @ctx.strokeStyle = '#ccc'
    @ctx.beginPath()
    @ctx.moveTo(nowX, y.range()[0])
    @ctx.lineTo(nowX, y.range()[1])
    @ctx.stroke()
```

sublime samples

Real-time prototyping

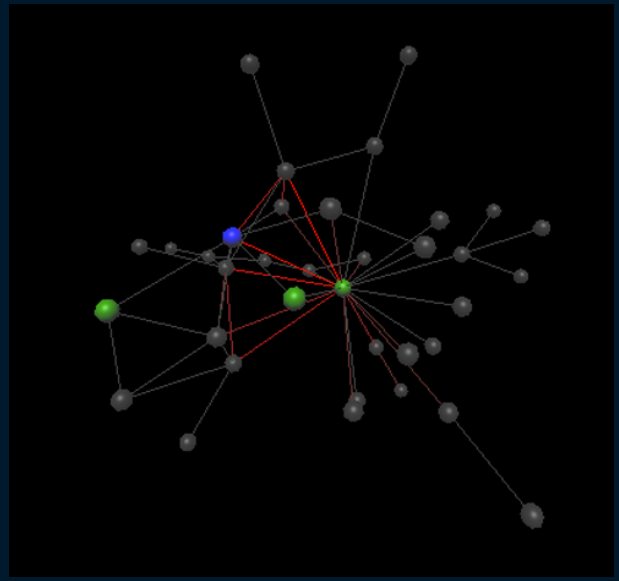
```
CanvasTimeline = React.createClass
  displayName: 'CanvasTimeline'

  render: ->
    dom.canvas {
      ref: 'canvas'
      width: @props.width
      height: @props.height
      onMouseMove: @whenMouseMoveInCanvas
      onClick: @whenCanvasClicked
    }

  componentDidMount: ->
    @ctx = @refs.canvas.getDOMNode().getContext '2d'
    @redrawCanvas()

  redrawCanvas: ->
    {x, y, binHeight, minBarWidth, nowMs} = @props.chartParams

    @ctx.clearRect 0, 0, @props.width, @props.height
    @props.filteredData.forEach (d) =>
      @ctx.fillStyle = @props.blockColor d
      @ctx.fillRect [
        (x(d.startTime) + @props.xOffset) *
        y(d) * @props.yScale
        Math.max(minBarWidth, x(d.endTime) -
        binHeight * @props.yScale
      ]...

    nowX = (x(nowMs) + @props.xOffset) * @pr
    @ctx.strokeStyle = '#ccc'
    @ctx.beginPath()
    @ctx.moveTo(nowX, y.range()[0])
    @ctx.lineTo(nowX, y.range()[1])
    @ctx.stroke()
```

sublime samples

Real-time prototyping

```
CanvasTimeline = React.createClass
  displayName: 'CanvasTimeline'

  render: ->
    dom.canvas {
      ref: 'canvas'
      width: @props.width
      height: @props.height
      onMouseMove: @whenMouseMoveInCanvas
      onClick: @whenCanvasClicked
    }

  componentDidMount: ->
    @ctx = @refs.canvas.getDOMNode().getContext '2d'
    @redrawCanvas()

  redrawCanvas: ->
    {x, y, binHeight, minBarWidth, nowMs} = @props.chartParams

    @ctx.clearRect 0, 0, @props.width, @props.height
    @props.filteredData.forEach (d) =>
      @ctx.fillStyle = @props.blockColor d
      @ctx.fillRect [
        (x(d.startTime) + @props.xOffset) *
        y(d) * @props.yScale
        Math.max(minBarWidth, x(d.endTime) -
        binHeight * @props.yScale
      ]...

    nowX = (x(nowMs) + @props.xOffset) * @pr
    @ctx.strokeStyle = '#ccc'
    @ctx.beginPath()
    @ctx.moveTo(nowX, y.range()[0])
    @ctx.lineTo(nowX, y.range()[1])
    @ctx.stroke()
```

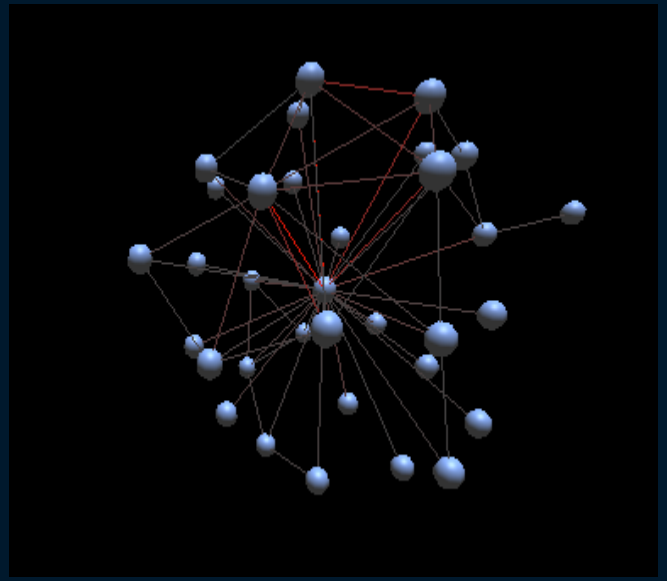
sublime samples



Real-time prototyping

```
CanvasTimeline = React.createClass
  displayName: 'CanvasTimeline'

  render: ->
    dom.canvas {
      ref: 'canvas'
      width: @props.width
      height: @props.height
      onMouseMove: @whenMouseMoveInCanvas
      onClick: @whenCanvasClicked
    }

  componentDidMount: ->
    @ctx = @refs.canvas.getDOMNode().getContext '2d'
    @redrawCanvas()

  redrawCanvas: ->
    {x, y, binHeight, minBarWidth, nowMs} = @props.chartParams

    @ctx.clearRect 0, 0, @props.width, @props.height
    @props.filteredData.forEach (d) =>
      @ctx.fillStyle = @props.blockColor d
      @ctx.fillRect [
        (x(d.startTime) + @props.xOffset) *
        y(d) * @props.yScale
        Math.max(minBarWidth, x(d.endTime) -
        binHeight * @props.yScale
      ]...

    nowX = (x(nowMs) + @props.xOffset) * @pr
    @ctx.strokeStyle = '#ccc'
    @ctx.beginPath()
    @ctx.moveTo(nowX, y.range()[0])
    @ctx.lineTo(nowX, y.range()[1])
    @ctx.stroke()
```

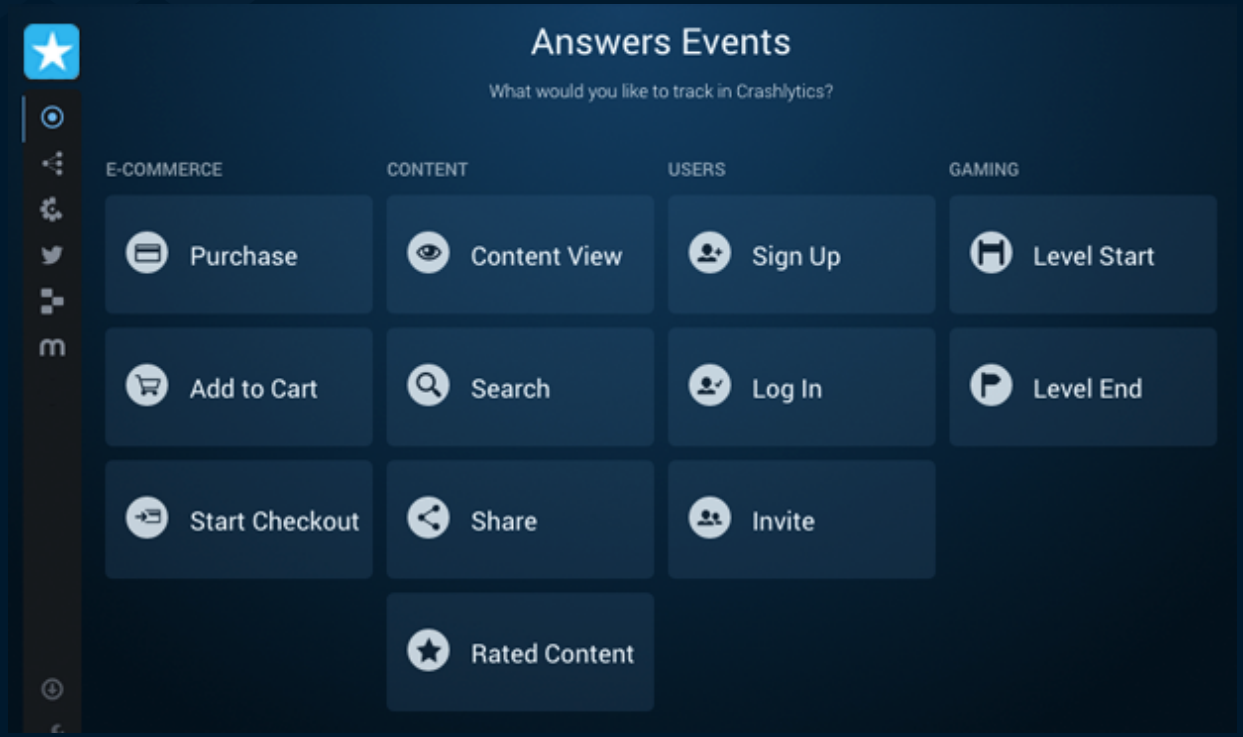
sublime samples



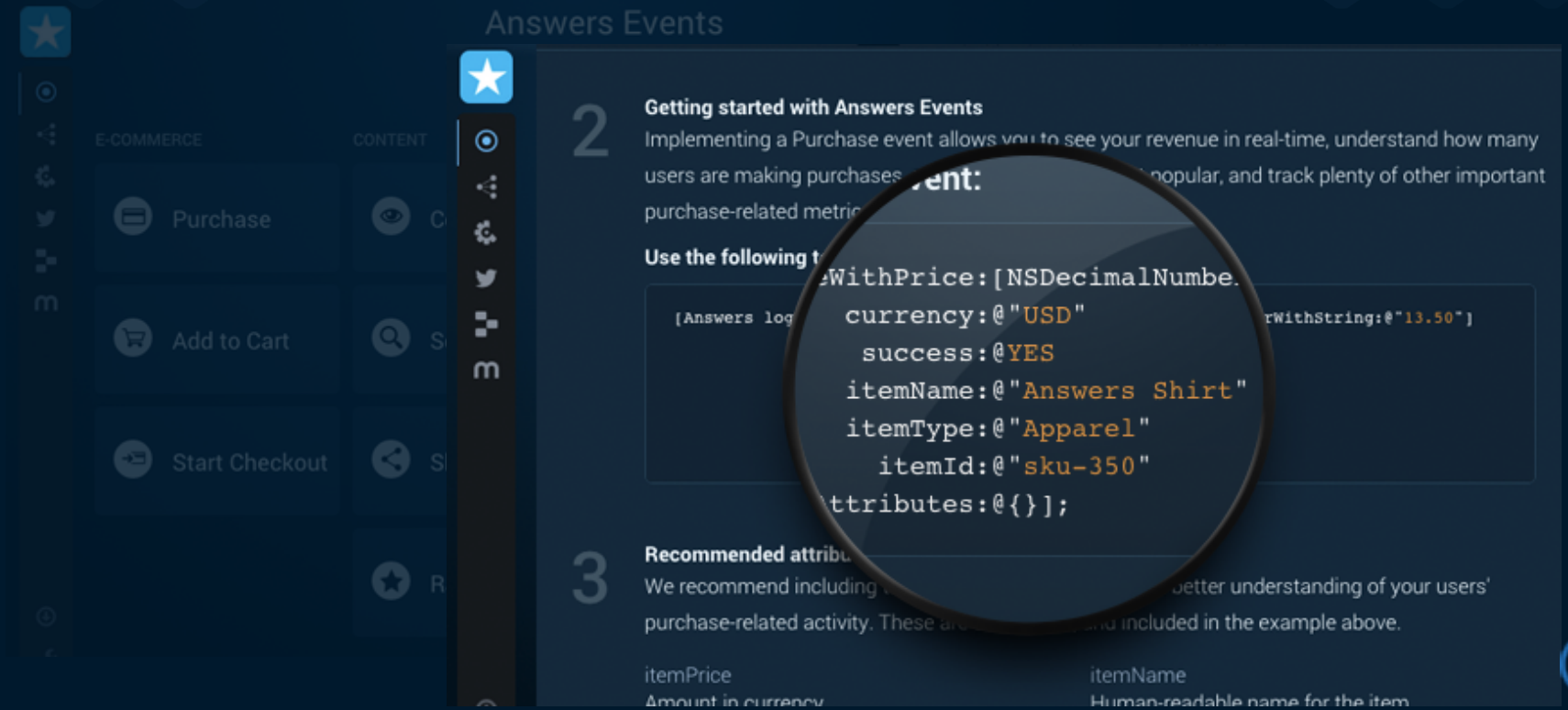
What’s the TL;DR?

Answers Events

Opinionated



Opinionated



TL;DR


TODAY'S TOP EVENTS
Search all events by name
4 EVENTS
Purchase
119.9k
4.0%
EVENT COUNT
112.3k
2.0%
EVENT USERS
Log In
75.1k
EVENT COUNT
28.2k
1.0%
EVENT USERS
Share
142.0k
3.0%
EVENT COUNT
32.0k
4.0%
EVENT USERS
Messages read
178.1k
12.0%
EVENT COUNT
79.7k
8.0%
EVENT USERS
See our best practices for setting up custom events

TL;DR


HelloWorld
com.helloworld
Messages read
178.1k
12.0%
EVENT COUNT
130.5k
29.0% below expected
200K
100K
0
JAN 26
SUN FEB 02
FEB 09
FEB 16
FEB 25
70.0k
8.0%
EVENT USERS
100K
50K
0
JAN 26
FEB 02
FEB 09
FEB
3.0
AVERAGE COUNT PER EVENT USER
23.8%
PERCENT OF DAILY ACTIVE USERS
30%

TL;DR


Purchase
$928.71
DAILY REVENUE
0,000
0,000
$0
AUG 05
AUG 09
AUG 16
AUG 23
AUG 30
SEP 04


INTRODUCING
answers
events

answers events

Initial assumptions about data

Shortest path to usefulness

Real users and data change everything

https://blog.twitter.com/2015/handling-five-billion-sessions-a-day-in-real-time

Conclusion

Lambda architecture

Opening everything enables re-use

Higher-level abstractions

Full-stack iteration

answers

Thanks!