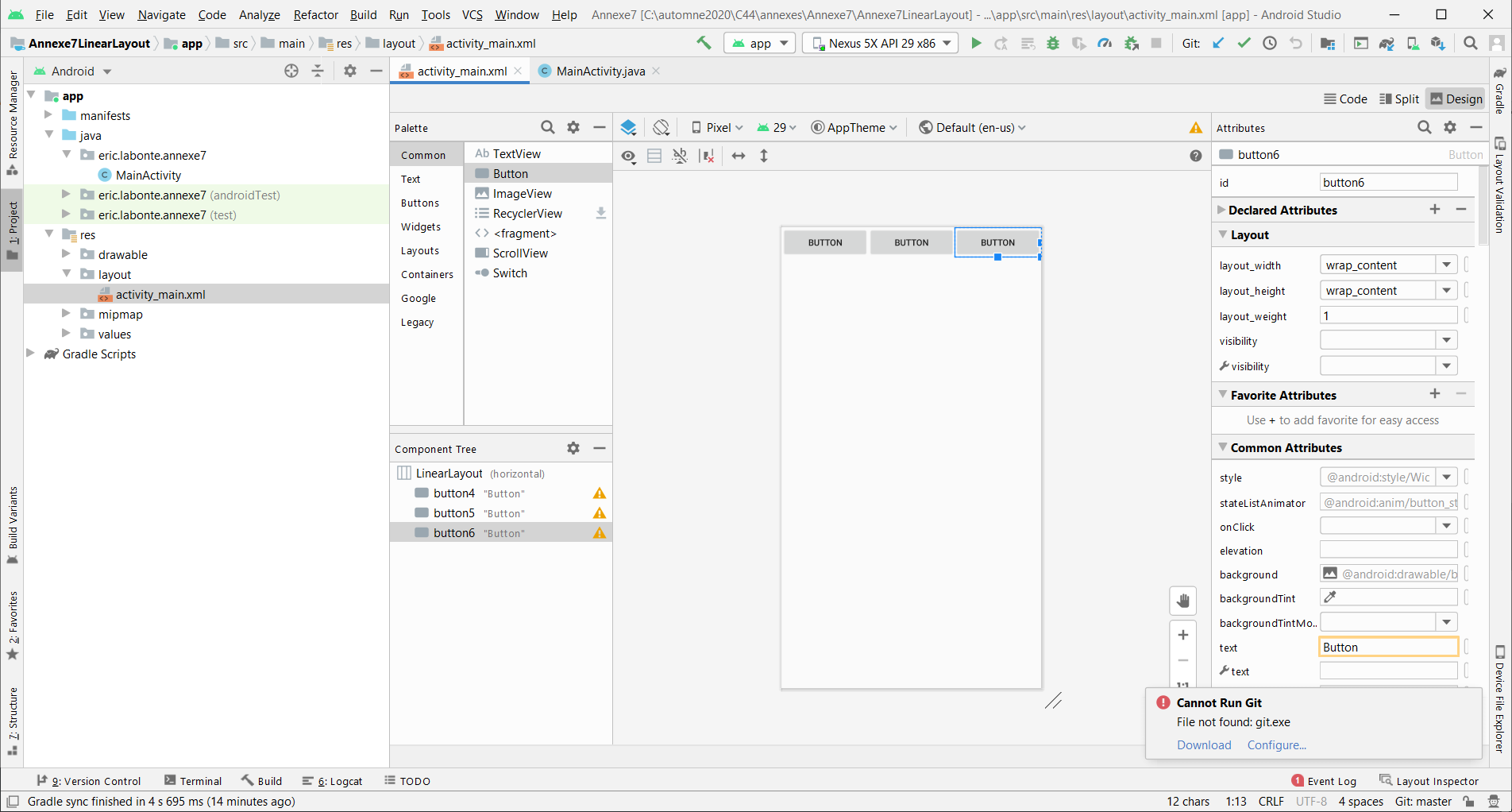
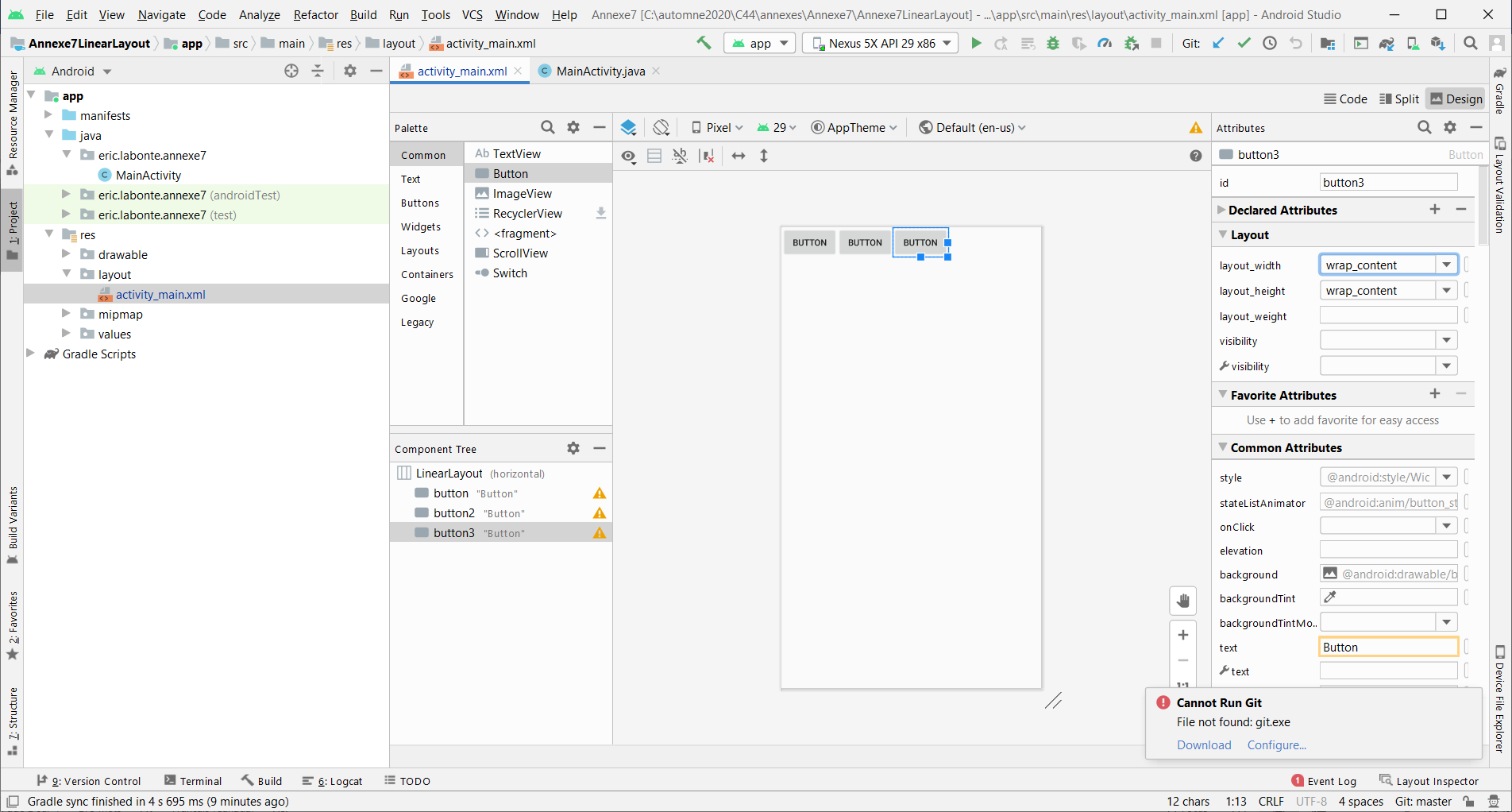
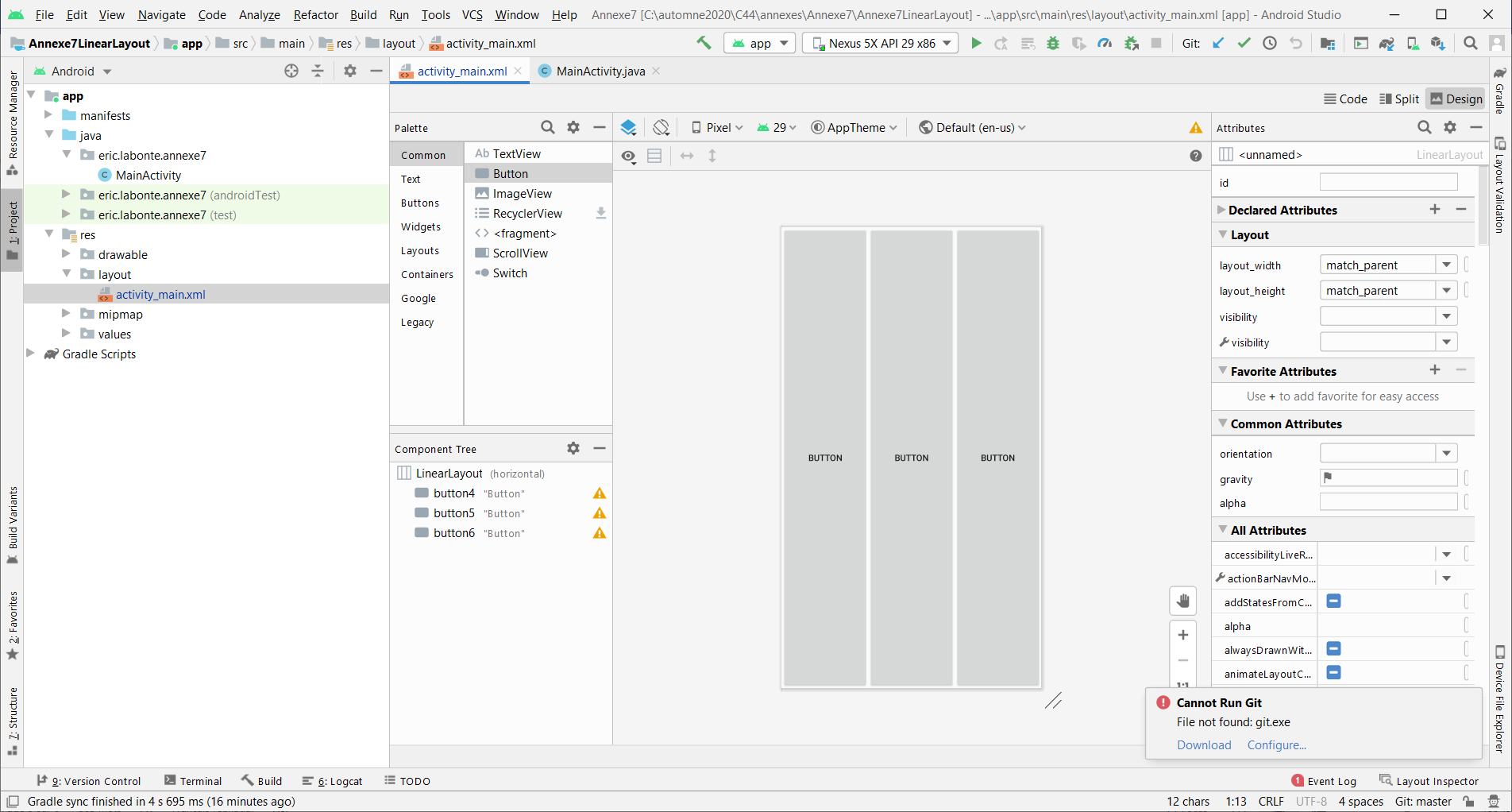
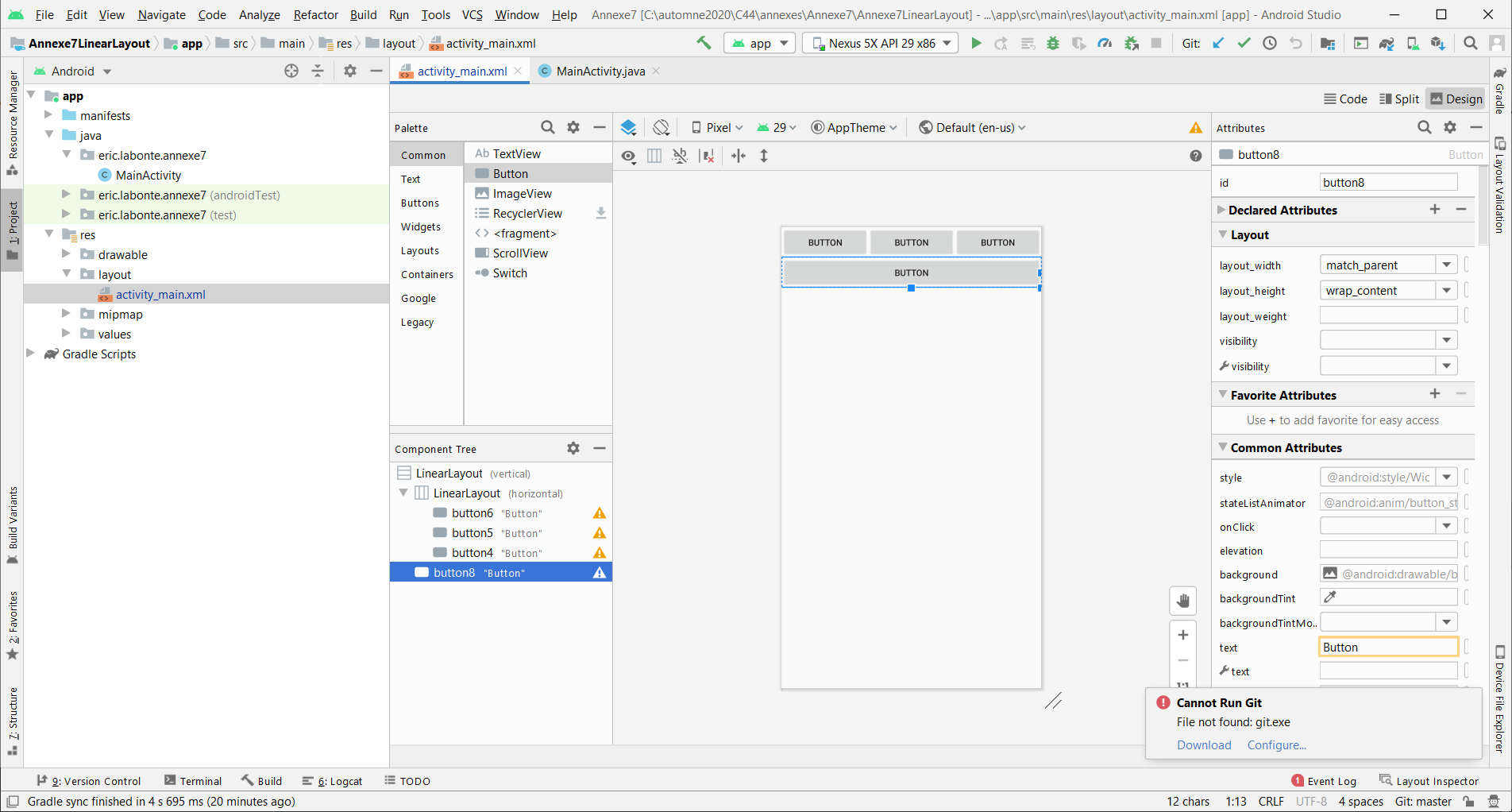
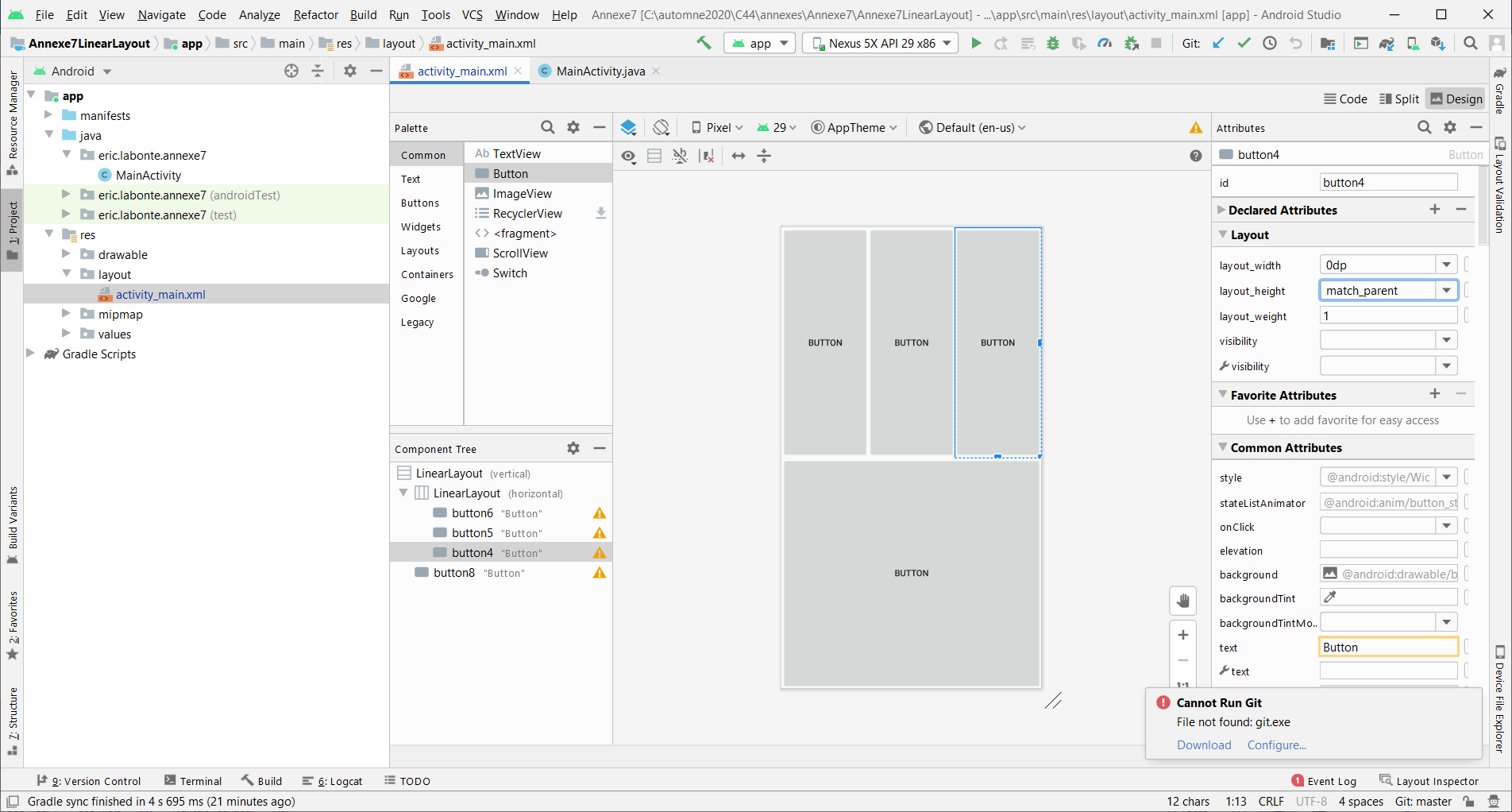
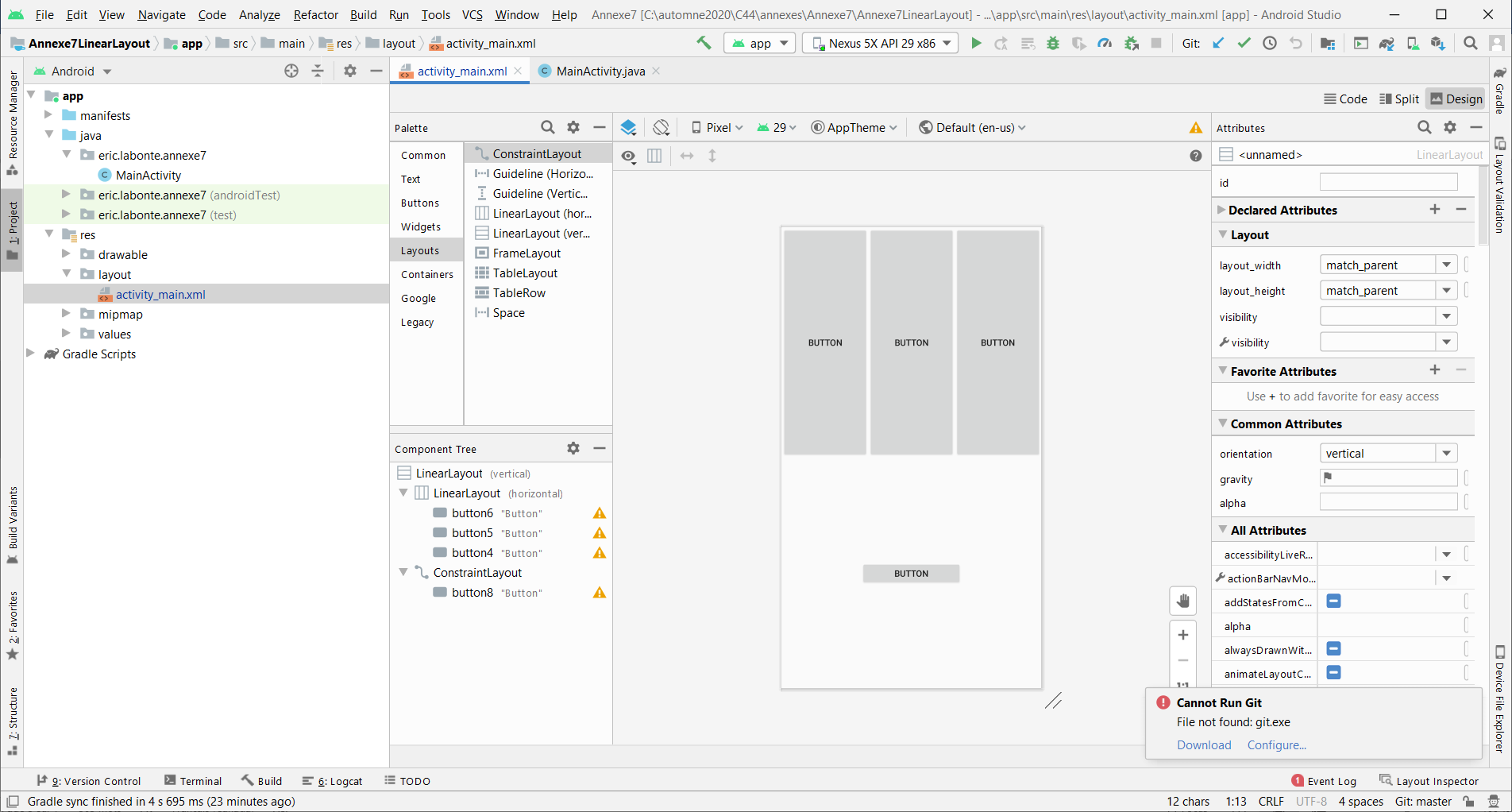
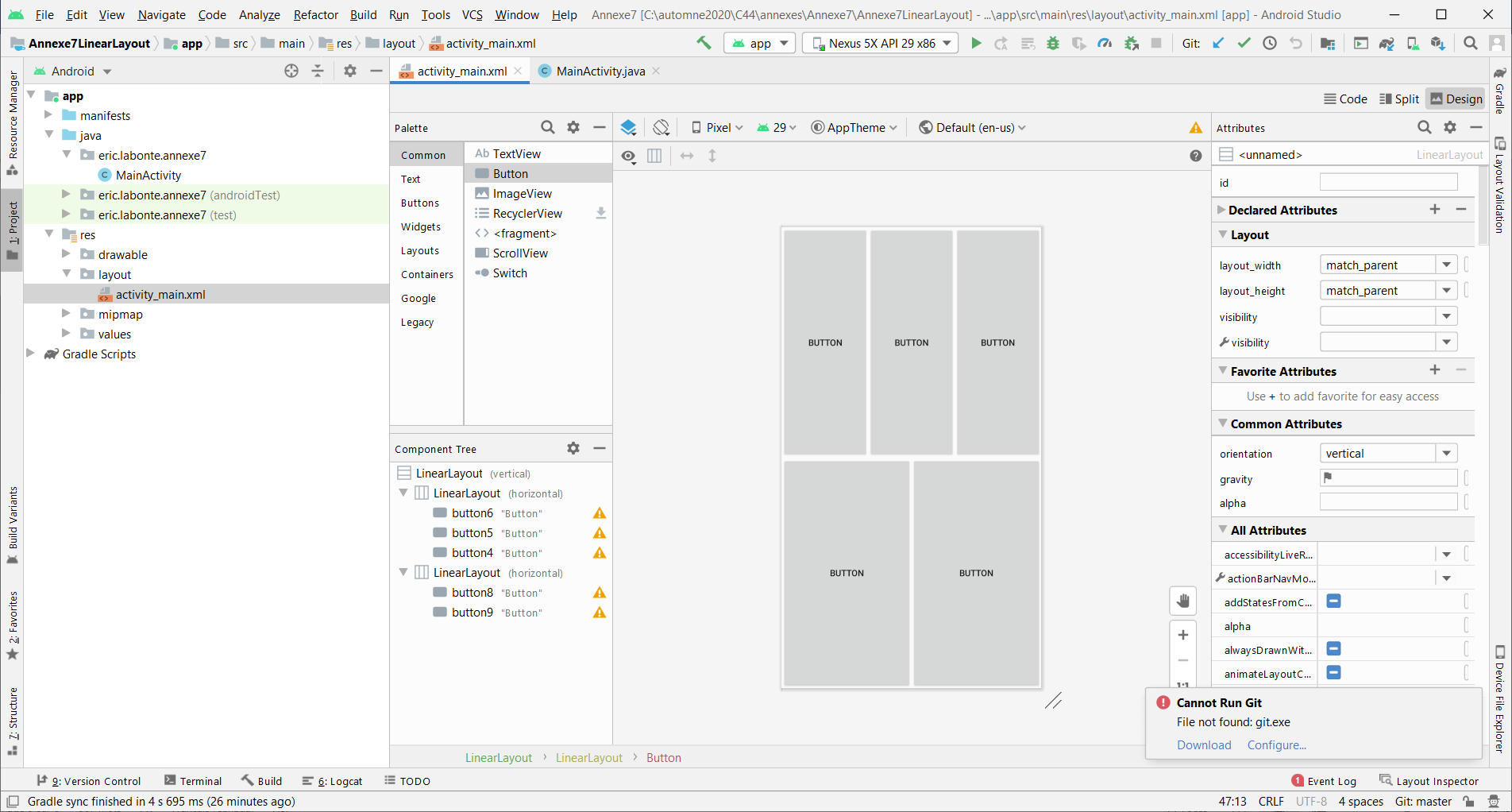
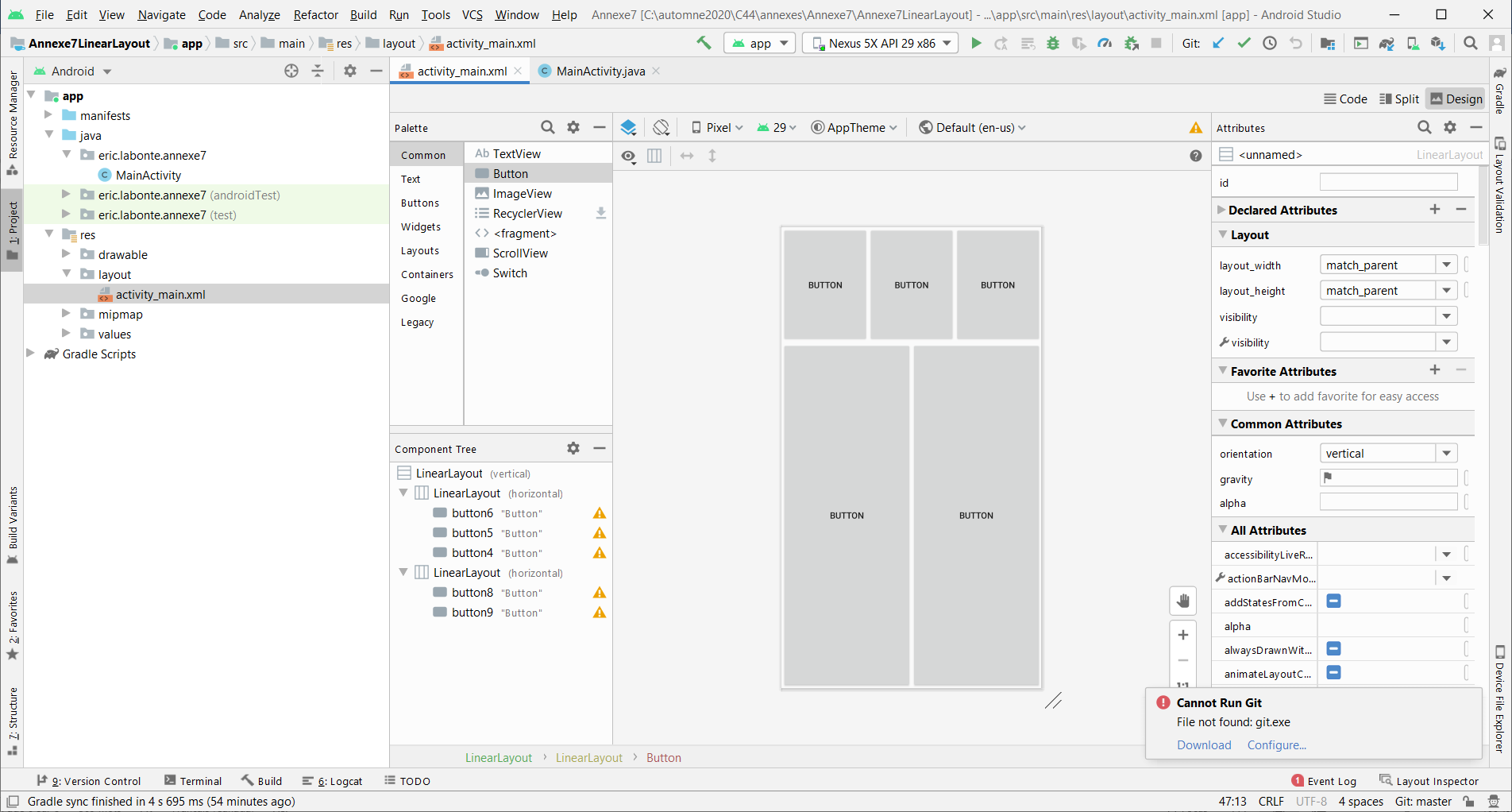
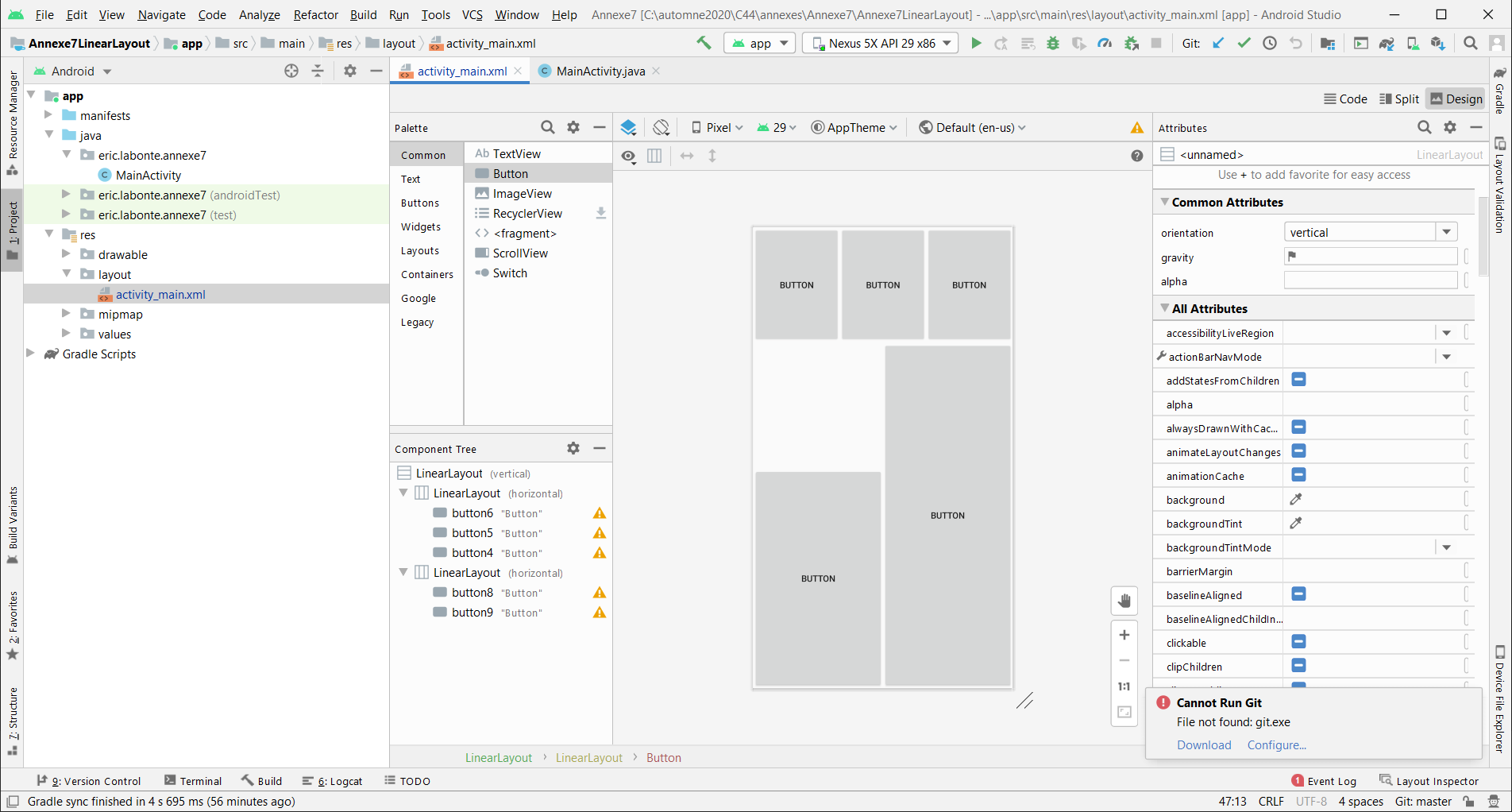
# Annexe 4 – LinearLayout / TableLayout

**LinearLayout : Un Layout où les composantes sont organisées en ligne ou en colonne**

|  |  |  |
| --- | --- | --- |
| **Propriétés XML** | **Equivalent Java** | **détails** |
| **android : orientation** | **setOrientation** | **"horizontal": 1 seul widget par colonne**  **"vertical" : 1 seul widget par ligne** |
| **android :layout\_width** | * **setLayoutParams** | **"match\_parent" : prend toute la place restante, une fois les autres widgets placés avant**  **"wrap\_content" :**  **Prend la place nécessaire ( texte présent sur le bouton )**    **Mesure en dp** |
| **android :layout\_height** | * **setLayoutParams** |
| **android :layout\_weight**  **( poids )** | * **setLayoutParams** | **Partager l’espace disponible ( pourcentage ou proportion )**  **IMPORTANT : lorsqu’on utilise des poids, on doit donner une taille de 0dp à la dimension sur laquelle on applique les poids**  *“So when android:layout\_weight is used on View X and LinearLayout is horizontal, then X's android:layout\_width is simply ignored.*  *Similar, when android:layout\_weight is used on View X and LinearLayout is vertical, then X's android:layout\_height is ignored*.  *This actually means, that you can put anything in those ignored fields: 0dp or fill\_parent or wrap\_content. It doesn't matter. But it's recommended to use 0dp so View's do not do extra calculation of their height or width (which is then ignored). This small trick simply saves CPU cycles. »* |
| **android :gravity** | **setGravity ( Gravity.LEFT ) etc.** | **Pour un parent 🡪 organise la disposition de tous les enfants contenus dans ce parent( "right", "center\_horizontal"… )** |
| **android :layout\_gravity** | **setLayoutGravity ()** | **Positionne l’élément par rapport à son parent ( si possible )** |
| **android :padding**  **android :paddingTop**  **android:layout\_marginBottom**  **Etc.** | **setPadding**  **setMargin**  **etc** | **Donne un padding ou une marge à l’élément**  **Padding : vers l’intérieur de la composante**  **Margin : vers l’extérieur de la composante**  **Padding permet de changer la forme des widgets car c’est la zone de dessin** |

## Pratique

En convertissant le conteneur ConstraintLayout initial en LinearLayout ( menu contextuel sur le ConstraintLayout dans l’arbre de composantes et faire « Convert view » ), réalisez les designs suivants :

1. 
2. 
3. 
4. 
5. 
6. 
7. 
8. 
9. 

## Exercices – Poids ( layout\_weight )

Reproduire le design de l’Activité suivante en donnant les valeurs correctes pour les attributs layout\_width, layout\_height, orientation et layout\_weight.

bouton

bouton

bouton

bouton

bouton

bouton

<**LinearLayout   
 android:orientation="vertical"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**>

<**LinearLayout   
 android:orientation="horizontal"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"**

**android:layout\_weight="1"**>

<**Button   
 android:layout\_weight="1"  
 android:layout\_width="0dp"  
 android:layout\_height="match\_parent"/**>

<**Button   
 android:layout\_weight="1"  
 android:layout\_width="0dp"  
 android:layout\_height="match\_parent"/**>

<**Button   
 android:layout\_weight="1"  
 android:layout\_width="0dp"  
 android:layout\_height="match\_parent"/**>

</**LinearLayout**>

<**LinearLayout   
 android:orientation="horizontal"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"**

**android:layout\_weight="2"**>

<**Button   
 android:layout\_weight="1"  
 android:layout\_width="0dp"  
 android:layout\_height="match parent"/**>

<**Button   
 android:layout\_weight="1"  
 android:layout\_width="0dp"  
 android:layout\_height="match content"/**>

</**LinearLayout**>

<**Button   
 android:layout\_weight="1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match-content"/**>

</**LinearLayout**>

## TableLayout

Le conteneur TableLayout est similaire au conteneur LinearLayout ( sa classe d'origine est en fait une sous-classe de LinearLayout )

Les différences sont celles-ci :

* Les enfants d'un TableLayout doivent habituellement être des objets TableRow, représentant une rangée/ligne du tableau
* Les TableRow sont organisés comme un LinearLayout horizontal 🡪 on peut donc donner des poids aux enfants des TableRow en s'assurant qu'on place la propriété layout\_width de ces enfants à 0dp.

