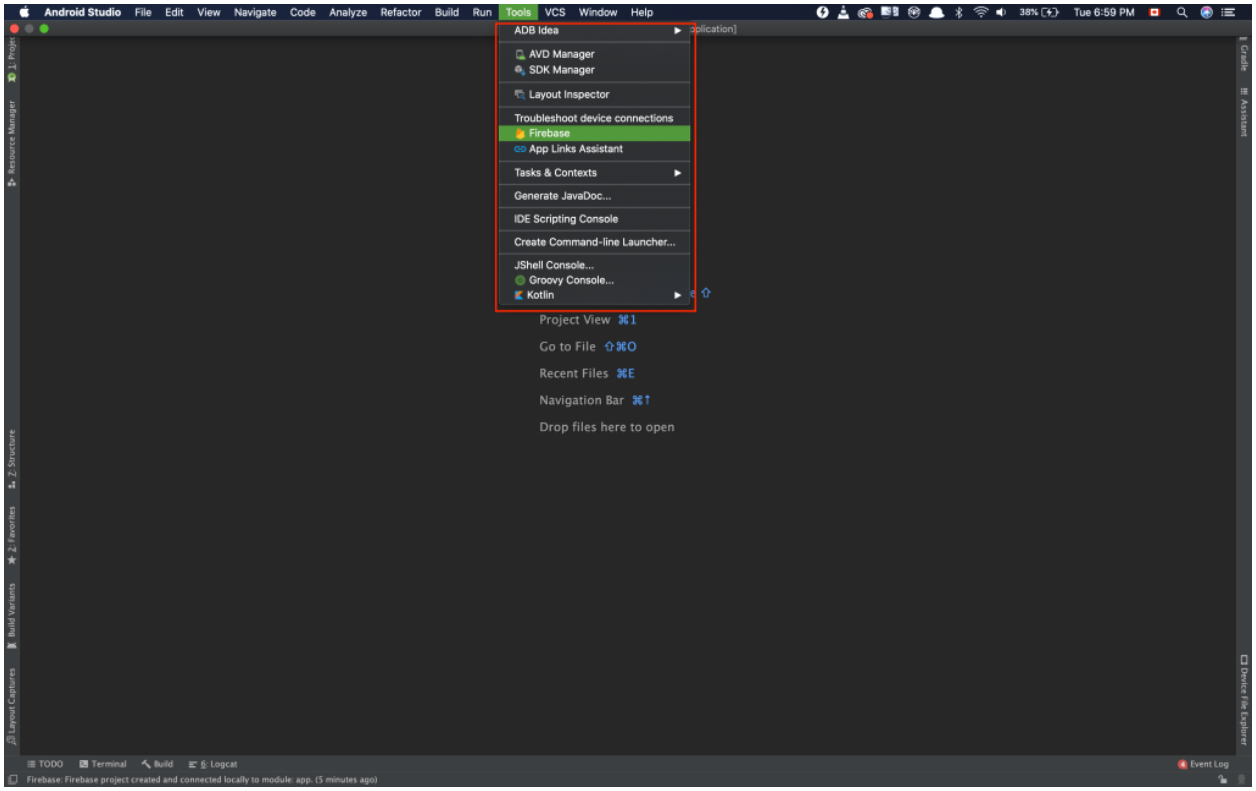
Firebase Storage Tutorial

Incorporating firebase storage in your app can be divided into 4 steps. They are:

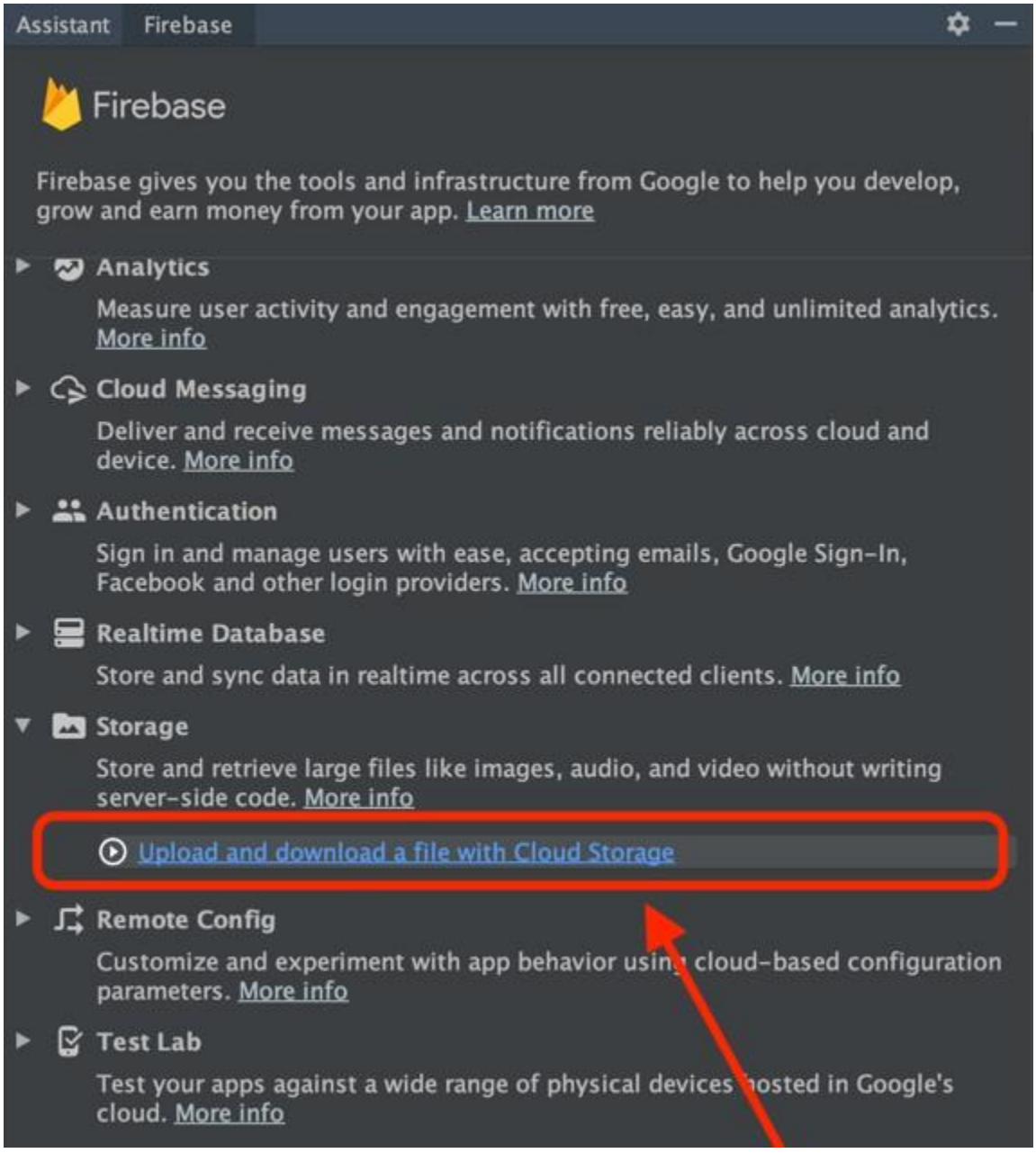
1. Setting up firebase storage
2. Creating a reference
3. Uploading a file
4. Downloading a file

Part 1: Setting up firebase storage

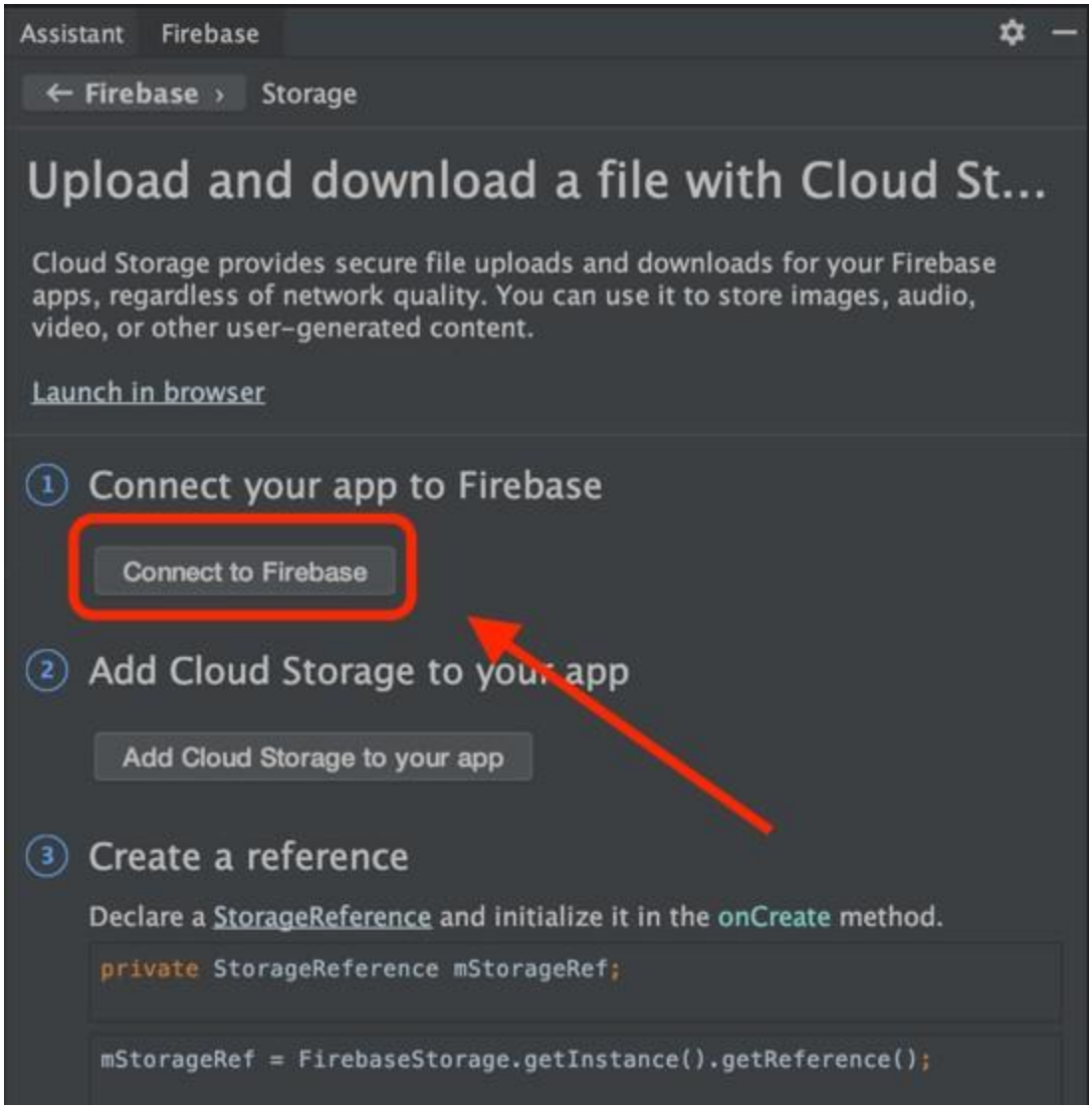
1) Create or open an Android Studio Project and open Firebase Assistant in Android Studio. You can access it here: Tools>Firebase.



2) It will open a pane on the right side, containing all the Firebase related products. We are only interested in Firebase Cloud Storage so we will expand the “Storage” dropdown. You will see a button named “Upload and download a file with Cloud Storage” click that button.

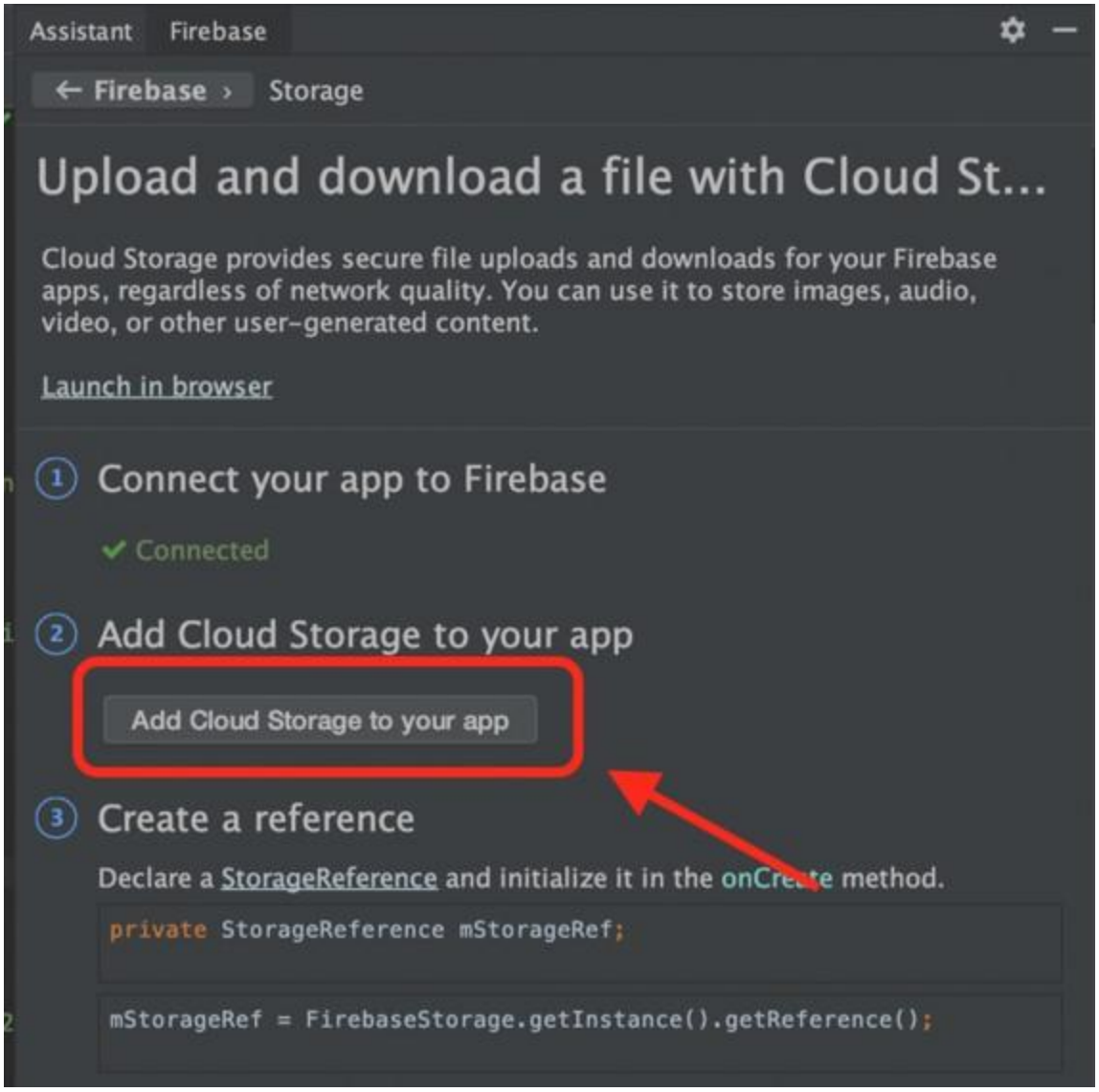


3) Now the first option you will see is, “Connect to Firebase” just click that. It will ask you to log in if you are not already.



4) After logging in, you will be asked to create a new or choose an existing Firebase Project. Do what you like and hit the “Connect to Firebase” button.

5) Click the “Add Cloud Storage to your app”, accept the changes that are going to be made and done!

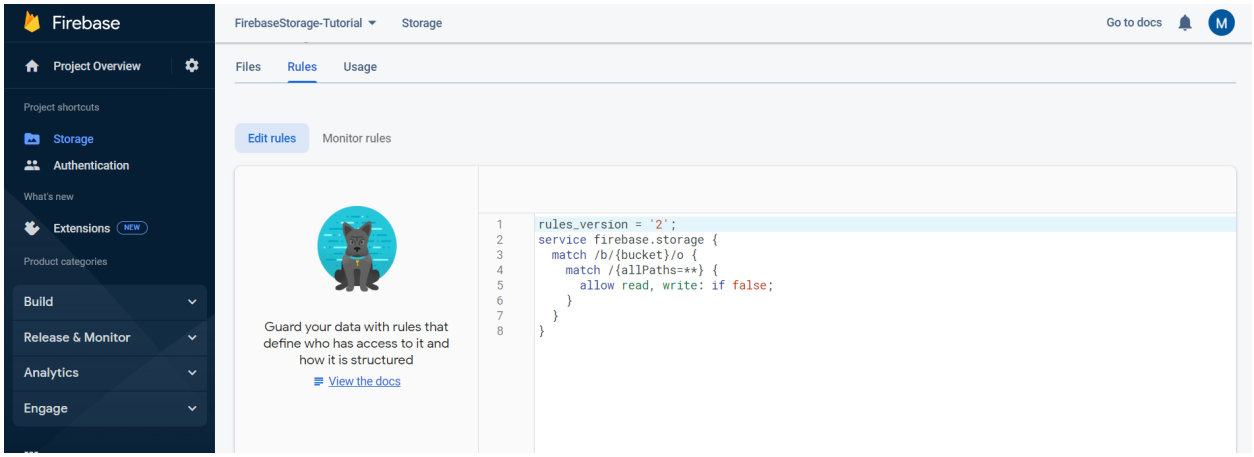


6) Once you have set up Firebase, add Storage SDK to app-level build.gradle and press Sync to finish.

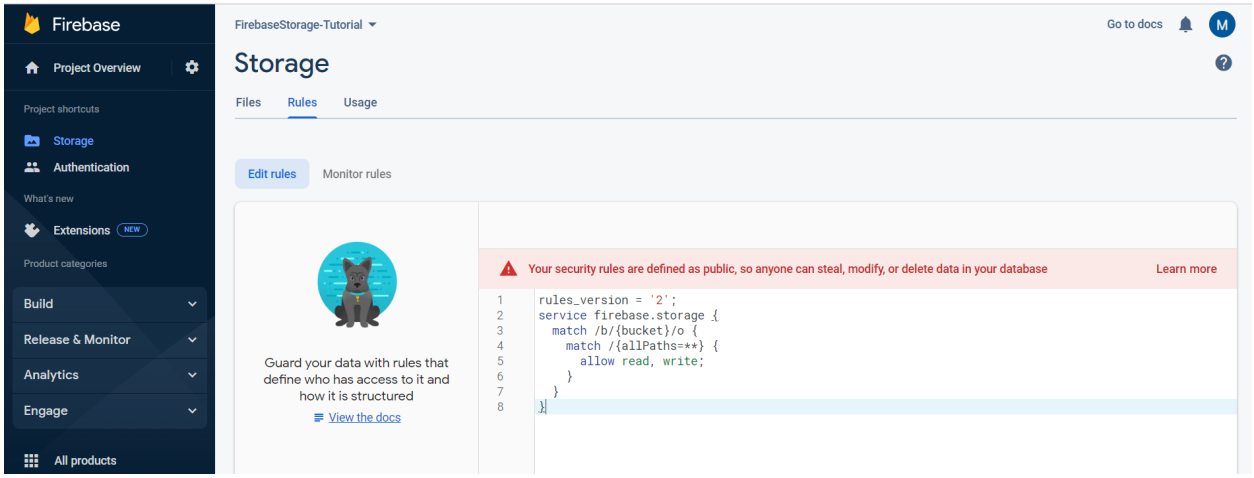
```
dependencies {
    compile 'com.google.firebase:firebase-storage :16.1.0'
}
```

6) To access files in Cloud Storage for Firebase, both read and write, we usually need Authentication via Firebase Authentication first, but so that we can understand this article without reading Firebase Authentication. We're going to make it publicly accessible by going to the Firebase Console for the project. Then

select the Storage menu and select the tab named RULES. You will see something like this.



Then modify the code to make the file publicly accessible as follows.



Part 2: Creating a reference

All the files we upload are stored in a bucket. The structure is the same as storing files on a hard disk. folders and files for upload, download, delete and metadata management.

1) Start by declaring variables. StorageReference Get Instance and refer to bucket.

```
StorageReference storageRef = FirebaseStorage . getInstance (). 
getReference ();
```

2) We can refer to folders and files in descending order with child()

```
StorageReference folderRef = storageRef. child ("photos");
StorageReference imageRef = storageRef. child 
("photos/firebase.png");
```

3) And if we want to see the values with the variable we are referring to, we can getPath() , getName() , and getBucket() to retrieve the String value.

```
// path reference: "photos/firebase.png"
imageRef. getPath() ;

// end of path: "firebase.png"
imageRef. getName() ;

// name of the bucket where we store the file: "xxx-xxxx
-xxxxx.appspot.com"
imageRef.getBucket () ;
```

Part 3: Upload from a local file

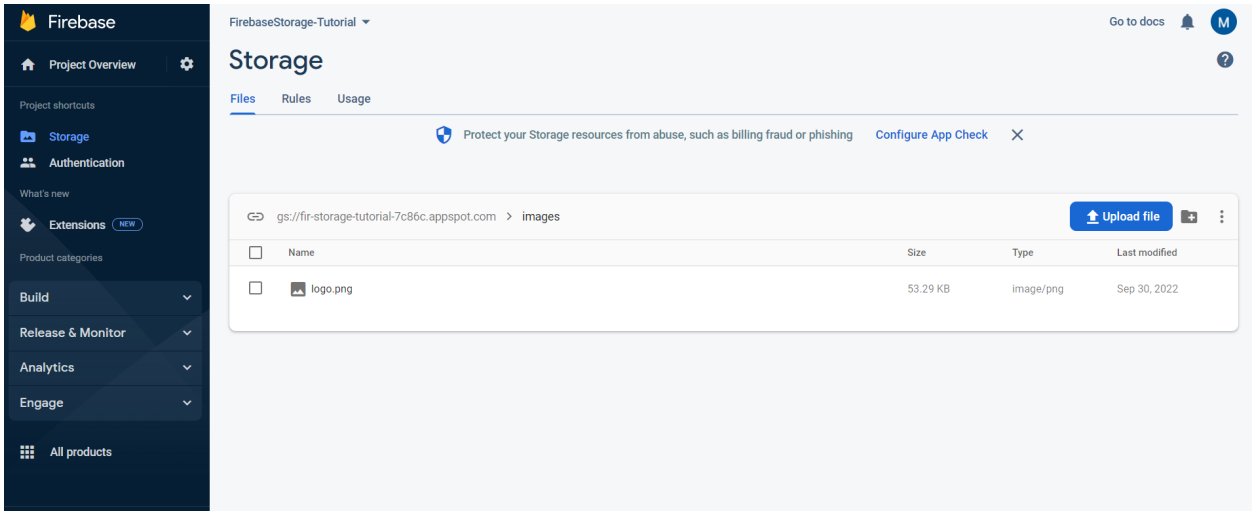
1) Select local files to upload to Cloud Storage for Firebase.

2) I extracted the path from the selected image in the gallery, converted it into File format and made it into Uri , and then used the putFile() command to upload it. If successful, it would take the URL from Cloud Storage for Firebase.

```
Uri file = Uri.fromFile(new File(path));
StorageReference imageRef =
folderRef.child(file.getLastPathSegment());
mUploadTask = imageRef.putFile(file);

mUploadTask.addOnFailureListener(new OnFailureListener() {
@Override
public void onFailure(@NonNull Exception exception) {
    Helper.dismissDialog();
    mTextView.setText(String.format("Failure: %s",
exception.getMessage()));
}
}).addOnSuccessListener(new
OnSuccessListener<UploadTask.TaskSnapshot>() {
@Override
public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
    Helper.dismissDialog();
    imageRef.getDownloadUrl().addOnSuccessListener(new
OnSuccessListener<Uri>() {
    @Override
    public void onSuccess(Uri uri) {
        mTextView.setText(uri.toString());
    }
    });
}
});
```

3) We can also view all files in Firebase Console by going to the Storage menu and then selecting the first tab named Files. You will see the photos folder and the filename.png (here it is logo.png) file inside the folder.



Part 4: Download to a local file

1) This method requires asking permission WRITE_EXTERNAL_STORAGE.

```
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

2) We can access the files when we are offline and can send the files further to other apps.

```
File dir = new File(Environment.getExternalStorageDirectory() +
"/photos");
final File file = new File(dir, UUID.randomUUID().toString() +
".png");
try {
if (!dir.exists()) {
    dir.mkdir();
}
```

```
file.createNewFile();
} catch (IOException e) {
e.printStackTrace();
}

final FileDownloadTask fileDownloadTask =
imageRef.getFile(file);
Helper.initProgressDialog(this);
Helper.mProgressDialog.setButton(DialogInterface.BUTTON_NEGATIVE
, "Cancel", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i)
{
        fileDownloadTask.cancel();
    }
});
Helper.mProgressDialog.show();

fileDownloadTask.addOnSuccessListener(new
OnSuccessListener<FileDownloadTask.TaskSnapshot>() {
    @Override
    public void onSuccess(FileDownloadTask.TaskSnapshot
taskSnapshot) {
        Helper.dismissProgressDialog();
        mTextView.setText(file.getPath());
        mImageView.setImageURI(Uri.fromFile(file));
    }
}).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception exception) {
        Helper.dismissProgressDialog();
        mTextView.setText(String.format("Failure: %s",
exception.getMessage()));
    }
}).addOnProgressListener(new
OnProgressListener<FileDownloadTask.TaskSnapshot>() {
```

```
    @Override
    public void onProgress(FileDownloadTask.TaskSnapshot
taskSnapshot) {
        int progress = (int) ((100 *
taskSnapshot.getBytesTransferred()) /
taskSnapshot.getTotalByteCount());
        Helper.setProgress(progress);
    }
});
```

3) This download method allows you to add a listener to view progress, so we can make a ProgressDialog.

References :

1) https://code.tutsplus.com/tutorials/firebase-for-android-file-storage--cms-27376
2) https://www.ashutec.com/blog/what-is-google-firebase-storage-and-how-to-use-it-for-application-development-4f9f462d1487
3) https://www.youtube.com/watch?v=ZmgncLHk_s4

Prepared by:

1) Yeamin Kaiser (01)
2) Tasnim Bin Anwar (05)
3) Mohima Ahmed Joyee (42)