ConstrantLayout

https://developer.android.com/develop/ui/views/layout/constraint-layout https://www.youtube.com/watch?v=VsgXFdynDuQ





















				🗮 Code 🛛 🖩 Sp	olit 🔤 Design	ሆ 🐠 🔶 🚺	4 • E 🖸 EK
Palette	Q. \$\$ -	activity_main.xml 🗸 📚 🔍 🔍 🛛 Pixel 🗸 🔺 34 🗸	» 🔒	Attributes	Q 🌣 —		
Common	Button	◎ U _0dp , Jx 涔 指 ⊫ I.	0	Button	button2	418 (0) (1)	
Text Buttons Widgets Layouts Containers Helpers Google Legacy	 ImageButton ChipGroup Chip CheckBox RadioGroup RadioButton ToggleButton Switch FloatingActionButton 	13:00		id butto	in2		
Component Tree ConstraintLayout button "Button" button 2 Button 2' button 3 'Button 3'		ConstraintLayout	4 +	Constraints Not Horizontally Constrained Not Vertically Constrained Iayout_width	ntent V ()		+
			1:1	View			

A chain is a group of views that are linked to each other with bi-directional position constraints. The views within a chain can be distributed either vertically or horizontally.

				≣ Code	🗄 Split 🔤 Design	ს ቀ ቀ ⊡ 0	. • •	Ô N >
Palette	Q. \$\$	activity_main.xml 🗸 🔍 🔍	📮 Pixel 🗸 🗯 34 🗸 👘 😣 😣	Attributes	Q ‡ —			
Common	Button	Q U _0dp」 ♪ ※ 潟 鳥 廴	0	5	<multiple></multiple>	Carro a		
Text	ImageButton			✓ Layout			•	V21
Buttons	ChipGroup	13:00		layout width	T	Button 3		
Widgets	✓ CheckBox			layout height	a content			
Layouts	RadioGroup			visibility				
Containers	RadioButton			for a second sec				
Helpers	ToggleButton			~ visibility	•			
Google	Switch			~ Transforms				
Legacy	TotalingActionButton							
Component Tree ConstraintLayo button 'But button'But button3 'Bu button3 'Bu	ton" 9 utton 2" 9 utton 3' 9		Jx Clear Constraints of Selection Jc Constrain Jc Constrain Diganize Align Convert low J Add helpers J Add helpers Øc Convert to MotionLayout Convert view Convert view Convert view Convert view Convert view Convert Constraintesyndt Befactor X Cut Ctrl+X Gopy Ctrl+C Paste Ctrl+V Delete Delete	 Horizontal Chain Style Vertical Chain Style Create Horizontal Chain Create Vertical Chain Create Vertical Chain rotation rotationX rotationY scaleX scaleY 				+ - 13







Chains can be styled in one of the following ways:

Spread: the views are evenly distributed after margins are accounted for. This is the default.





Spread inside: the first and last views are affixed to the constraints on each end of the chain, and the rest are evenly distributed.

Packed: the views are packed together after margins are accounted for.



Weighted: when the chain is set to **spread** or **spread inside**, you can fill the remaining space by setting one or more views to "match constraints" (0dp). By default, the space is evenly distributed between each view that's set to "match constraints," but you can assign a weight of importance to each view using the layout_constraintHorizontal_weight and layout_constraintVertical_weight attributes. This works the same way as layout_weight in a linear layout: the view with the highest weight value gets the most space, and views that have the same weight get the same amount of space.

