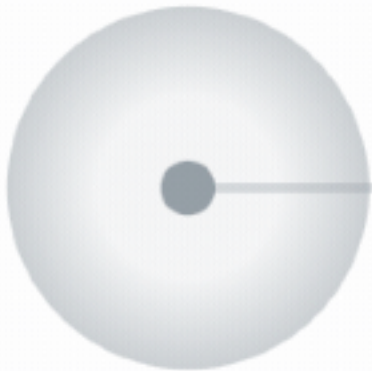
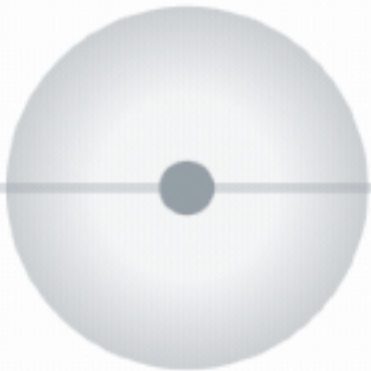


Customer
.....



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,

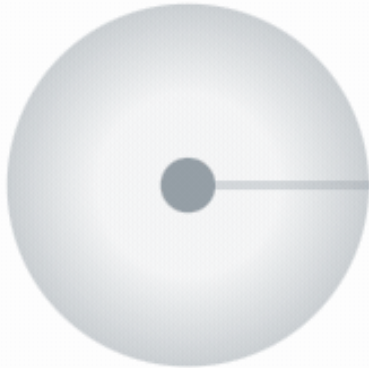

```
const newOrders = Array();
```

```
console.log(newOrders) []
```



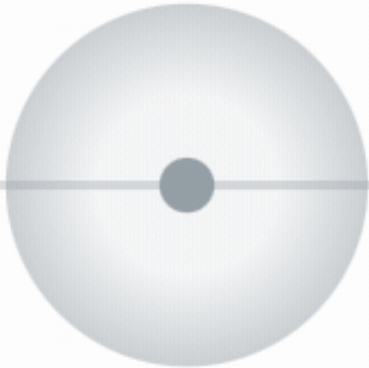


Customer
.....



1

Active carts



1

Checking out



0

Purchased

← Open custom settings

```
const newOrders = Array();
```

```
console.log(newOrders) []
```

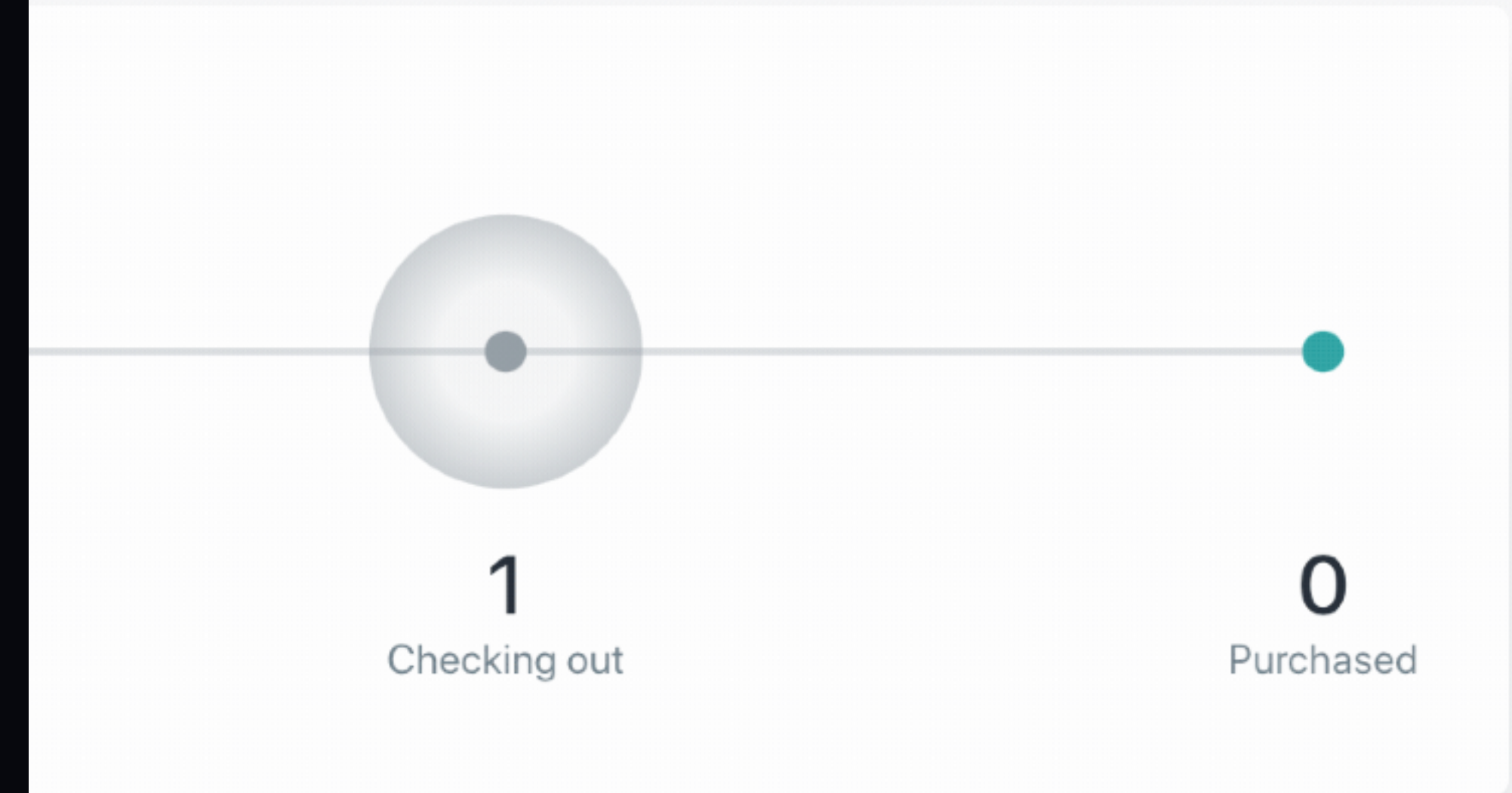
```
const newOrders = Array();
```

```
console.log(newOrders) []
```



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]);
```



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]);
```

