**PUSH NOTIFICATION**

**How to push Notification ?**

Notification is a message which appears outside of our Application’s normal UI. Through the notification, we can notify users about any important updates, events of our application. By clicking the notification user can open any activity of our application or can do some action like opening any webpage etc.

**Some Important Concepts of Push a Notification**

1. **Creating a basic notification**
2. **Creating notification channel**
3. **Adding large icon**
4. **Making notification expandable**
5. **Making notification clickable**
6. **Adding an action button to our notification**

**1. Creating a basic notification**

Tocreate a basic notification at first we need to build a notification. Now to build notification, we must use **NotificationCompat.Builder()** class where we need to pass a context of activity and a channel id as an argument while making an instance of the class.

* Kotlin

|  |
| --- |
| val nBuilder = NotificationCompat.Builder(this,CHANNEL\_ID)                      .setContentTitle(et1.text.toString())                      .setContentText(et2.text.toString())                      .setSmallIcon(R.drawable.spp\_notification\_foreground)                      .setPriority(NotificationCompat.PRIORITY\_DEFAULT)                      .build() |

Now to deliver the notification we need an object of NotificationManagerCompat class and then we notify it.

* Kotlin

|  |
| --- |
| val nManager = NotificationManagerCompat.from(this)  // Here we need to set an unique id for each  // notification and the notification Builder              nManager.notify(1, nBuilder) |

**2. Creating a notification channel**

Now to deliver notifications on android version 8.0 and above versions, we need to create a notification channel. This  **Notification Channel** concept comes from android 8.0. Here every application may have multiple channels for different types of notifications and each channel has some type of notification. Before you can deliver the notification on Android 8.0 and above versions, you must register your app’s notification channel with the system by passing an instance of **NotificationChannel** to **createNotificationChannel().**

* Kotlin

|  |
| --- |
| if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.O) {              val channel = NotificationChannel(CHANNEL\_ID, CHANNEL\_NAME, NotificationManager.IMPORTANCE\_DEFAULT).apply {                  description = CHANNEL\_DESCRIPTION}              val nManager: NotificationManager =              getSystemService(Context.NOTIFICATION\_SERVICE) as NotificationManager              nManager.createNotificationChannel(channel)  } |

**3. Adding a large icon**

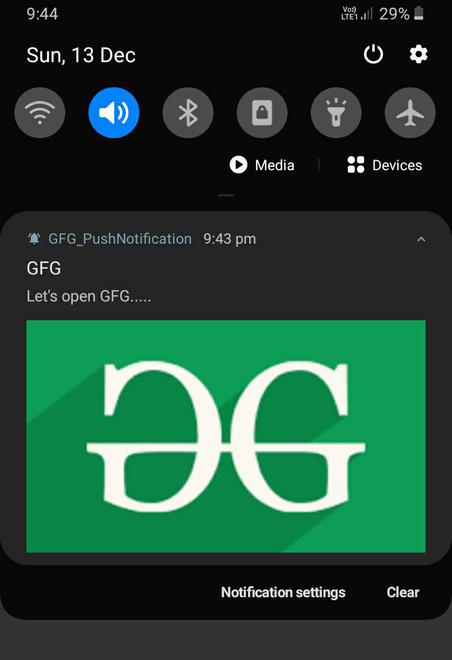
To set a large icon we use the **setLargeIcon()**method. In this method, we need to pass a Bitmap form of an image. Now to convert an image file (e.g. jpg, jpeg, png, etc.)  of the drawable folder into a Bitmap

* Kotlin

|  |
| --- |
| val imgBitmap=BitmapFactory.decodeResource(resources,R.drawable.gfg\_green)    // Building notification  val nBuilder= NotificationCompat.Builder(this,CHANNEL\_ID)                      .setContentTitle(et1.text.toString())                      .setContentText(et2.text.toString())                      .setSmallIcon(R.drawable.spp\_notification\_foreground)                      .setPriority(NotificationCompat.PRIORITY\_DEFAULT)                      // passing the Bitmap object as an argument                      .setLargeIcon(imgBitmap)                      .build() |

**4. Making notification expandable**

In the short template of the notification, large information can’t be shown. Therefore we need  to make the notification expandable like this:



to make such an expandable notification we use the **setStyle()** method on the **notification builder (nBuilder)** object. In this expanded area we can display an image, any text, different messages, etc. In our Application, we have added an image by passing the instance of the  **NotificationCompat.BigPictureStyle**class to setStyle() method.

* Kotlin

|  |
| --- |
| val imgBitmap = BitmapFactory.decodeResource(resources,R.drawable.gfg\_green)  val nBuilder = NotificationCompat.Builder(this,CHANNEL\_ID)                      .setContentTitle(et1.text.toString())                      .setContentText(et2.text.toString())                      .setSmallIcon(R.drawable.spp\_notification\_foregrond)                      .setPriority(NotificationCompat.PRIORITY\_DEFAULT)                      .setLargeIcon(imgBitmap)                      .setStyle(NotificationCompat.BigPictureStyle()                              .bigPicture(imgBitmap)                              .bigLargeIcon(null))                      .build() |

**5. Making notification clickable**

We need to make our notification clickable to perform some action by clicking the notification such as open an activity or system setting or any webpage etc. Now to perform such actions intent is needed (e.g. explicit or implicit intent). In our Application, we are making an Implicit intent to open the GFG official home page.

|  |
| --- |
| val intent1= Intent()          intent1.action=Intent.ACTION\_VIEW intent1.data=Uri.parse("<https://www.geeksforgeeks.org/>") |

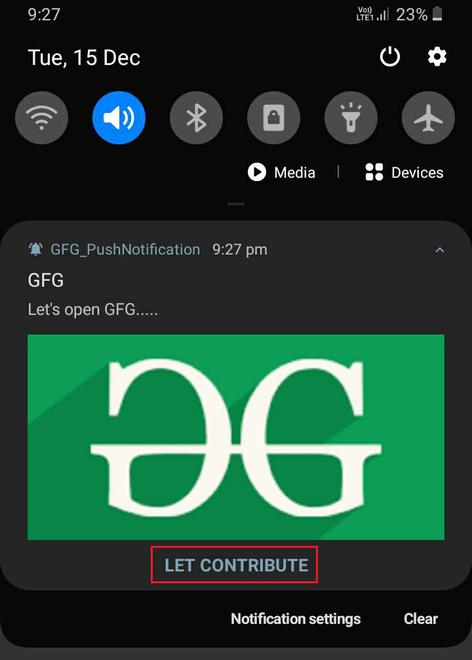
Now it is not necessary that whenever the notification will appear then the user will click it instantly, user can click it whenever he /she wants and therefore we also need to make an instance of **PendingIntent**which basically makes the intent action pending for future purpose.

* Kotlin

|  |
| --- |
| val pendingIntent1=PendingIntent.getActivity(this, 5, intent1, PendingIntent.FLAG\_UPDATE\_CURRENT)  val nBuilder = NotificationCompat.Builder(this,CHANNEL\_ID)                      .setContentTitle(et1.text.toString())                      .setContentText(et2.text.toString())                      .setSmallIcon(R.drawable.notifications)                      .setPriority(NotificationCompat.PRIORITY\_DEFAULT)                      .setLargeIcon(imgBitmap)                      .setStyle(NotificationCompat.BigPictureStyle()                              .bigPicture(imgBitmap)                              .bigLargeIcon(null)                      .setContentIntent(pendingIntent1)                      .setAutoCancel(true)                      .build() |

**6. Adding an action button to our notification**

Sometimes there exists some action button at our notification template that is used to perform some action.



Here we also need an**Intent** and a **PendingIntent**. Then we need to pass the instance of the **PendingIntent** to **addAction()** method at the time of building the notification.

* Kotlin

|  |
| --- |
| // Creating the Implicit Intent  // to open the GFG contribution page  val intent2 = Intent()          intent2.action = Intent.ACTION\_VIEW          intent2.data = Uri.parse("<https://www.geeksforgeeks.org/contribute/>")    val nBuilder = NotificationCompat.Builder(this,CHANNEL\_ID)                      .setContentTitle(et1.text.toString())                      .setContentText(et2.text.toString())                      .setSmallIcon(R.drawable.notifications)                      .setPriority(NotificationCompat.PRIORITY\_DEFAULT)                      .setLargeIcon(imgBitmap)                      .setStyle(NotificationCompat.BigPictureStyle()                              .bigPicture(imgBitmap)                              .bigLargeIcon(null))                      .setContentIntent(pendingIntent1)                      .setAutoCancel(true)                      .addAction(0,"LET CONTRIBUTE",pendingIntent2)                      .build() |