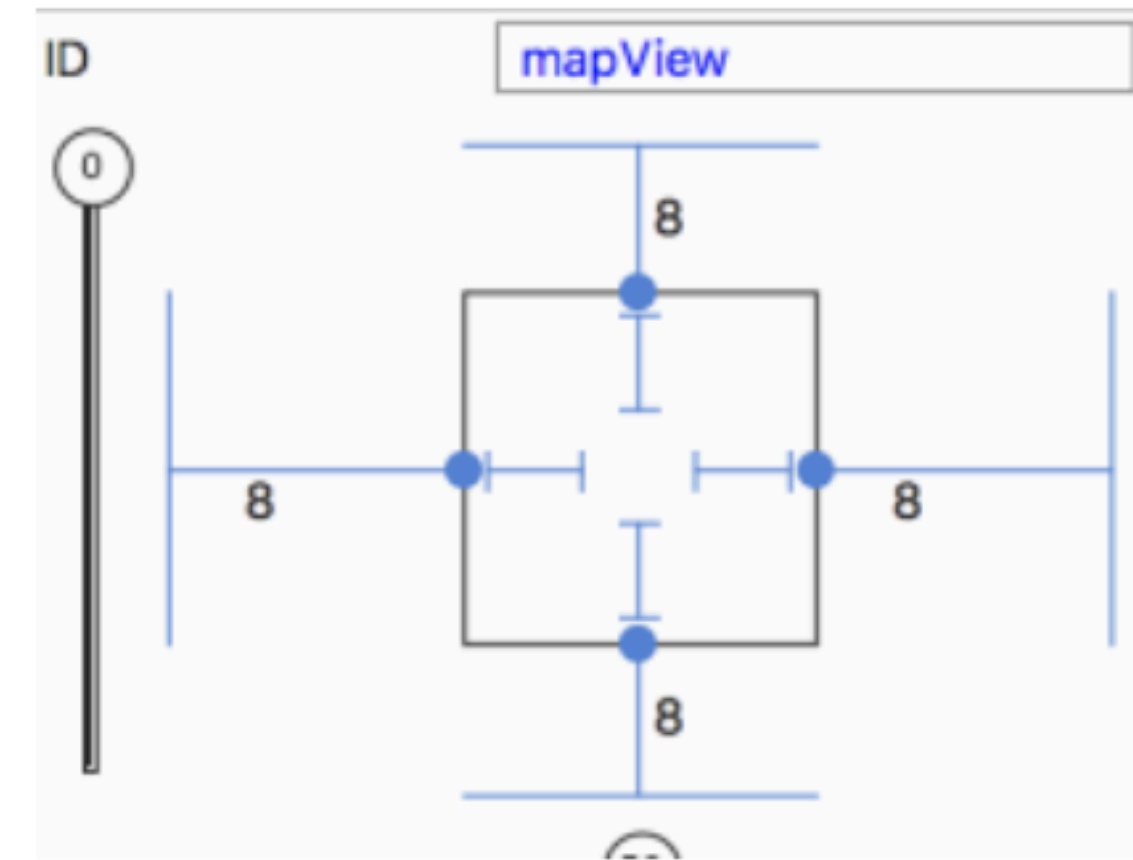
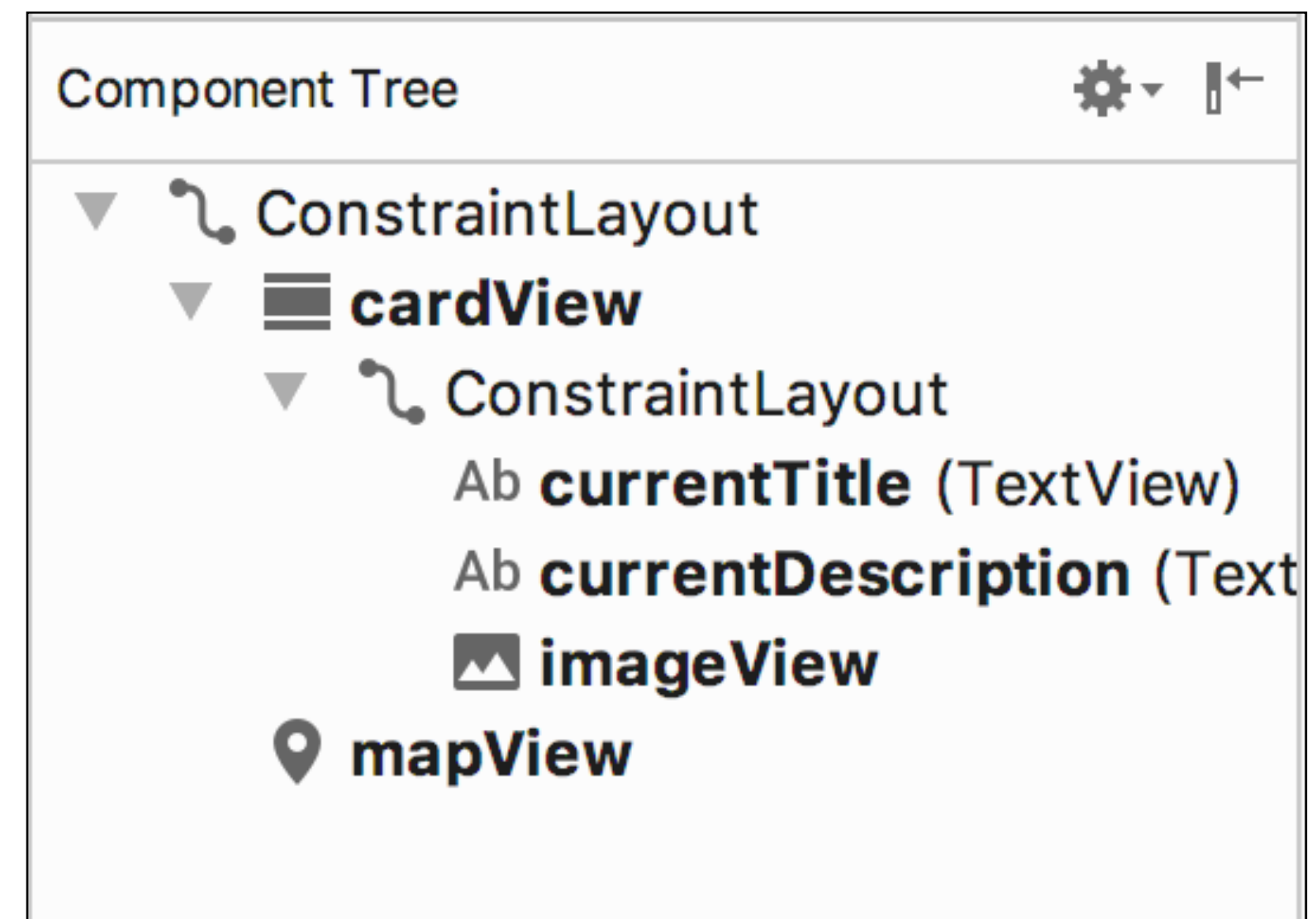
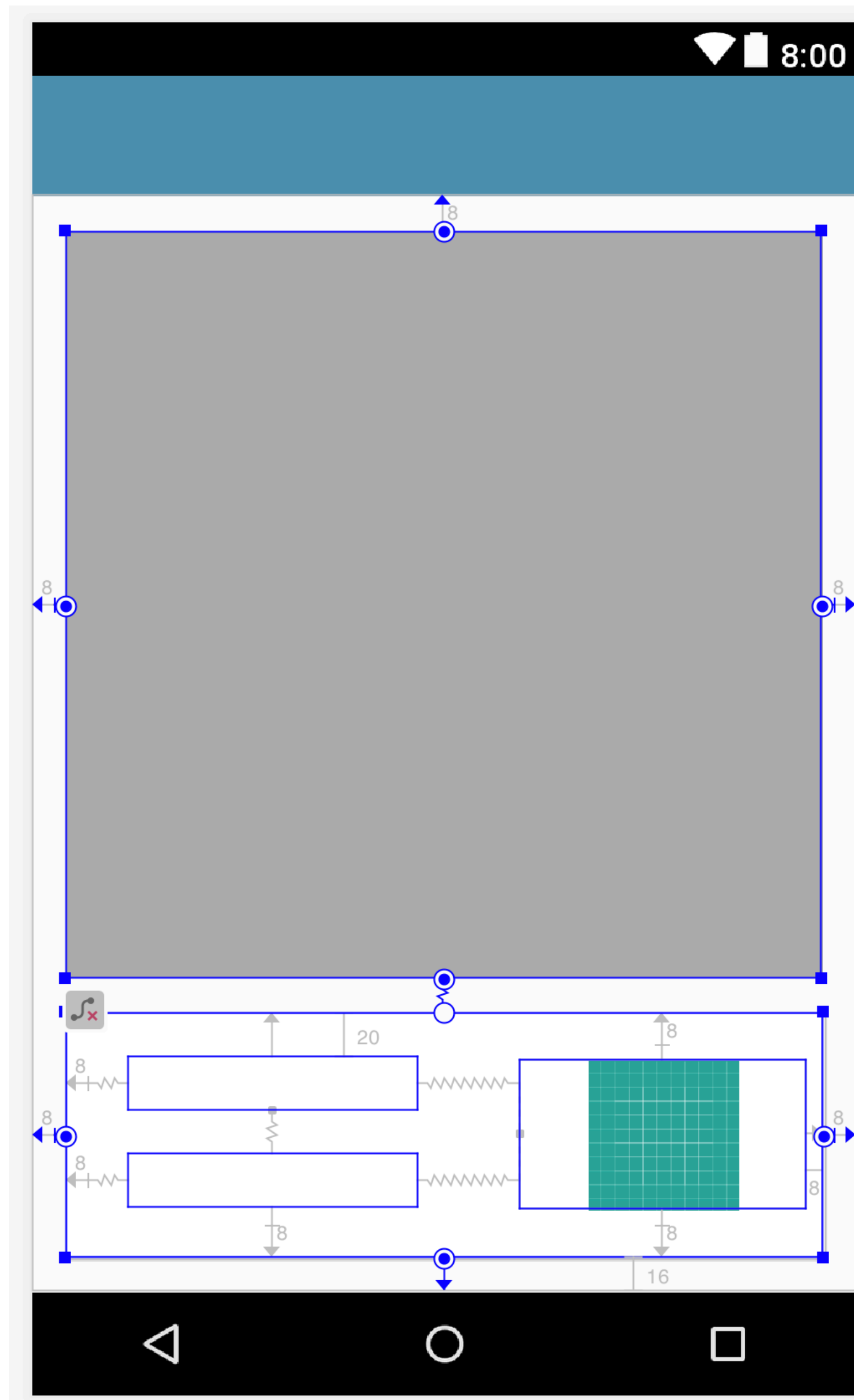


Constraint Layout

Constraint Layout



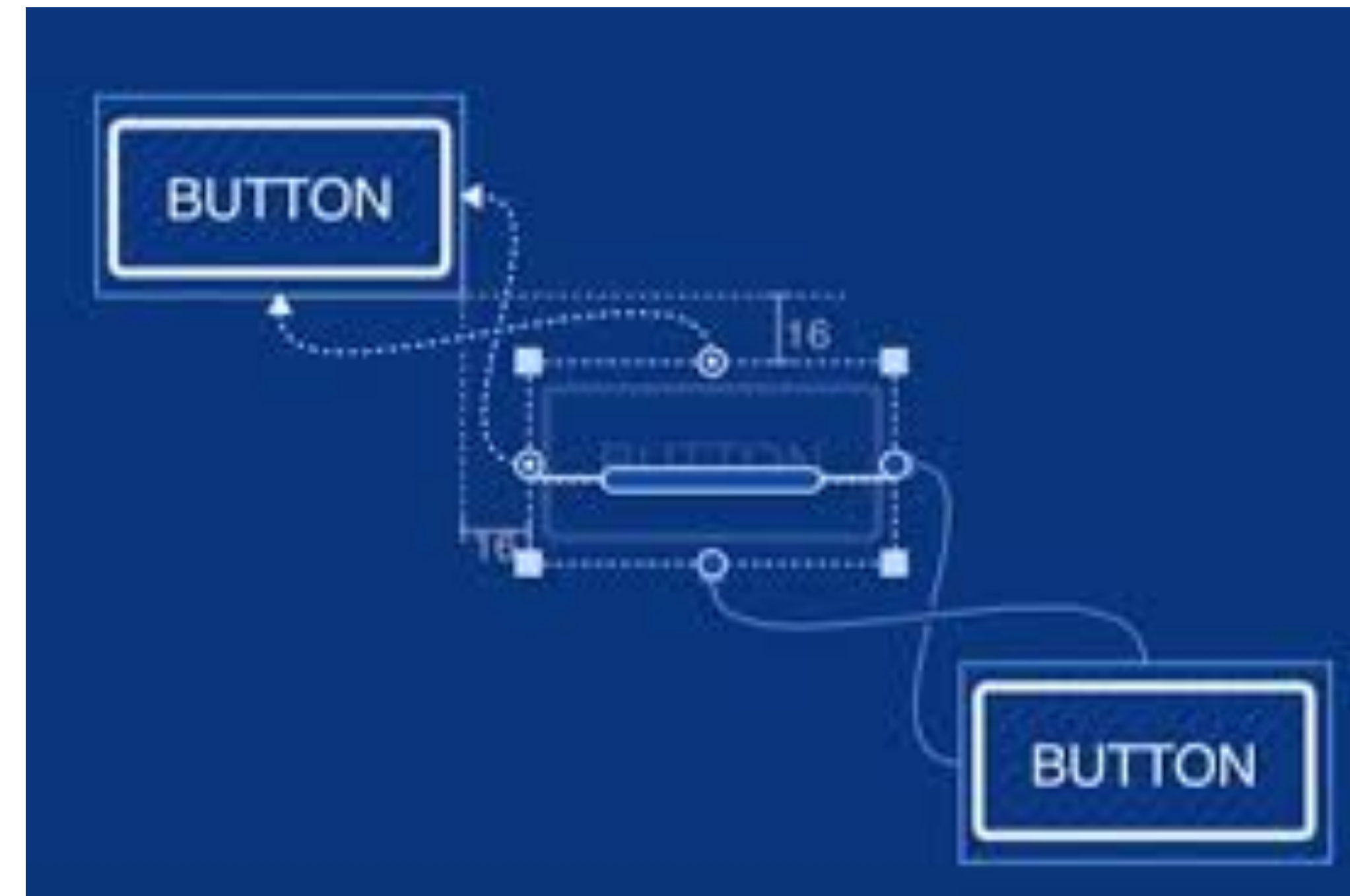
ConstraintLayout allows you to create large and complex layouts with a flat view



<https://developer.android.com/training/constraint-layout/>

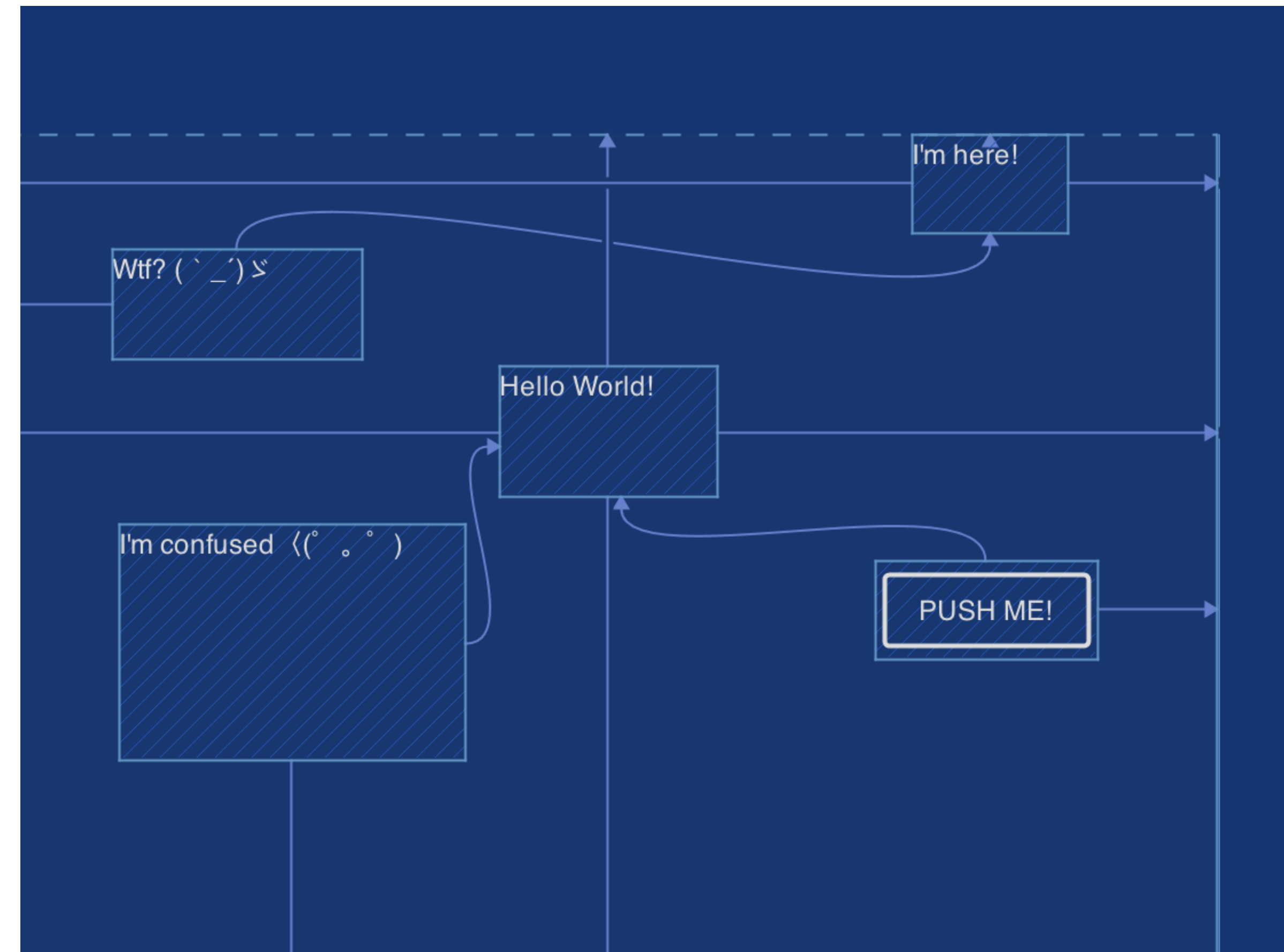
Constraint Layout

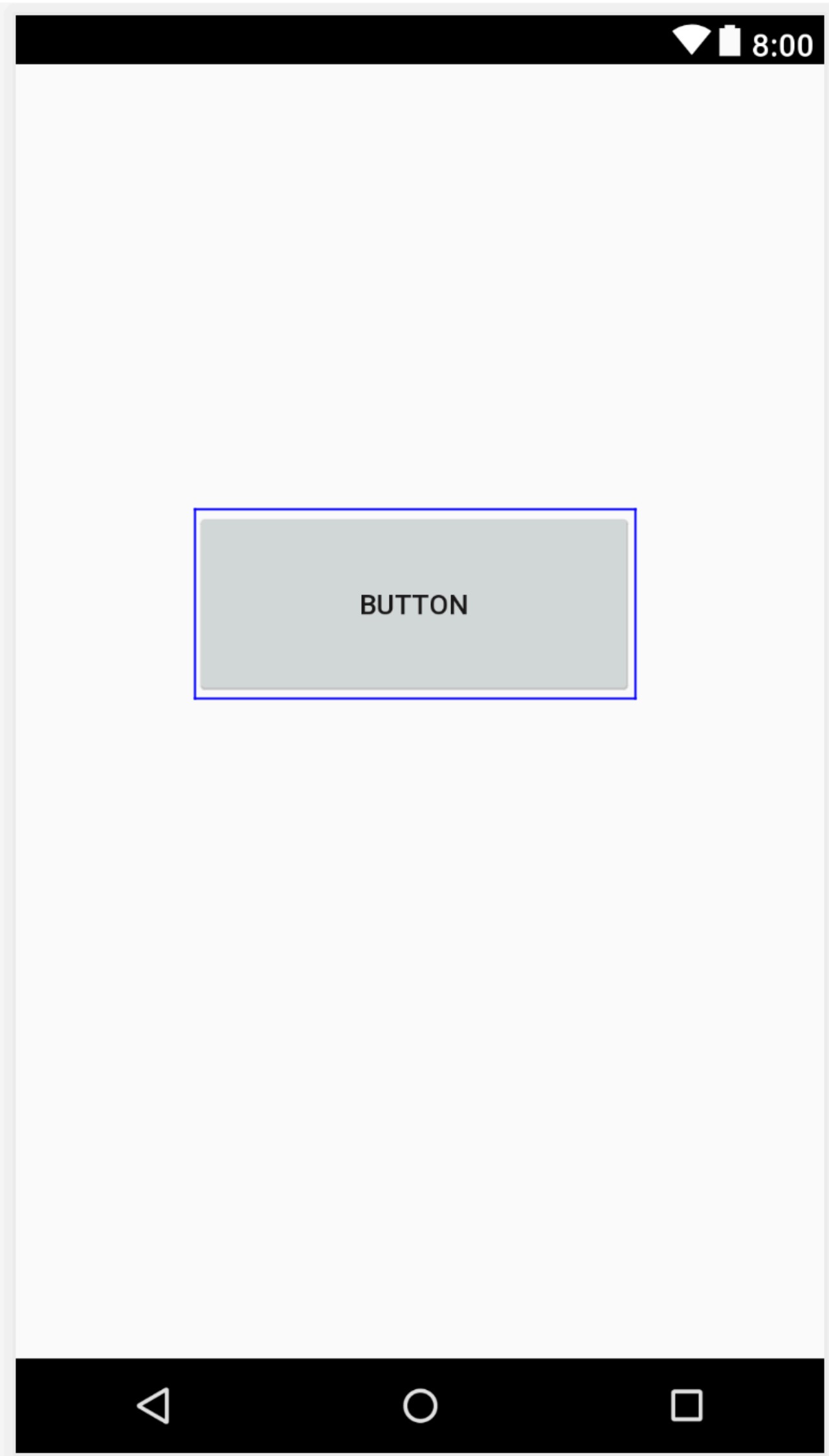
- Similar to RelativeLayout in that all views are laid out according to relationships between sibling views and the parent layout.
- All the power of ConstraintLayout is available directly from the Layout Editor's visual tools, because the layout API and the Layout Editor were specially built for each other.
- So you can build your layout with ConstraintLayout entirely by drag-and-dropping instead of editing the XML



Horizontal & Vertical Constraints

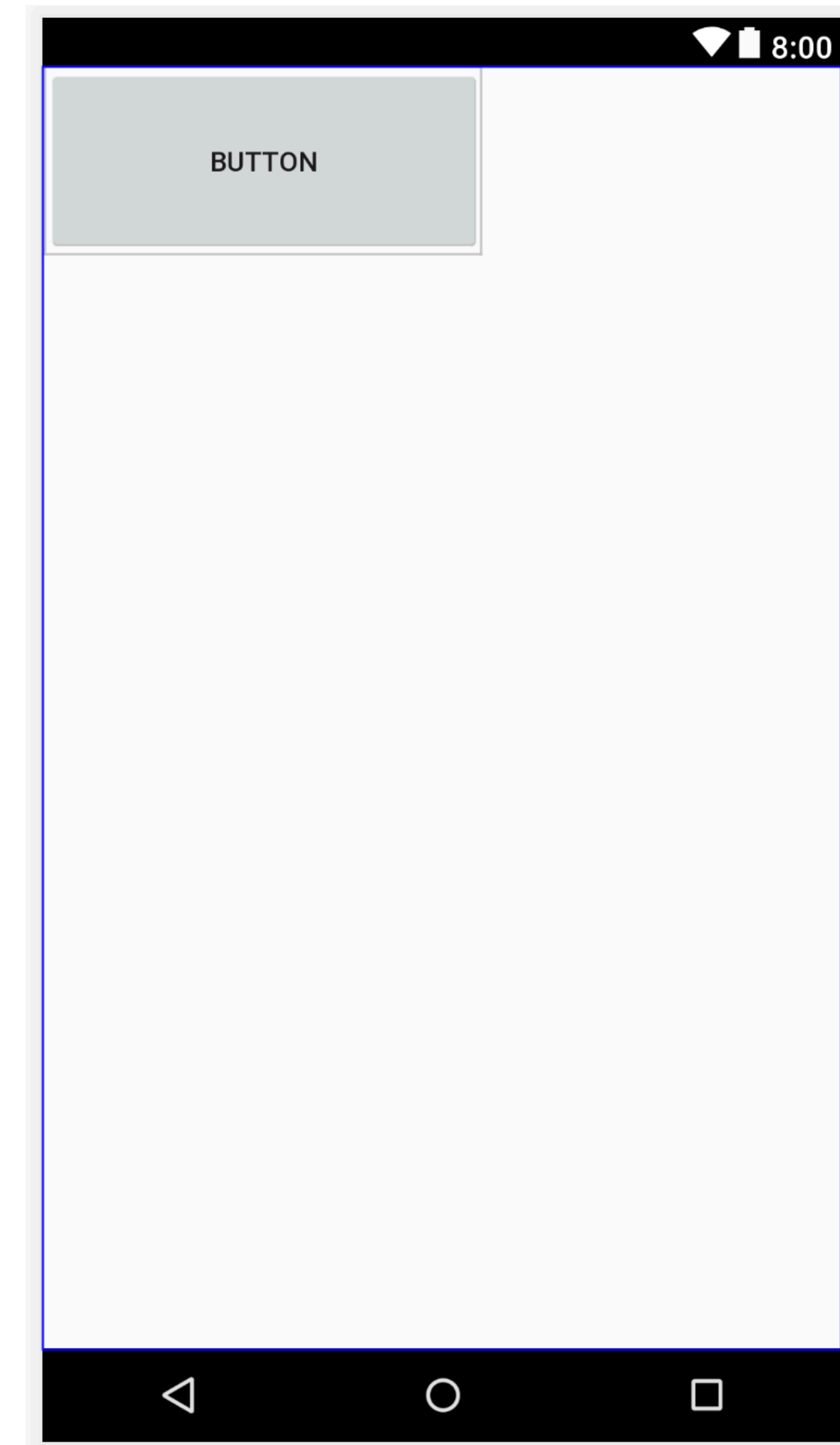
- To define a view's position you must add at least one horizontal and one vertical constraint for the view.
- Each constraint represents a connection or alignment to another view, the parent layout, or an invisible guideline.
- Each constraint defines the view's position along either the vertical or horizontal axis; so each view must have a minimum of one constraint for each axis, but often more are necessary.

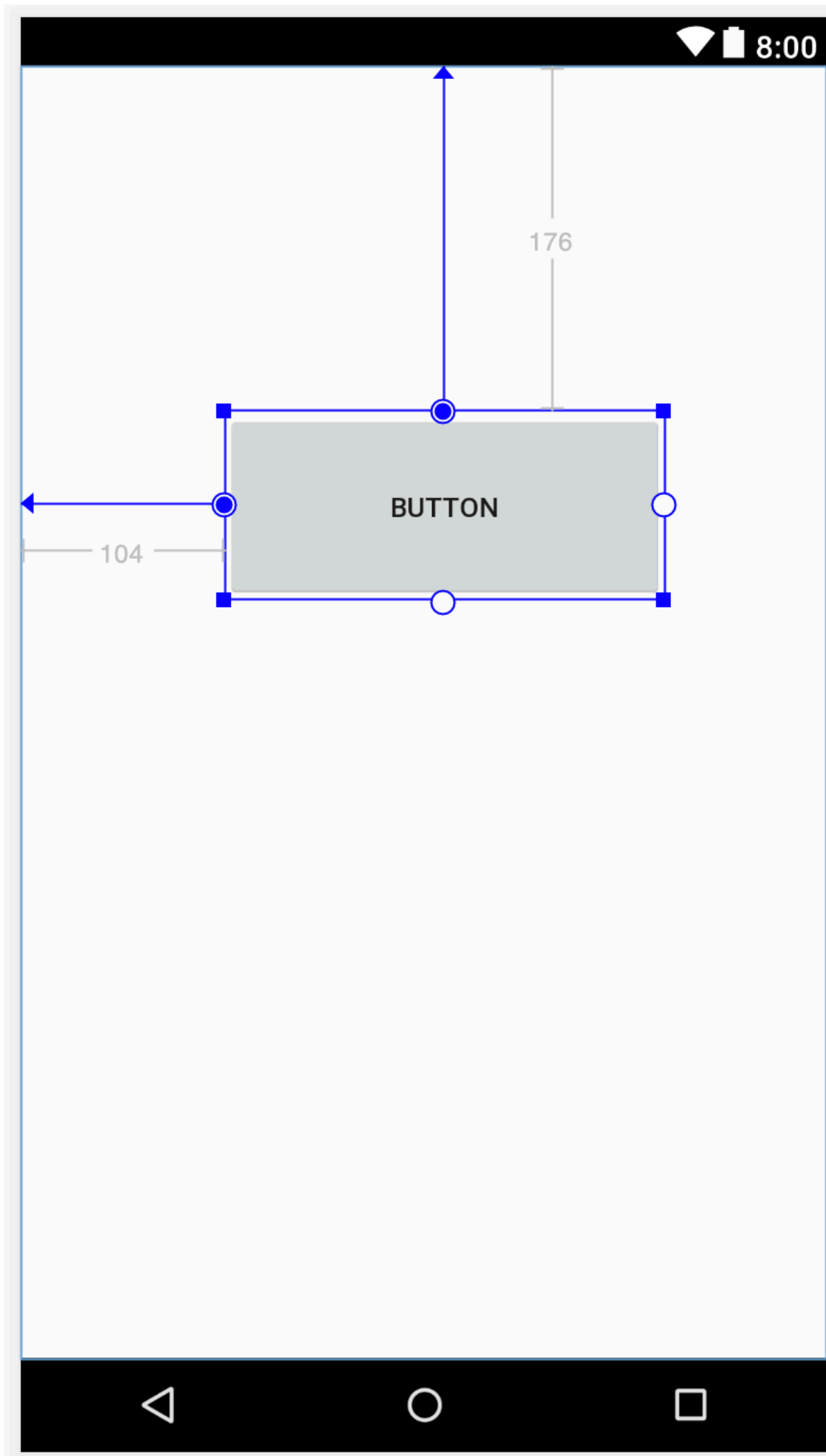




- When you drop a view into the Layout Editor, it stays where you leave it even if it has no constraints.
- However, this is only to make editing easier; if a view has no constraints when you run your layout on a device, it is drawn at position [0,0] (the top-left corner).

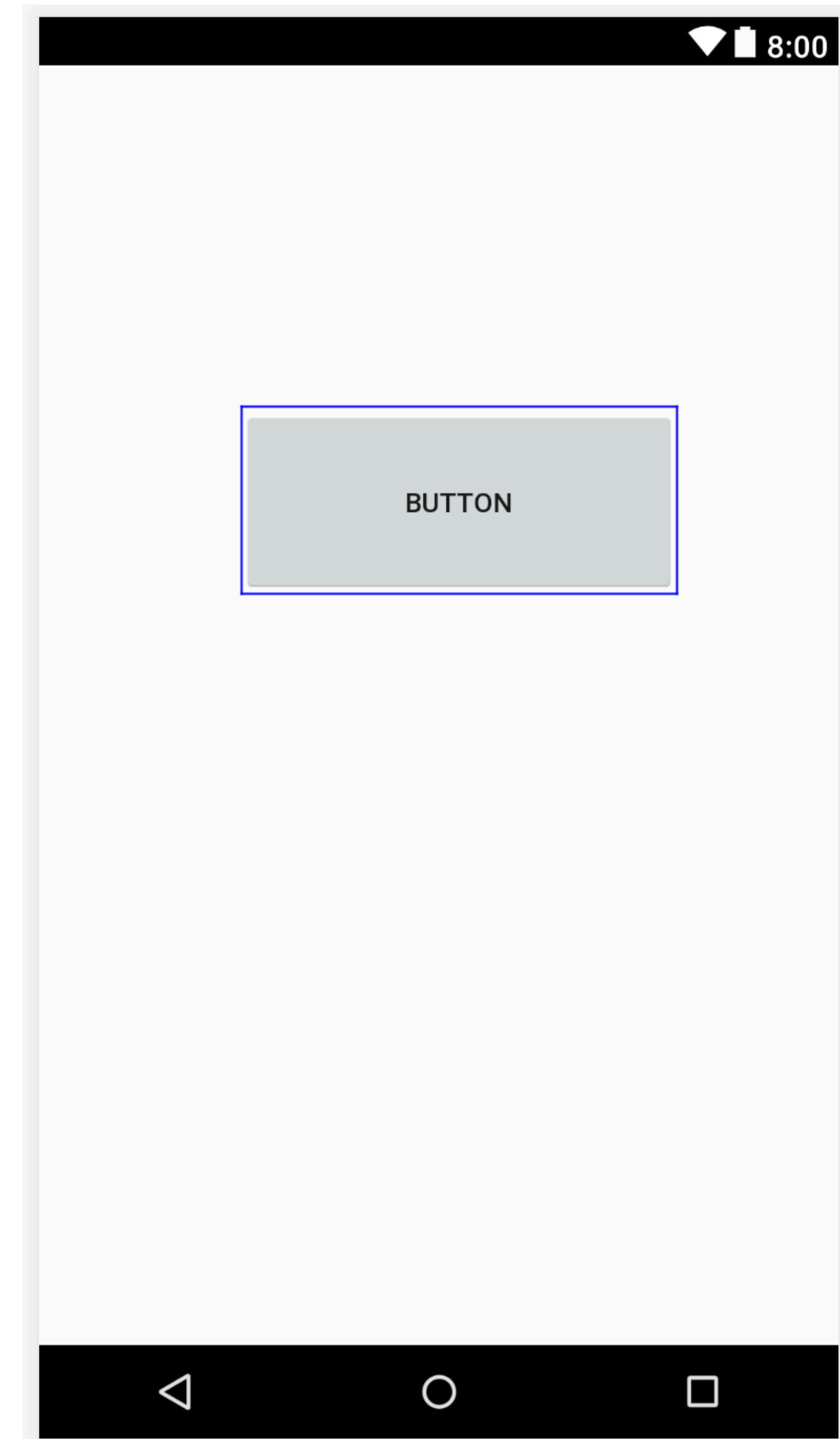
```
<Button  
  android:id="@+id/button"  
  android:layout_width="224dp"  
  android:layout_height="97dp"  
  android:text="Button"  
  tools:layout_editor_absoluteX="107dp"  
  tools:layout_editor_absoluteY="175dp" />
```



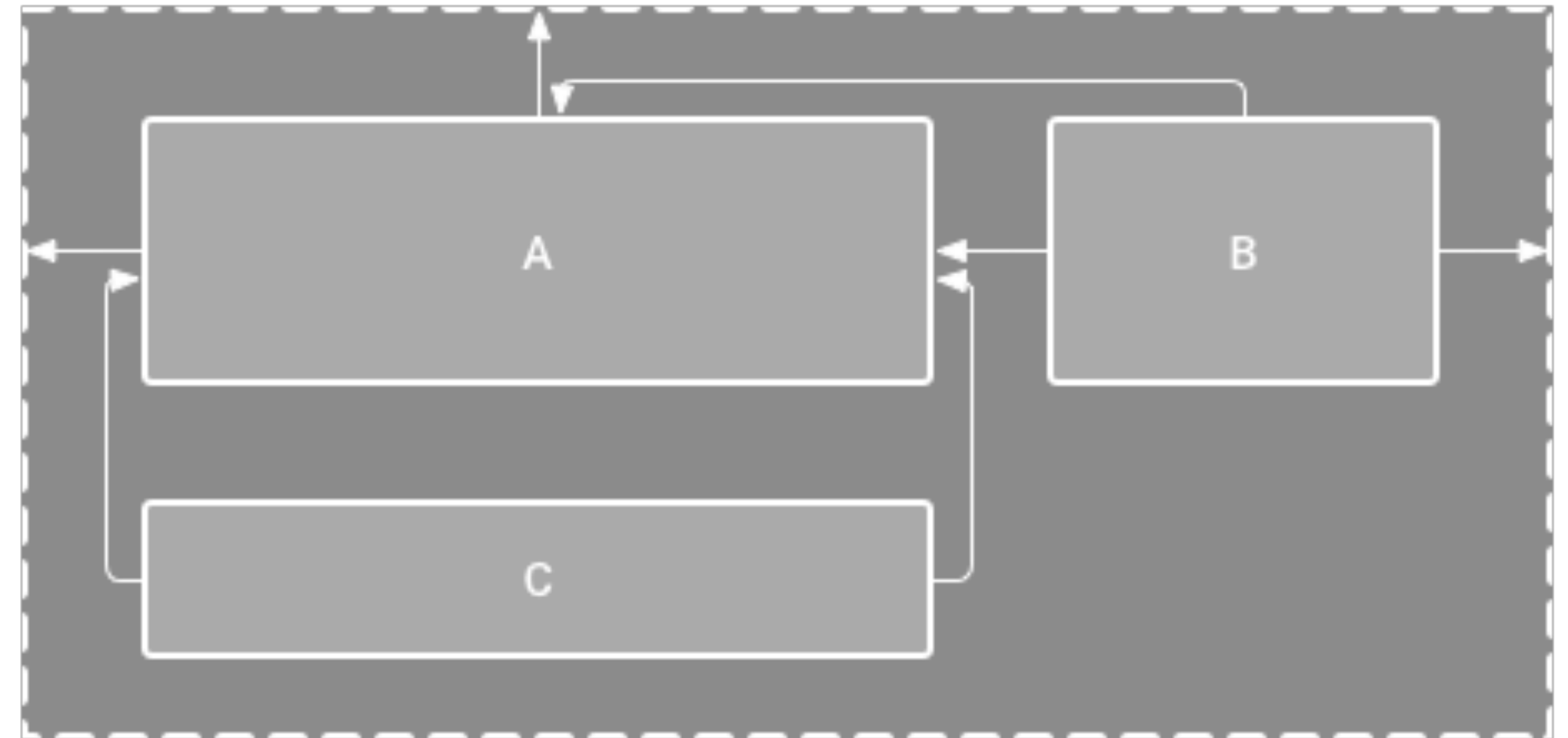


- Constraints attached to top & left.
- These constraints then influence the layout at runtime

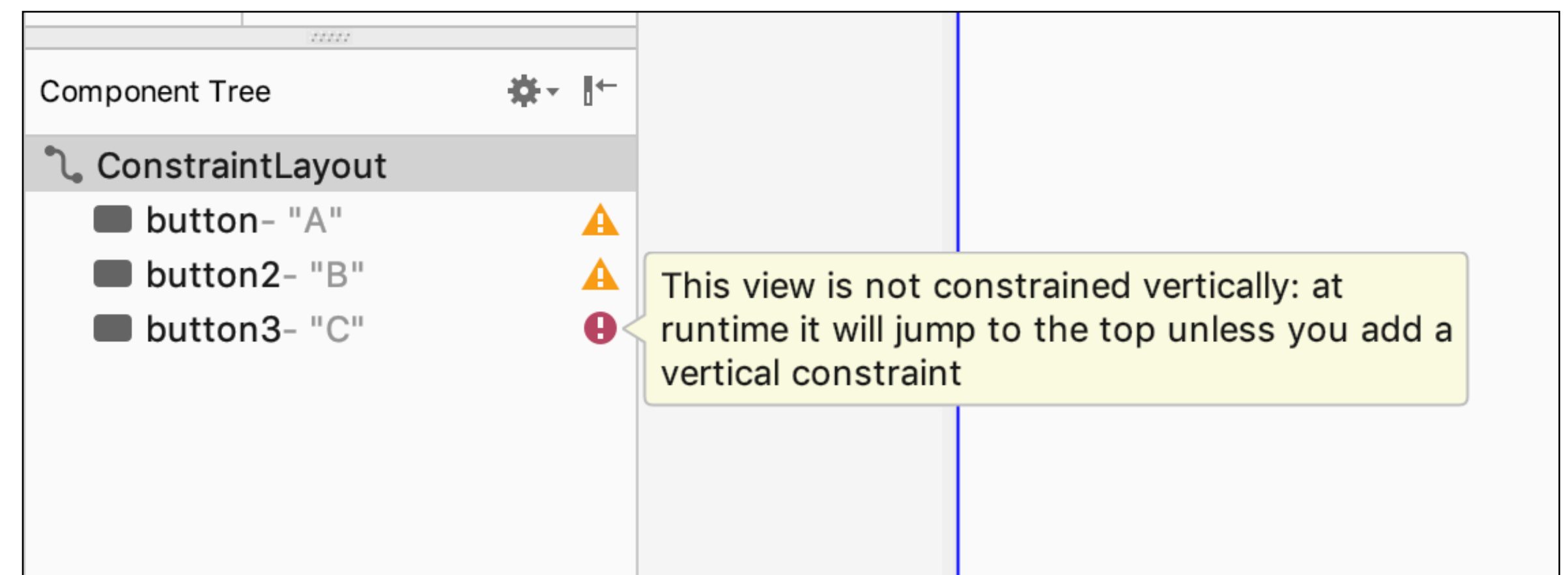
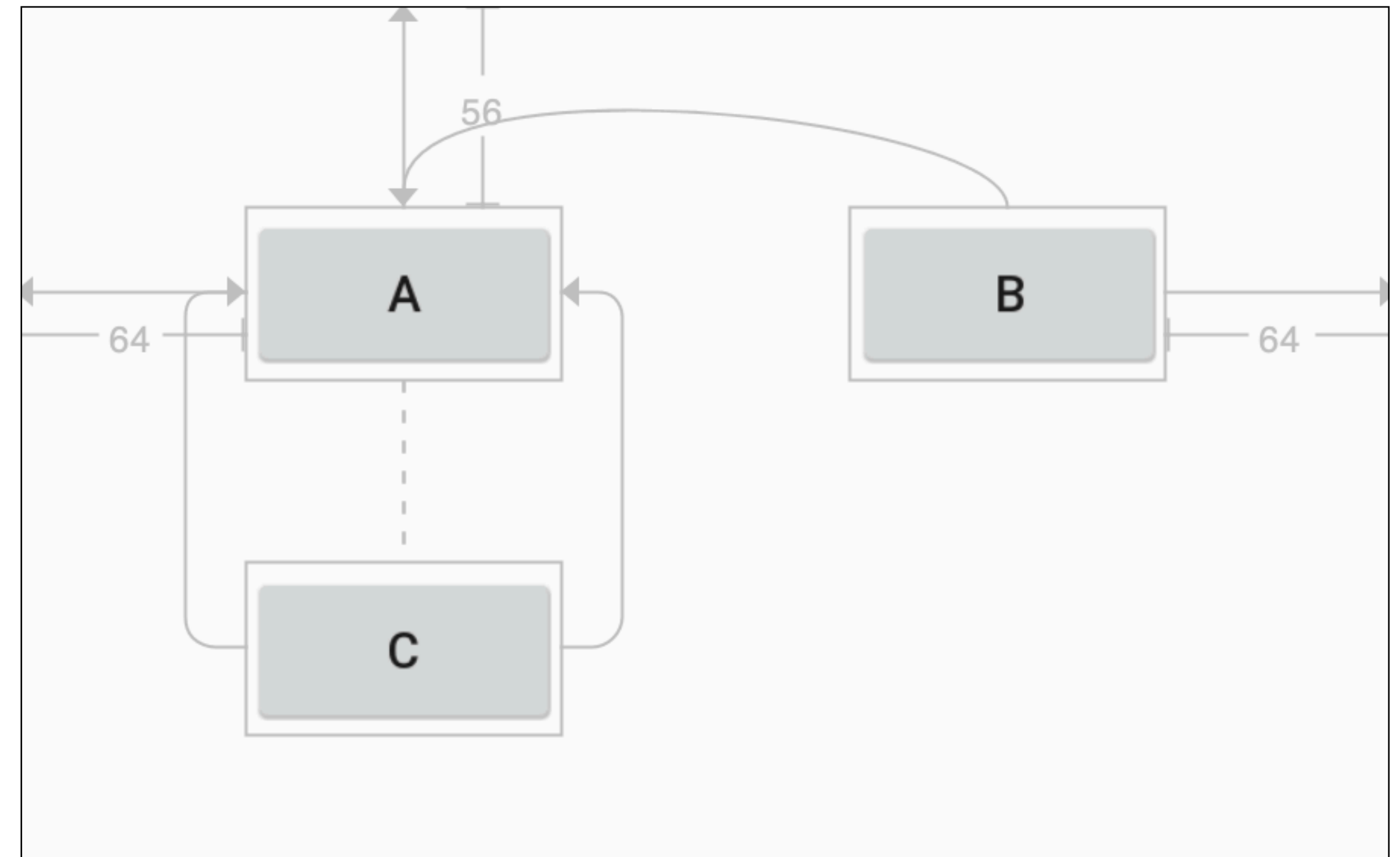
```
<Button
  android:id="@+id/button"
  android:layout_width="224dp"
  android:layout_height="97dp"
  android:layout_marginStart="104dp"
  android:layout_marginTop="176dp"
  android:text="Button"
  app:layout_constraintStart_toStartOf="parent"
  app:layout_constraintTop_toTopOf="parent" />
```



- This layout looks good in the editor, but there's no vertical constraint on view C.
- When this layout draws on a device, view C horizontally aligns with the left and right edges of view A, but appears at the top of the screen because it has no vertical constraint.

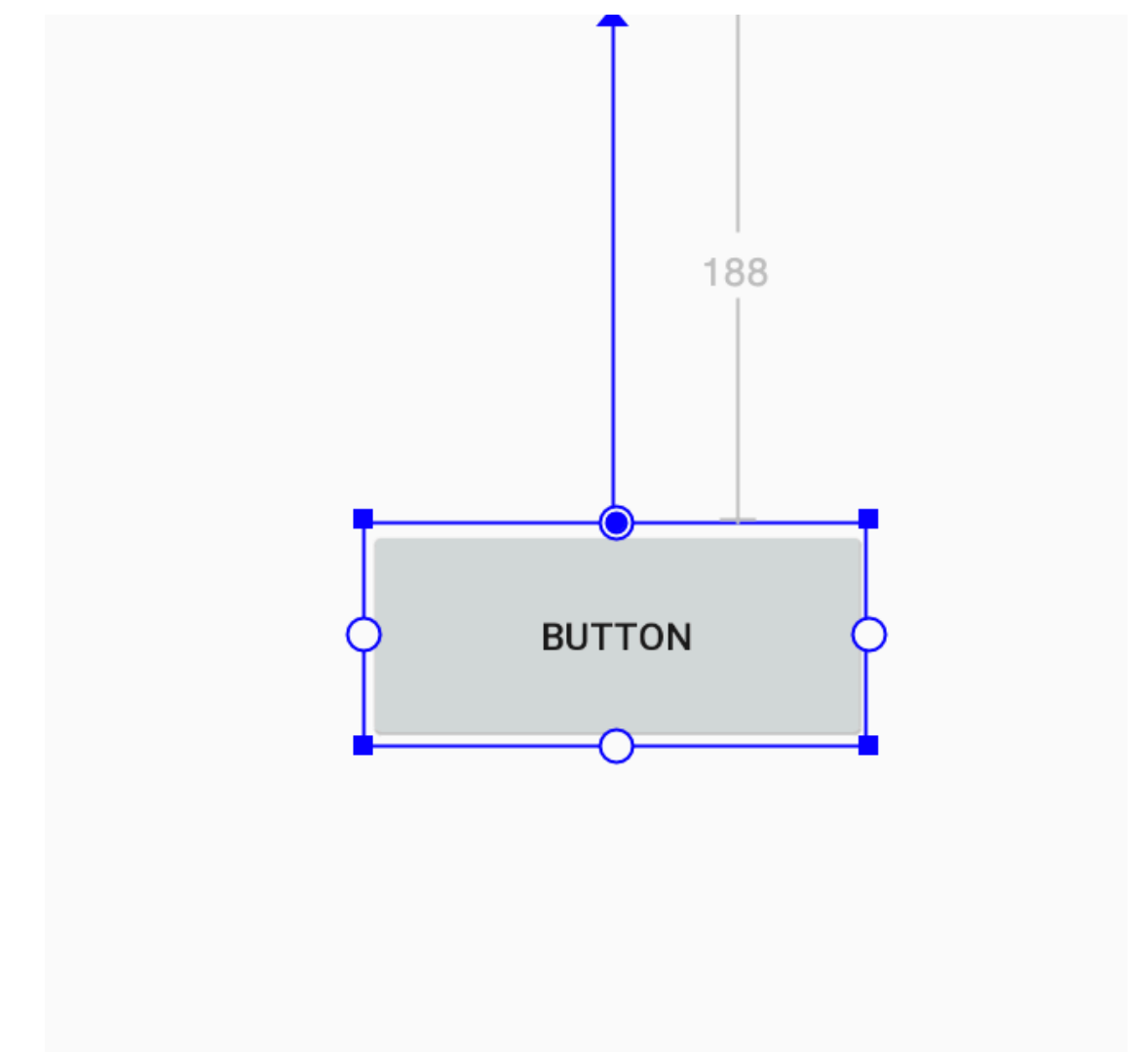
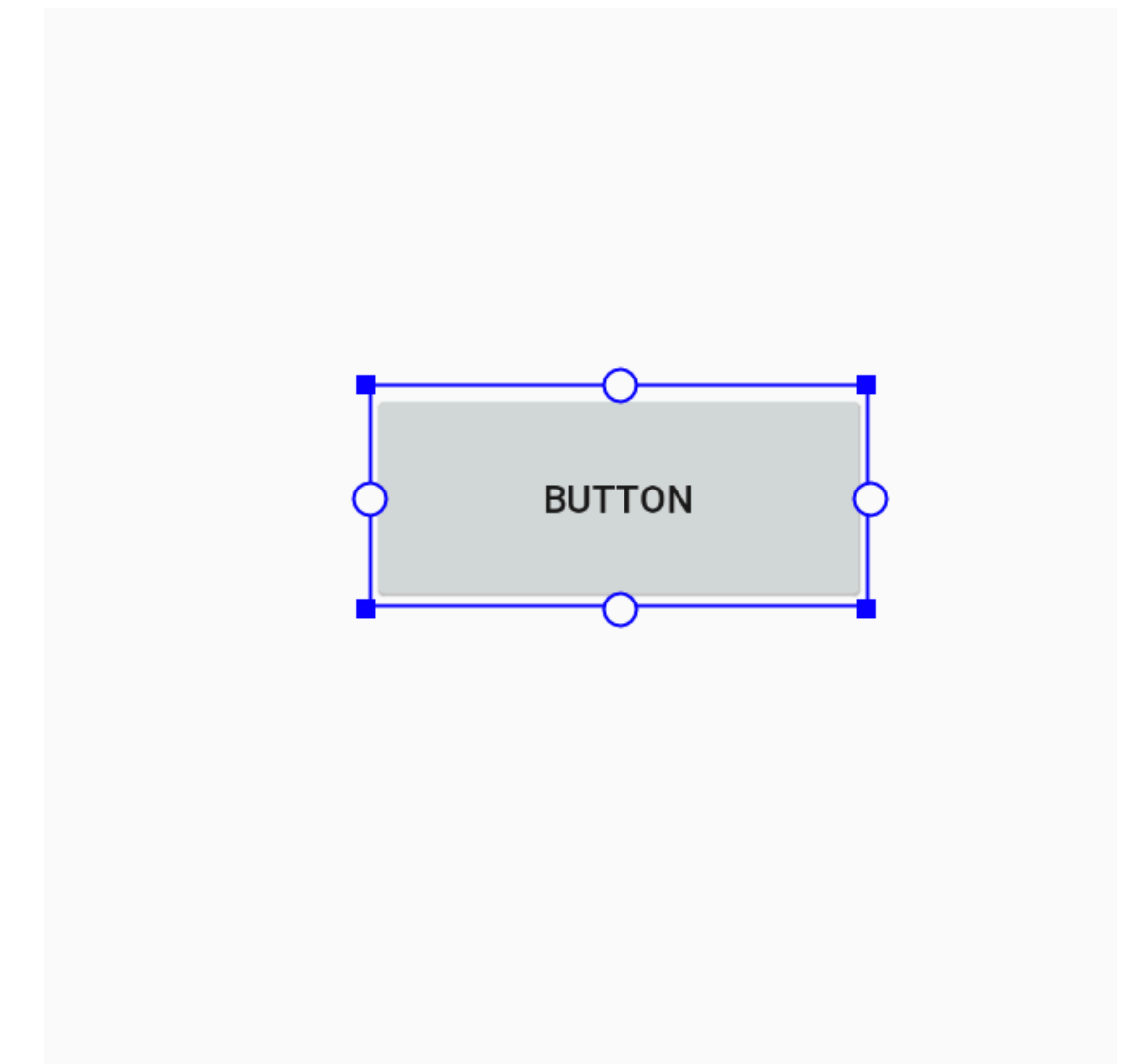


- Although a missing constraint won't cause a compilation error, the Layout Editor indicates missing constraints as an error in the toolbar.
- Here, C is not constrained vertically - so will drift to the top when rendered.
- To help you avoid missing constraints, the Layout Editor can automatically add constraints for you with the Autoconnect and infer constraints features.



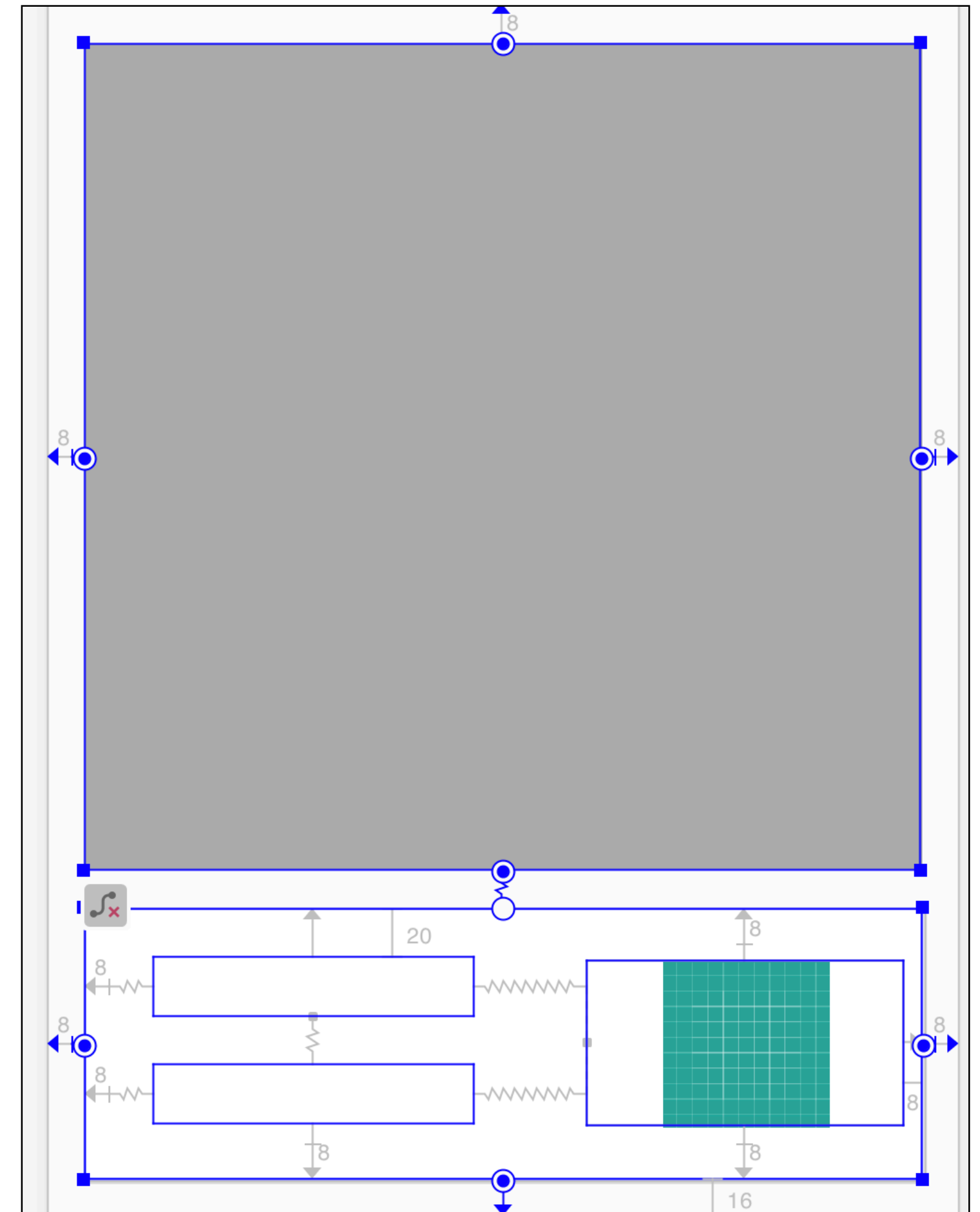
Using Constraints

- Drag a view from the Palette window into the editor. When you add a view in a ConstraintLayout, it displays a bounding box with square resizing handles on each corner and circular constraint handles on each side.
- Click a constraint handle and drag it to an available anchor point (the edge of another view, the edge of the layout, or a guideline).



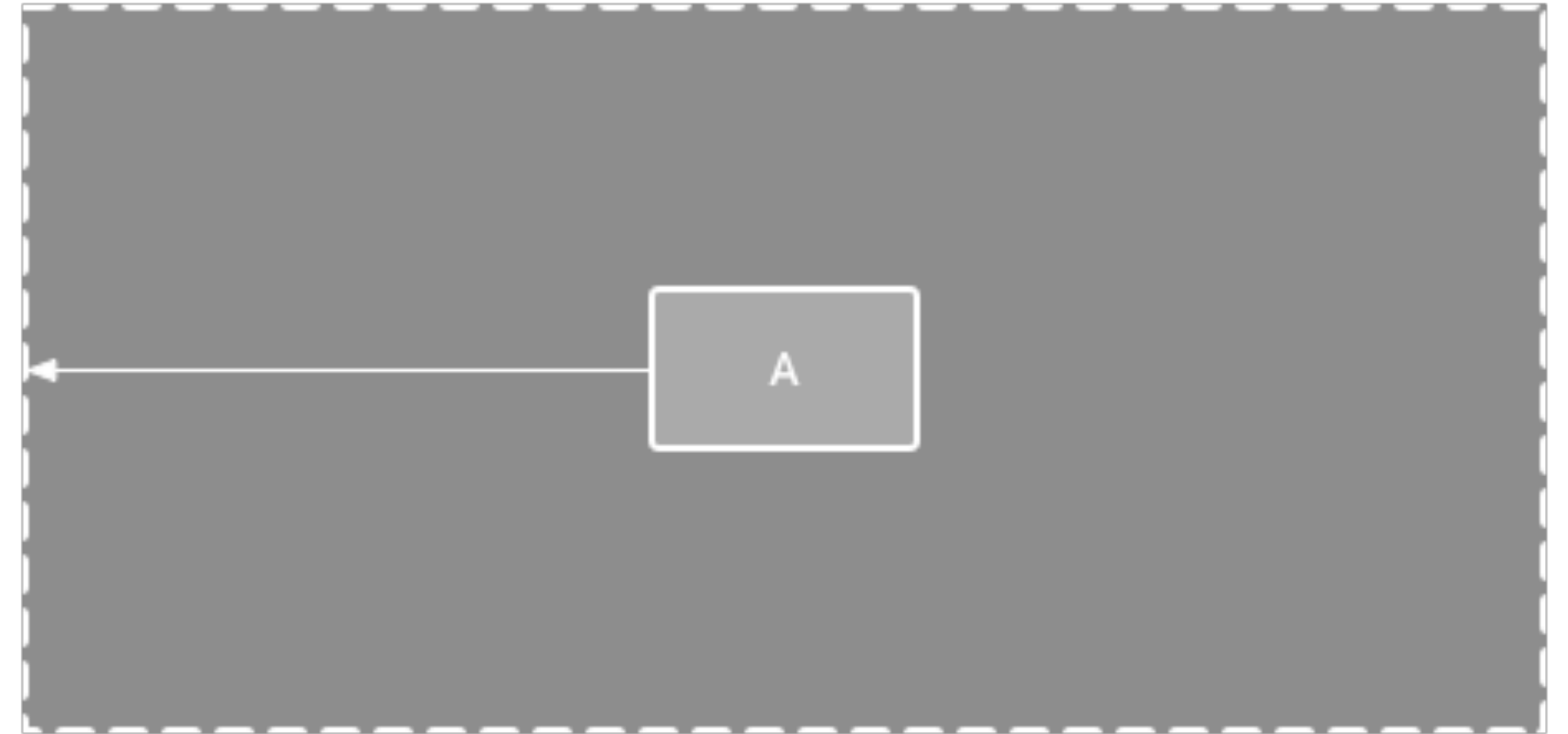
Constraint Rules

- Every view must have at least two constraints: one horizontal and one vertical.
- You can create constraints only between a constraint handle and an anchor point that share the same plane. So a vertical plane (the left and right sides) of a view can be constrained only to another vertical plane; and baselines can constrain only to other baselines.
- Each constraint handle can be used for just one constraint, but you can create multiple constraints (from different views) to the same anchor point.



Constraint Behaviours: Parent Position

- Constrain the side of a view to the corresponding edge of the layout.
- The left side of the view is connected to the left edge of the parent layout. You can define the distance from the edge with margins

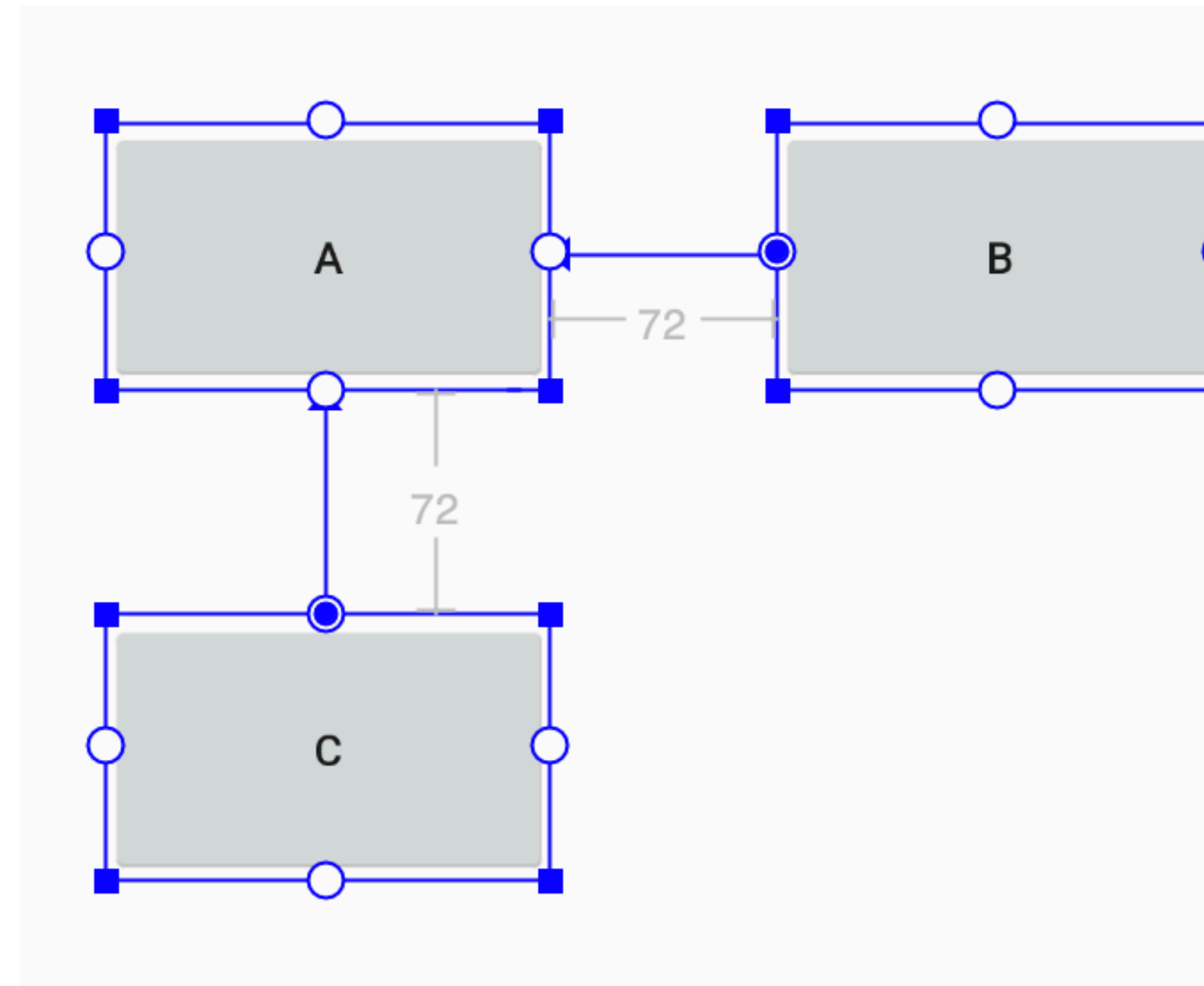


```
<Button
  android:id="@+id/button4"
  android:layout_width="142dp"
  android:layout_height="85dp"
  android:layout_marginStart="132dp"
  android:text="Button"
  app:layout_constraintStart_toStartOf="parent"
  tools:layout_editor_absoluteY="186dp" />
```

Constraint Behaviours: **Order Position**

Define the order of appearance for two views, either vertically or horizontally.

B is constrained to always be to the right of A, and C is constrained below A. However, these constraints do not imply alignment, so B can still move up and down.



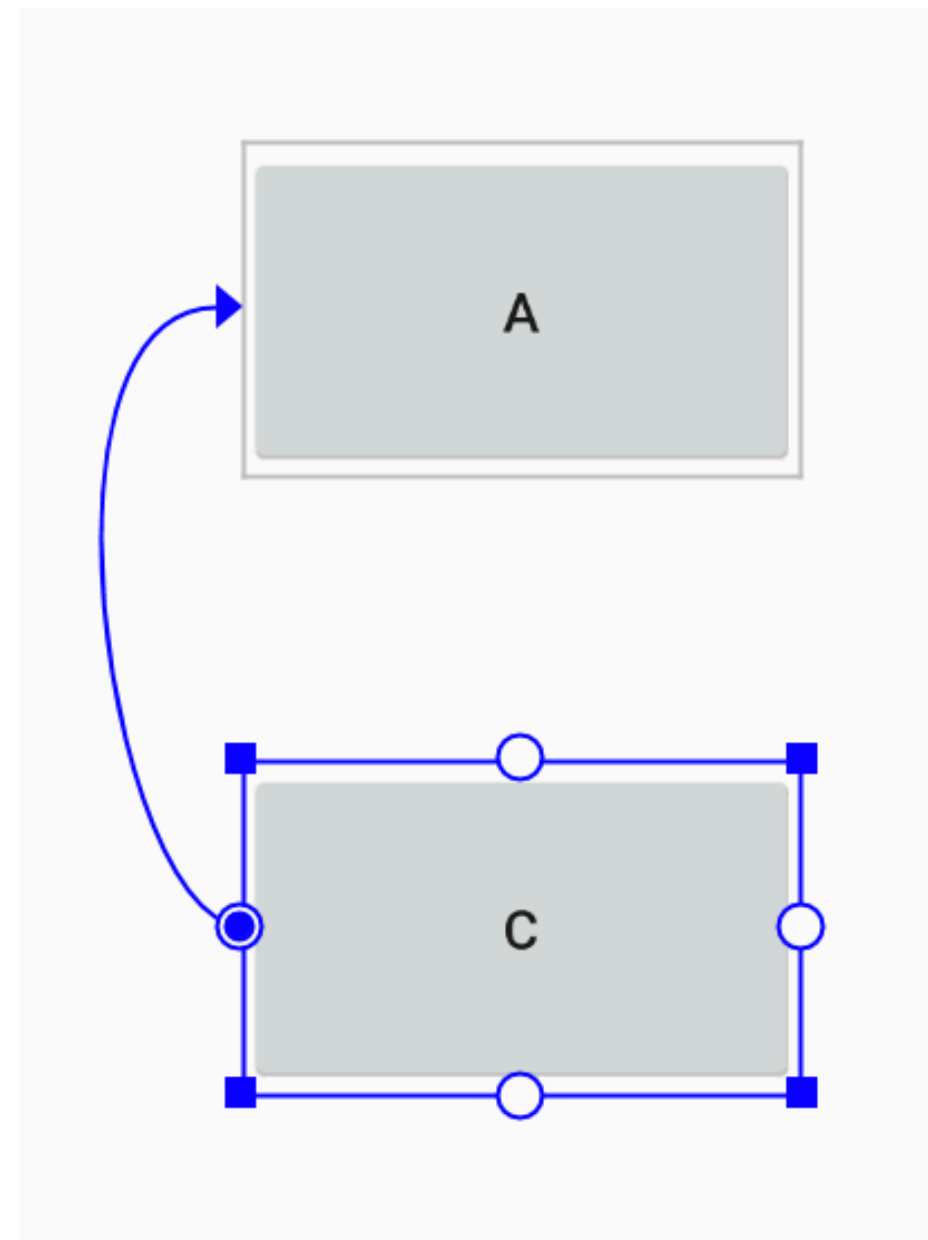
```
<Button
  android:id="@+id/A"
  android:layout_width="142dp"
  android:layout_height="85dp"
  android:text="A"
  tools:layout_editor_absoluteX="47dp"
  tools:layout_editor_absoluteY="186dp" />

<Button
  android:id="@+id/C"
  android:layout_width="142dp"
  android:layout_height="85dp"
  android:layout_marginTop="72dp"
  android:text="C"
  app:layout_constraintTop_toBottomOf="@+id/A"
  tools:layout_editor_absoluteX="47dp" />

<Button
  android:id="@+id/B"
  android:layout_width="142dp"
  android:layout_height="85dp"
  android:layout_marginStart="72dp"
  android:text="B"
  app:layout_constraintStart_toEndOf="@+id/A"
  tools:layout_editor_absoluteY="186dp" />
```

Constraint Behaviours: **Alignment**

- Align the edge of a view to the same edge of another view.
- The left side of B is aligned to the left side of A. If you want to align the view centers, create a constraint on both sides

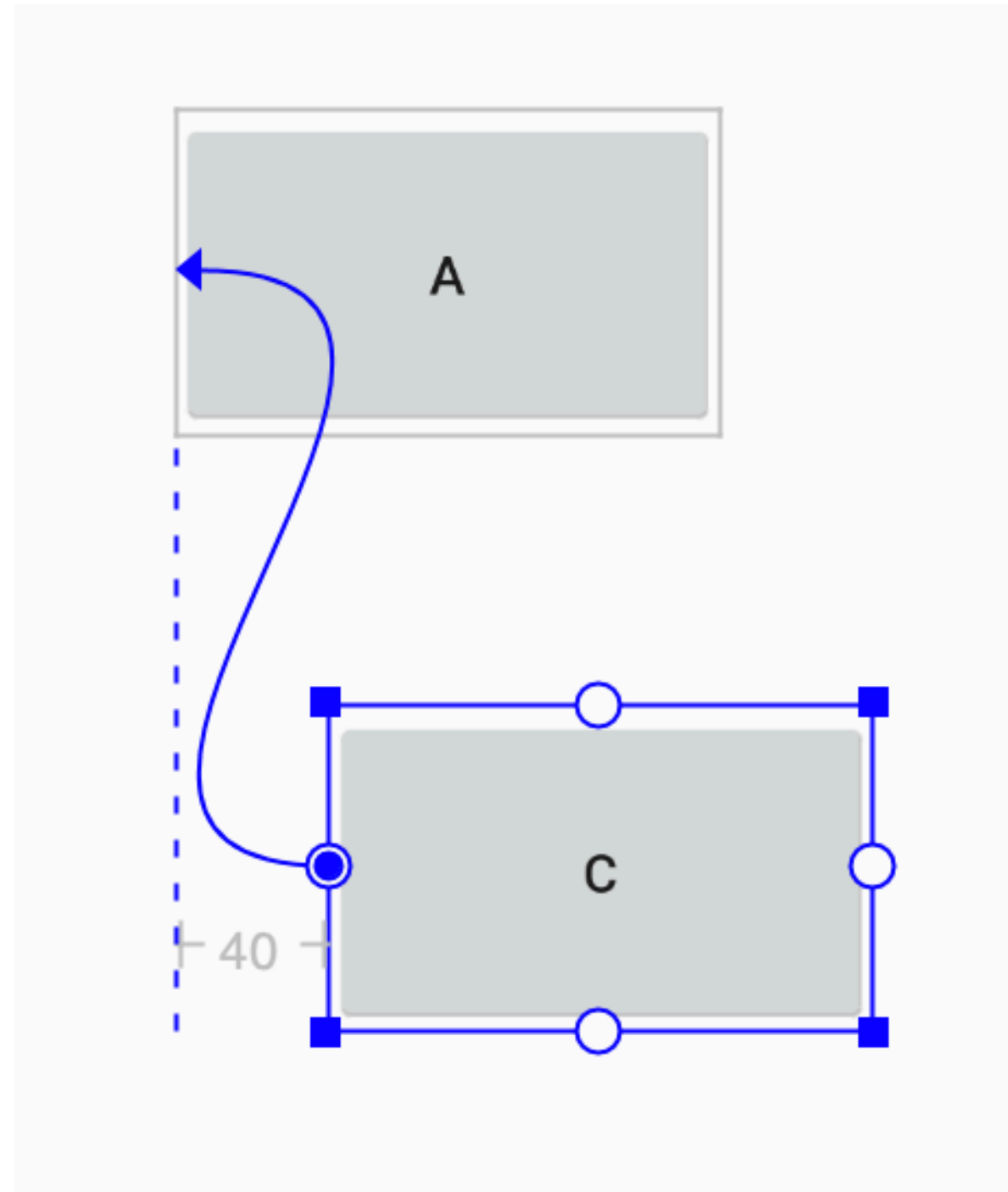


```
<Button
  android:id="@+id/A"
  android:layout_width="142dp"
  android:layout_height="85dp"
  android:text="A"
  tools:layout_editor_absoluteX="76dp"
  tools:layout_editor_absoluteY="183dp" />

<Button
  android:id="@+id/B"
  android:layout_width="142dp"
  android:layout_height="85dp"
  android:text="C"
  app:layout_constraintStart_toStartOf="@+id/A"
  tools:layout_editor_absoluteY="340dp" />
```

Constraint Behaviours: **Alignment with Margin**

- You can offset the alignment by dragging the view inward from the constraint.
- B with a 40dp offset alignment. The offset is defined by the constrained view's margin.
- You can also select all the views you want to align, and then click **Align** in the toolbar to select the alignment type.

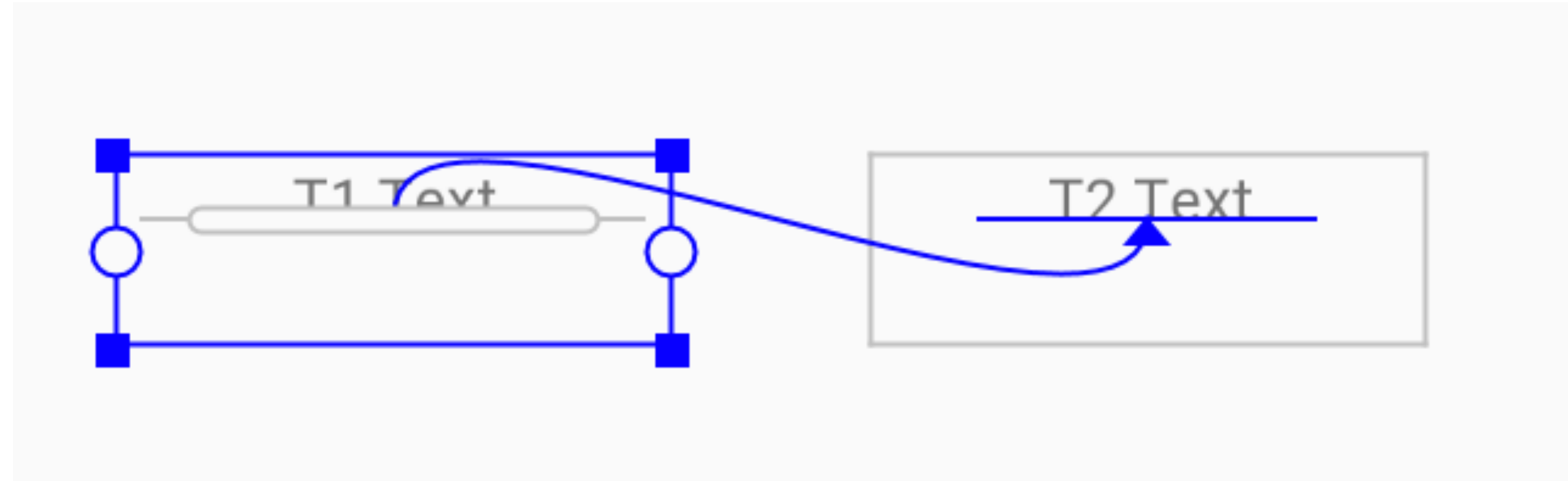


```
<Button
  android:id="@+id/A"
  android:layout_width="142dp"
  android:layout_height="85dp"
  android:text="A"
  tools:layout_editor_absoluteX="76dp"
  tools:layout_editor_absoluteY="183dp" />

<Button
  android:id="@+id/B"
  android:layout_width="142dp"
  android:layout_height="85dp"
  android:layout_marginStart="40dp"
  android:text="C"
  app:layout_constraintStart_toStartOf="@+id/A"
  tools:layout_editor_absoluteY="339dp" />
```

Constraint Behaviours: **Baseline Alignment**

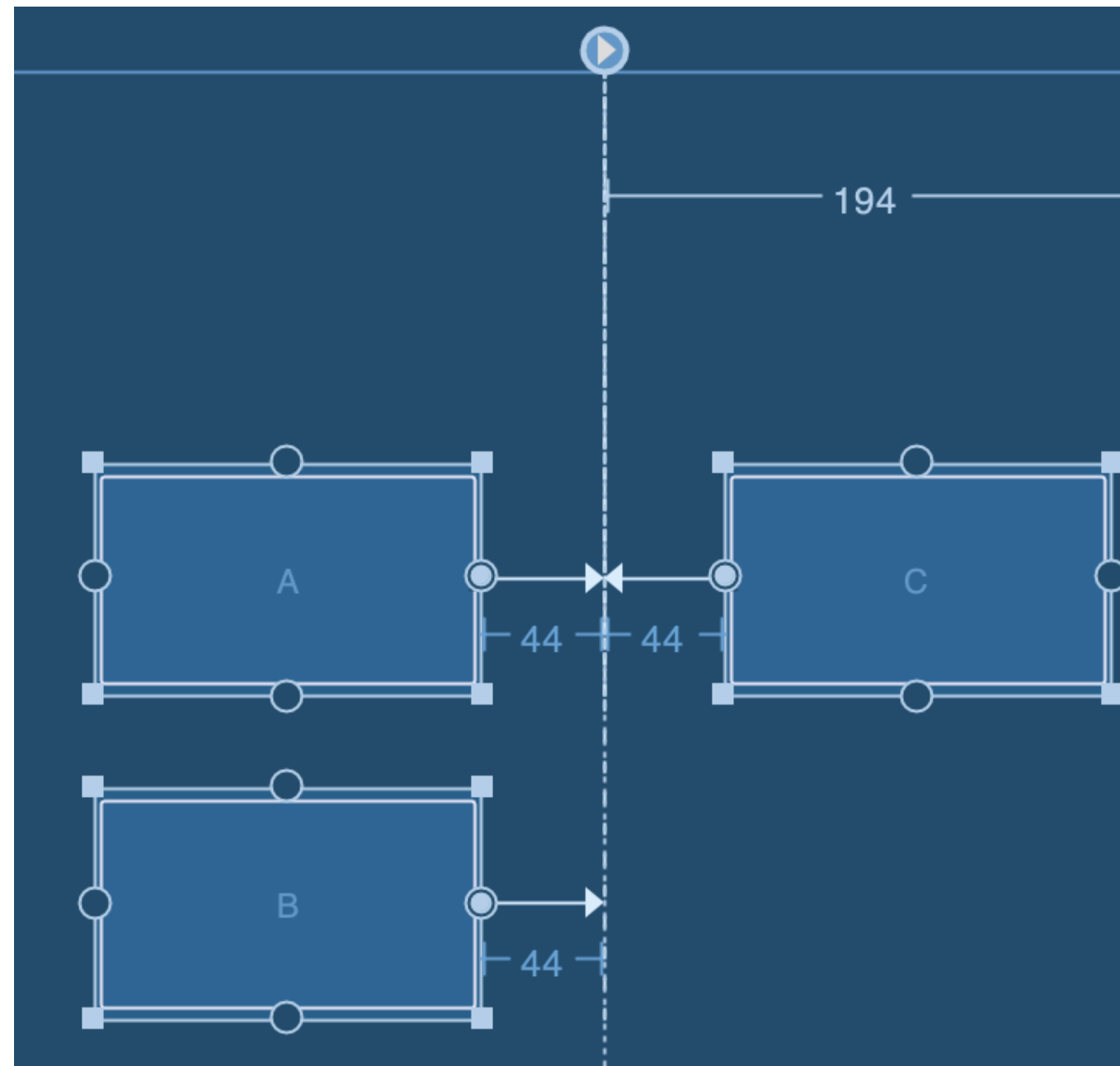
- Align the text baseline of a view to the text baseline of another view.
- In figure 8, the first line of is aligned with the text in T2
- To create a baseline constraint, select the text view you want to constrain and then click Edit Baseline , which appears below the view. Then click the text baseline and drag the line to another baseline.



```
<TextView
  android:id="@+id/T1"
  android:layout_width="130dp"
  android:layout_height="45dp"
  android:text="T1 Text"
  android:textAlignment="center"
  app:layout_constraintBaseline_toBaselineOf="@+id/T2"
  tools:layout_editor_absoluteX="65dp" />
```

```
<TextView
  android:id="@+id/T2"
  android:layout_width="130dp"
  android:layout_height="45dp"
  android:text="T2 Text"
  android:textAlignment="center"
  tools:layout_editor_absoluteX="241dp"
  tools:layout_editor_absoluteY="497dp" />
```

Constraint Behaviours: **Guideline**



- Add a vertical or horizontal guideline to which you can constrain views, and the guideline will be invisible to app users.
- Position the guideline within the layout based on either dp units or percent, relative to the layout's edge.

```
<Button
    android:id="@+id/A"
    android:layout_width="142dp"
    android:layout_height="85dp"
    android:layout_marginEnd="44dp"
    android:text="A"
    app:layout_constraintEnd_toStartOf="@+id/guideline"
    tools:layout_editor_absoluteY="144dp" />
```

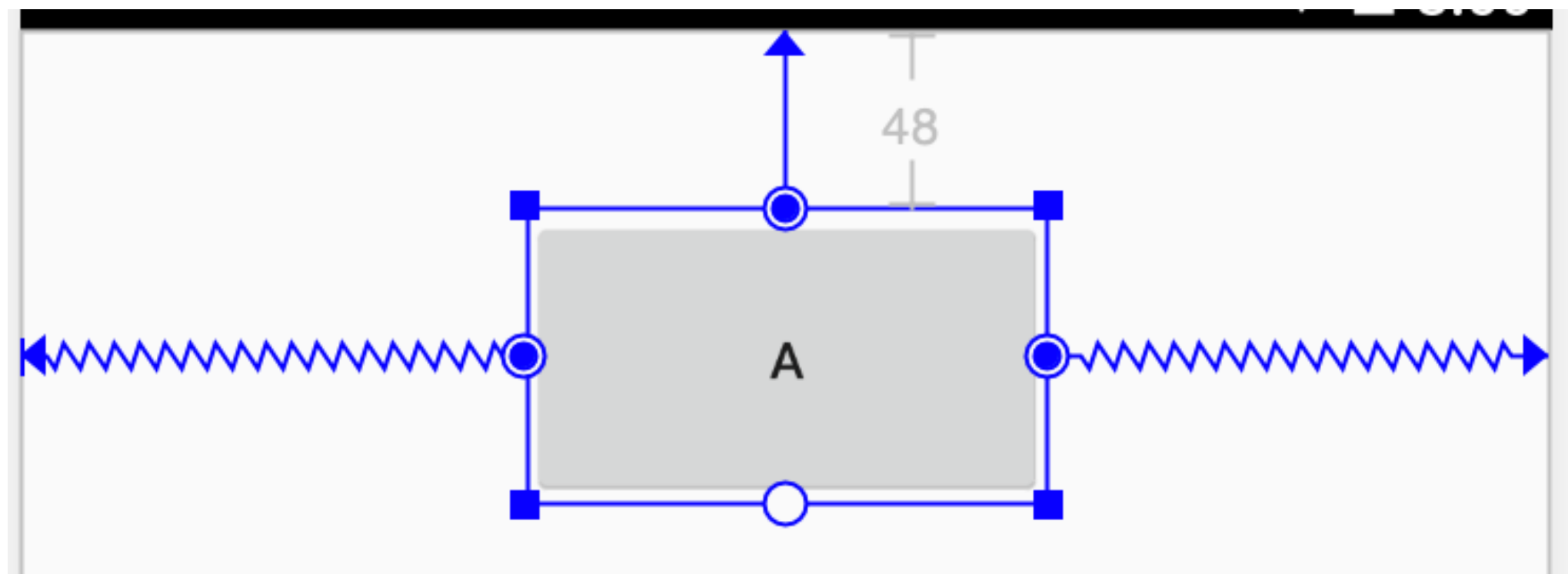
```
<Button
    android:id="@+id/B"
    android:layout_width="142dp"
    android:layout_height="85dp"
    android:layout_marginStart="44dp"
    android:text="C"
    app:layout_constraintStart_toStartOf="@+id/guideline"
    tools:layout_editor_absoluteY="144dp" />
```

```
<Button
    android:id="@+id/C"
    android:layout_width="142dp"
    android:layout_height="85dp"
    android:layout_marginEnd="44dp"
    android:text="B"
    app:layout_constraintEnd_toStartOf="@+id/guideline"
    tools:layout_editor_absoluteY="263dp" />
```

```
<android.support.constraint.Guideline
    android:id="@+id/guideline"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintGuide_end="194dp" />
```


Constraint Behaviours - Bias

- If you add opposing constraints on a view, the constraint lines become like a spring to indicate the opposing forces.
- The view becomes centered between the two constraints with a bias of 50% by default.

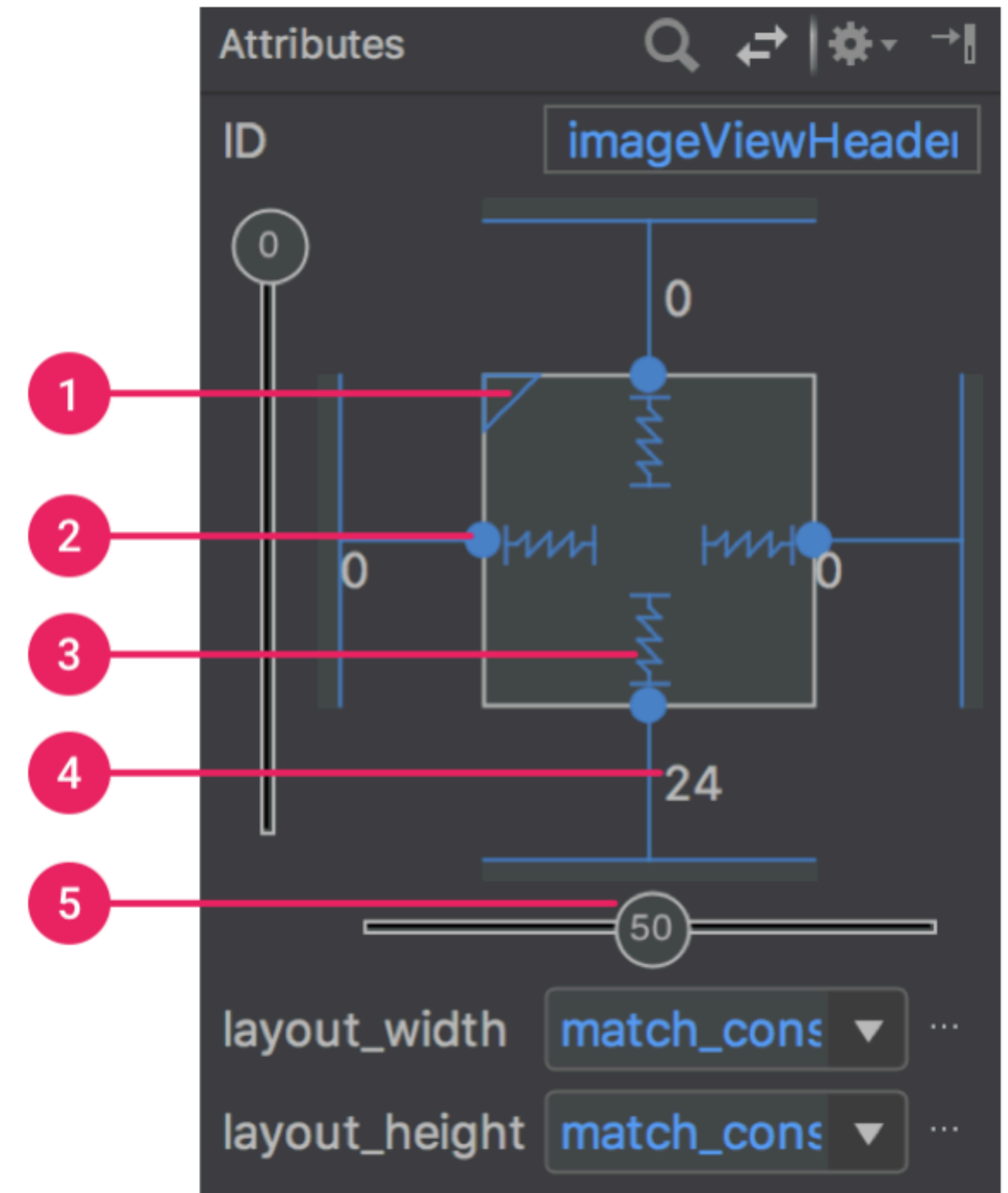


```
<Button
  android:id="@+id/A"
  android:layout_width="140dp"
  android:layout_height="80dp"
  android:layout_marginTop="48dp"
  android:text="A"
  app:layout_constraintEnd_toEndOf="parent"
  app:layout_constraintHorizontal_bias="0.5"
  app:layout_constraintStart_toStartOf="parent"
  app:layout_constraintTop_toTopOf="parent" />
```

- You can adjust the bias by dragging the view,




View Size

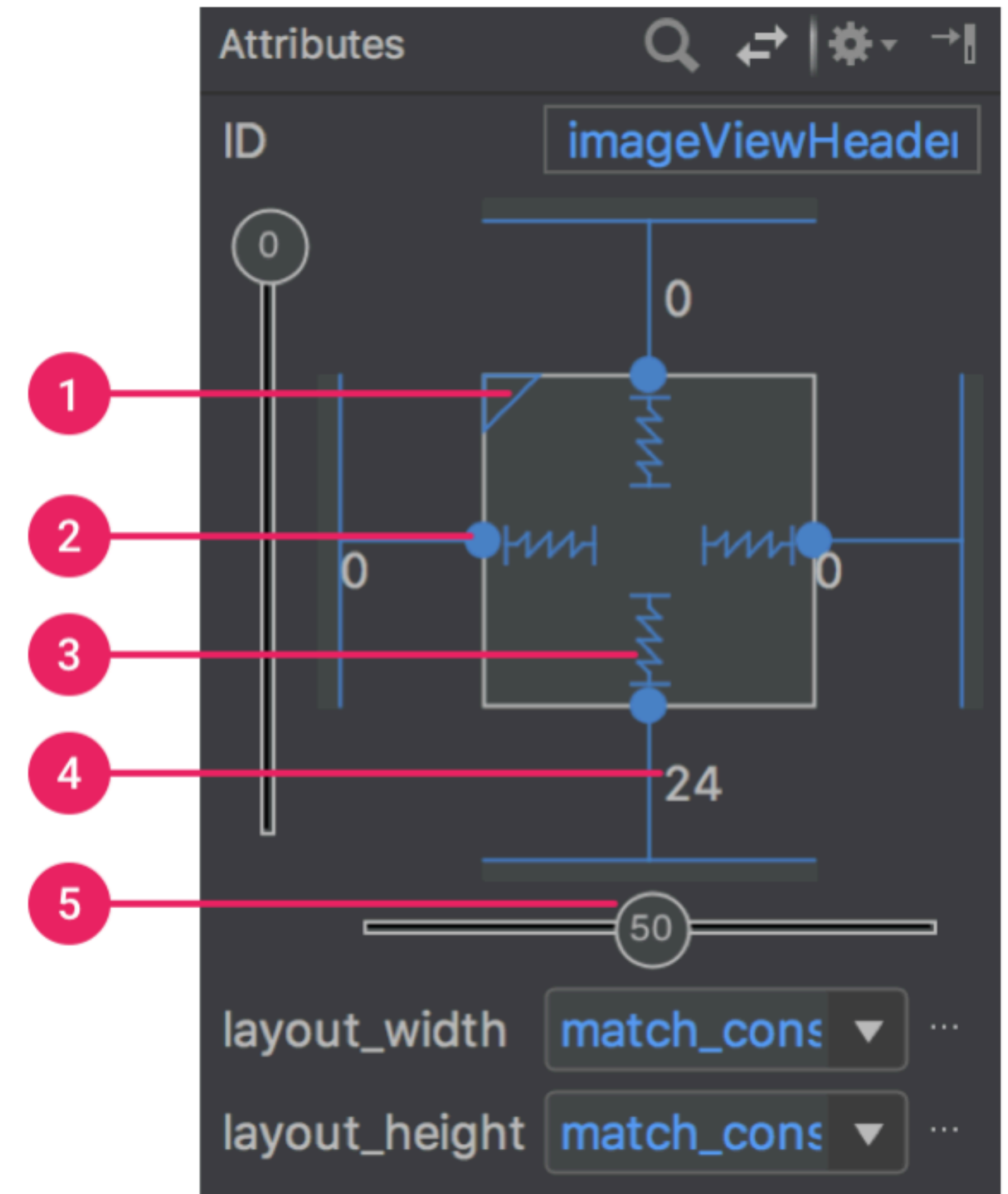
- You can use the corner handles to resize a view, but this hard codes the size so the view will not resize for different content or screen sizes.
- To select a different sizing mode, click a view and open the Attributes window on the right side of the editor.
- Near the top of the Attributes window is the view inspector, which includes controls for several layout attributes



The **Attributes** window includes controls for **1** size ratio, **2** delete constraint, **3** height/width mode, **4** margins, and **5** constraint bias.

View Size

-  **Fixed:** You specify a specific dimension in the text box below or by resizing the view in the editor.
-  **Wrap Content:** The view expands only as much as needed to fit its contents.
-  **Match Constraints:** The view expands as much as possible to meet the constraints on each side (after accounting for the view's margins). However, you can modify that behavior with the following attributes and values (these attributes take effect only when you set the view width to match constraints):



The **Attributes** window includes controls for **1** size ratio, **2** delete constraint, **3** height/width mode, **4** margins, and **5** constraint bias.