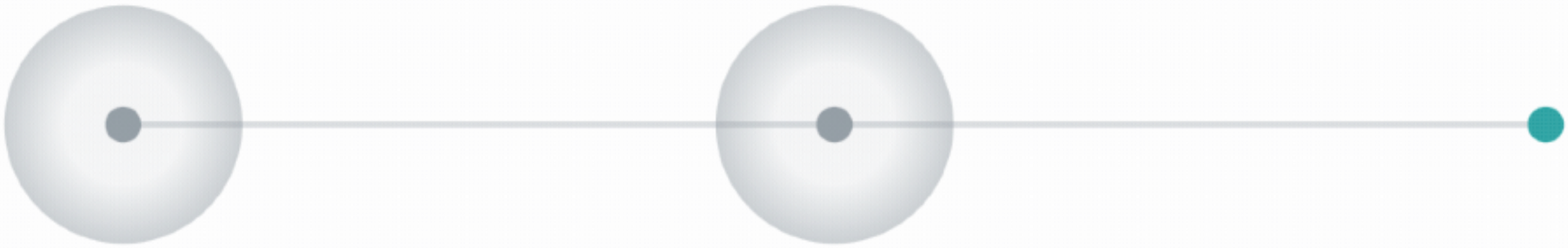
Customer



1
Active carts

1
Checking out

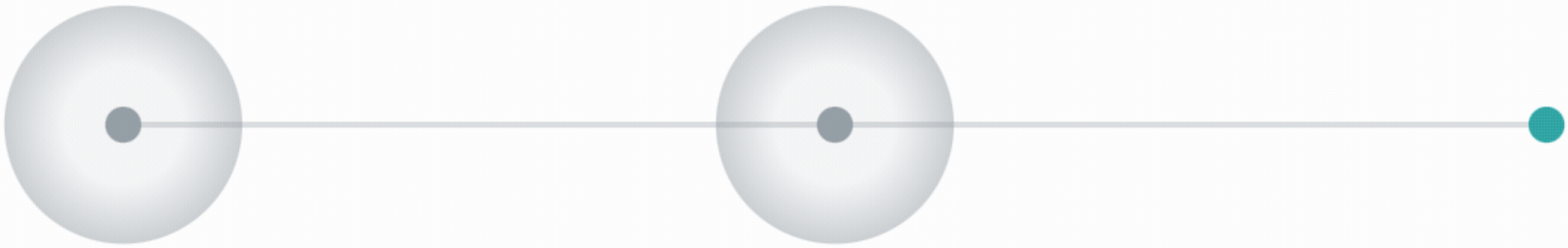
0
Purchased

Open custom settings

```
useEffect(() => {
  if (orders.count > prevOrders.count) {

    let increment = totalOrders.count - prevTotalOrders.count;
    increment = clamp(increment, 1, maxCirclesPerBucket);

    const newOrders = Array(increment)
      .fill({
        // "Checking out" bucket position:
        fromCx:  xScale(1),
        // "Purchased" bucket position:
        toCx: xScale(2),
        // how big the radius should be based
        // on the increment size:
        radius: getRunningDotRadius(increment)
      });

    setNewOrders(newOrders);
  }
}, [data]);
```

Run dots,

Customer



1
Active carts

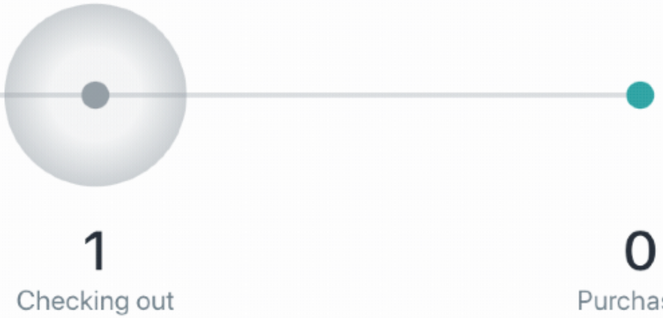
1
Checking out

0
Purchased

Open custom settings

Run dots, run!

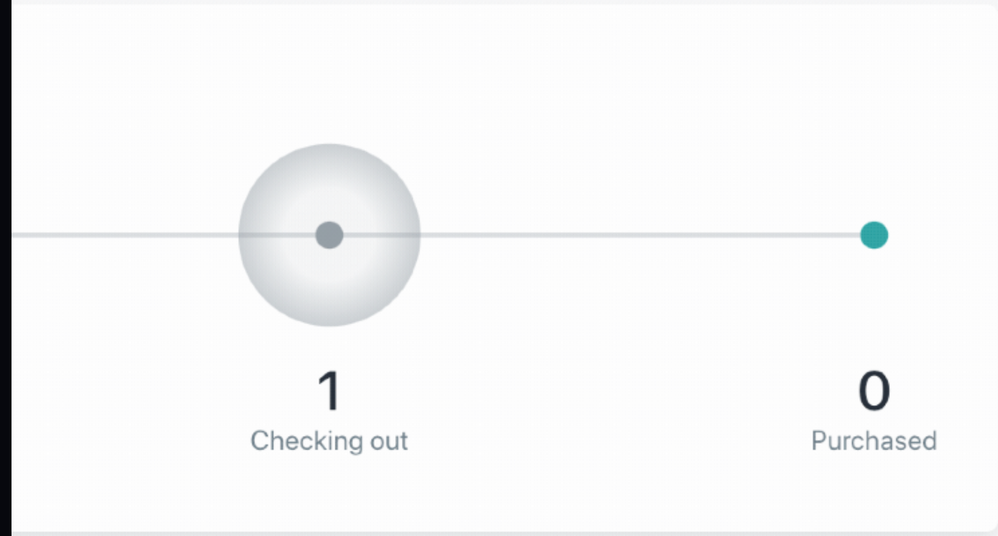
```
useEffect(() => {
  if (orders.count > prevOrders.count) {

    let increment = totalOrders.count - prevTotalOrders.count;
    increment = clamp(increment, 1, maxCirclesPerBucket);

    const newOrders = Array(increment)
      .fill({
        // "Checking out" bucket position:
        fromCx:  xScale(1),
        // "Purchased" bucket position:
        toCx: xScale(2),
        // how big the radius should be based
        // on the increment size:
        radius: getRunningDotRadius(increment)
      });

    setNewOrders(newOrders);
  }
}, [data]);
```



Open custom settings

Run dots, run!

```
useEffect(() => {
  if (orders.count > prevOrders.count) {

    let increment = totalOrders.count - prevTotalOrders.count;
    increment = clamp(increment, 1, maxCirclesPerBucket);

    const newOrders = Array(increment)
      .fill({
        // "Checking out" bucket position:
        fromCx:  xScale(1),
        // "Purchased" bucket position:
        toCx: xScale(2),
        // how big the radius should be based
        // on the increment size:
        radius: getRunningDotRadius(increment)
      });

    setNewOrders(newOrders);
  }
}, [data]);
```



Open custom settings