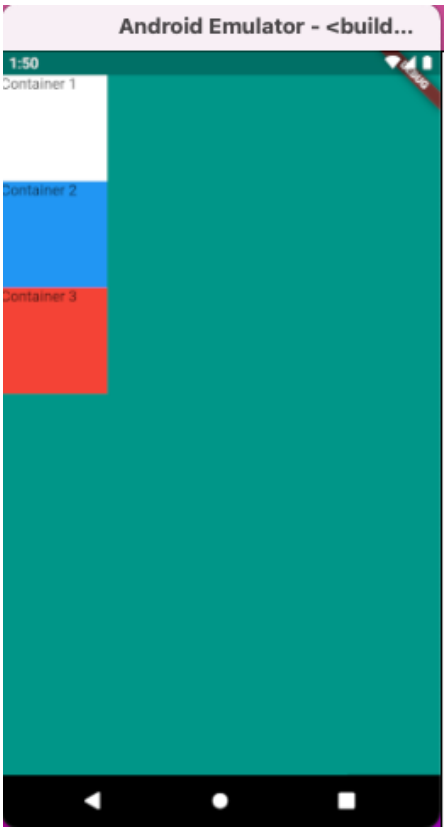
Column () and Row () widget:

The rows and columns are not a single widget; they are two different widgets, namely Row and Column. Here, we will integrate these two widgets together because they have similar properties that help us understand them efficiently and quickly.

Row and column are the two essential widgets in Flutter that allows developers to **align children horizontally and vertically according to our needs**. These widgets are very necessary when we design the application user interface in Flutter.

Let's start with the practical session: In previous topic we have added a single container but if want to add the 3 containers in a screen. How can I do that.

To do the above task, a multiple child widget- Column and Row widget come into existence which helps us to add the multiple containers.



main.dart

lib > main.dart > main

```dart
import 'package:flutter/material.dart';

Run | Debug | Profile
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        backgroundColor: Colors.teal,
        body: SafeArea(
          child: Column(
            children: [
              Container(
                width: 100,
                height: 100,
                color: Colors.white,
                child: const Text('Container 1'),
              ), // Container
              Container(
                width: 100,
                height: 100,
                color: Colors.blue,
                child: const Text('Container 2'),
              ), // Container
              Container(
                width: 100,
                height: 100,
                color: Colors.red,
                child:const Text('Container 3'),
              ), // Container
            ],
          )), // Column // SafeArea
      ), // Scaffold
    ); // MaterialApp
  }
}
```

Now, replace the Column widget with the Row widget and see the output.

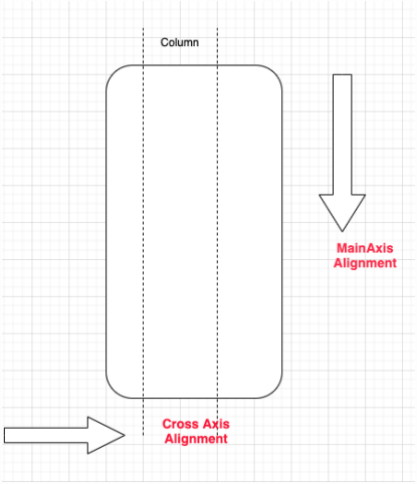
Notice the difference that you will see the output.

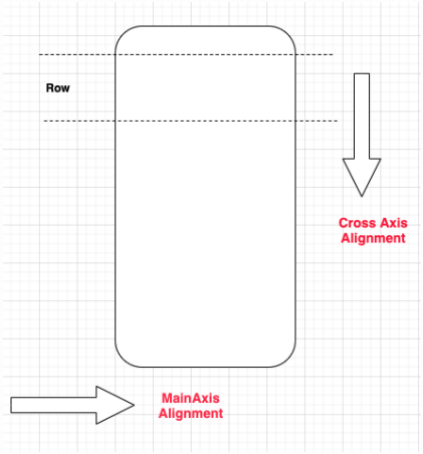
Column- representing a vertical- axis.

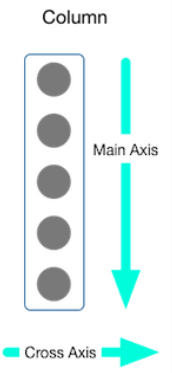
Row- representing a horizontal- axis.

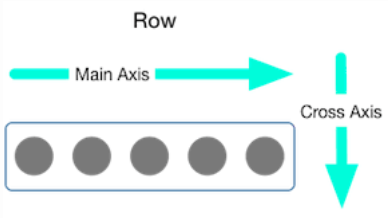
Column and Row have the same properties. So, in the examples below we are working in the same time with both widgets.

The difference of Column and Row is main axis and cross axis alignment.









mainAxisSize: this property of Column will change the size of main axis. It has **values**-> MainAxisSize.min or MainAxisSize.max. How much space should be occupied in the main axis?

mainAxisAlignment: (spacing between the widget)this property of column will change the vertical alignment. How the children should be placed along the main axis? **Values**-> start, end, space around, space between, center.

verticalDirection: Determine the order to lay children out vertically and how to interpret start and end in the vertical direction. **Values**-> down and up.

crossAxisAlignment: How the children should be placed along the cross axis.
Note: Remember if you use this property in above code, then you will not see the changes. To see the changes and working of crossAxisAlignment just change the width of any one container more than 300. **Values**-> baseline, stretch, start, end, center

```
14        backgroundColor: Colors.teal,
15        body: SafeArea(
16          child: Column(
17          // It will take space upto the area occupy by column widget
18          // instead of taking whole column space of screen.
19          // mainAxisSize: MainAxisSize.min,
20          // mainAxisAlignment: MainAxisAlignment.spaceEvenly,
21          // verticalDirection: VerticalDirection.down,
22          crossAxisAlignment: CrossAxisAlignment.end,
23          children: [
24            Container(
```

If you want the width of a single widget inside the column (here- container) widget as wide as screen size then we will give with property as:

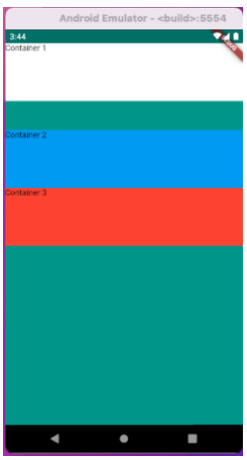
Width: double.infinity

But if you want to stretch all the containers or widget inside the column what will you do?
→ your answer will be changing the width of each container as **double.infinity.**
This may be the right way, but the easiest way is using the column property **crossAxisAlignment: CrossAxisAlignment.center**

Spacing between Widgets (in our case - containers):

Use the **SizedBox( )** widget between the container.
Note: height for column and width for row.

```
15          body: SafeArea(
16            child: Column(
17              children: [
18                Container(
19                  width: double.infinity,
20                  height: 100,
21                  color: Colors.white,
22                  child: const Text('Container 1'),
23                ), // Container
24                const SizedBox(
25                  height: 50,
26                  width: 0,
27                ), // SizedBox
28                Container(
29                  width: double.infinity,
30                  height: 100,
31                  color: Colors.blue,
32                  child: const Text('Container 2'),
33                ), // Container
34                Container(
35                  width: double.infinity,
36                  height: 100,
37                  color: Colors.red,
38                  child: const Text('Container 3'),
39                ), // Container
40              ],
41            )), // Column // SafeArea
```



We can change the widget column into row and use all those properties used for column. Notice all the changes that you have seen during the execution. Remember the things when you are using main axis alignment and cross axis alignment with row.

Workout: (Do yourself).