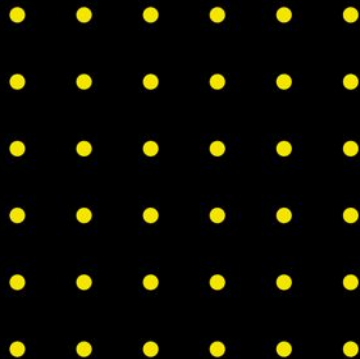
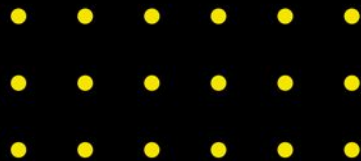




Firestore suite





Build

Accelerate app development
with fully managed backend
infrastructure

[View all build products](#)

 Cloud Firestore

 Authentication



Release & Monitor

Release with confidence and
monitor performance and
stability

[View all release & monitor
products](#)

 Crashlytics

 Google Analytics



Engage

Boost user engagement with
rich analytics, A/B testing, and
messaging campaigns

[View all engage products](#)

 Remote Config

 Cloud Messaging

Build

Authentication

Authentication

- Email & password / Email & link
- Google / Apple / Facebook / Twitter / Github
- Phone number & SMS
- Anonymous user
- Custom auth provider

```
try {
    UserCredential userCredential = await FirebaseAuth.instance.signInWithEmailAndPassword(
        email: "barry.allen@example.com",
        password: "SuperSecretPassword!"
    );
} on FirebaseAuthException catch (e) {
    if (e.code == 'user-not-found') {
        print('No user found for that email.');
```

```
    } else if (e.code == 'wrong-password') {
```

```
        print('Wrong password provided for that user.');
```

```
    }
```

```
}
```

Cloud Firestore

Cloud Firestore

- NoSQL document database
- Flexible & scalable - you just use it (but also pay)
- Ordering & limiting
- Compound queries & indexing
- Security connected with Firebase Auth


```
@override
Widget build(BuildContext context) {
  // Create a CollectionReference called users that references the firestore collection
  CollectionReference users = FirebaseFirestore.instance.collection('users');

  Future<void> addUser() {
    // Call the user's CollectionReference to add a new user
    return users
      .add({
        'full_name': fullName, // John Doe
        'company': company, // Stokes and Sons
        'age': age // 42
      })
      .then((value) => print("User Added"))
      .catchError((error) => print("Failed to add user: $error"));
  }

  return TextButton(
    onPressed: addUser,
    child: Text(
      "Add User",
    ),
  );
}
```

Realtime Database

Realtime Database

- Original Firebase DB
- Cloud Firestore emerged from this
- Focused on low-latency synchronization
- Simple queries only

Storage

Storage

- File storage for user-generated content
- Robust operations (connectivity awareness)
- Integrated with Firebase Auth
- Scalability

```
import 'package:path_provider/path_provider.dart';

Future<void> downloadFileExample() async {
  Directory appDocDir = await getApplicationDocumentsDirectory();
  File downloadToFile = File('${appDocDir.path}/download-logo.png');

  try {
    await firebase_storage.FirebaseStorage.instance
      .ref('uploads/logo.png')
      .writeToFile(downloadToFile);
  } on firebase_core.FirebaseException catch (e) {
    // e.g, e.code == 'canceled'
  }
}
```

Machine Learning

Machine Learning

- Production-ready models for popular use cases (recognize text, label images, recognize landmarks)
- Hosting custom TensorFlow Lite models
- Training custom models in the cloud (using your uploaded data)

Hosting

Hosting

- SSL for free
- Static HTML (Flutter Web) or redirect to Cloud Function
- Easy upload (through Firebase CLI)
- Cache / CDN included



```
$ firebase deploy --only hosting
```

Cloud Functions

Cloud Functions

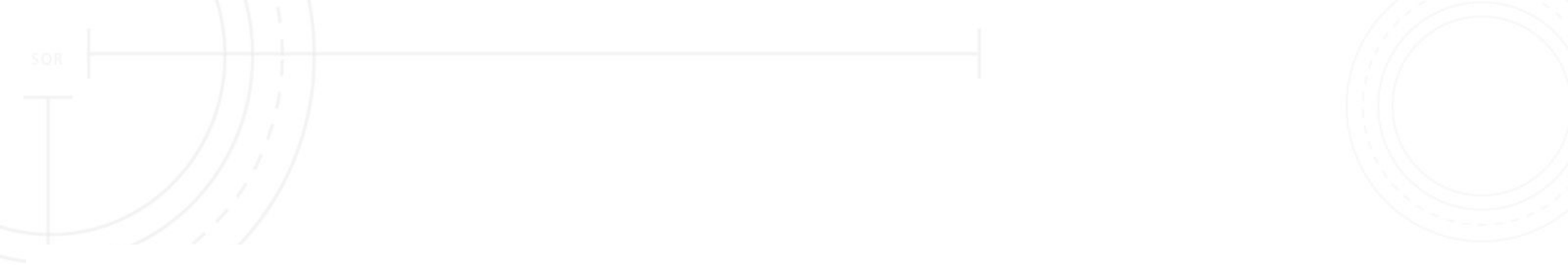
- Integrated with Firebase (Auth / Storage / Firestore triggers)
- Serverless = zero maintenance
- Look for Google Cloud Functions vs Cloud Functions for Firebase because they are **NOT** the same

```
exports.addMessage = functions.https.onRequest(async (req, res) => {  
  // Grab the text parameter.  
  const original = req.query.text;  
  // Push the new message into Firestore using the Firebase Admin SDK.  
  const writeResult = await admin.firestore().collection('messages').add({original: original});  
  // Send back a message that we've successfully written the message  
  res.json({result: `Message with ID: ${writeResult.id} added.`});  
});
```



```
// Always change the value of "/hello" to "world!"
exports.hello = functions.database.ref('/hello').onWrite(event => {
  // set() returns a promise. We keep the function alive by returning it.
  return event.data.ref.set('world!').then(() => {
    console.log('Write succeeded!');
  });
});
```

AppCheck








- 
1. Your app interacts with the provider of your choice to obtain an attestation of the app or device's authenticity (or both, depending on the provider).
 2. The attestation is sent to the App Check server, which verifies the validity of the attestation using parameters registered with the app, and returns to your app an App Check token with an expiration time. This token might retain some information about the attestation material it verified.
 3. The App Check client SDK caches the token in your app, ready to be sent along with any requests your app makes to protected services.

AppCheck

DeviceCheck and AppAttest on iOS
SafetyNet on Android
reCAPTCHA on Web

Extensions



Categories

- ☐  Marketing
- ☐  Messaging
- ☐  Payments
- ☐  Search
- ☐  Shipping
- ☐  Social
- ☐  Utilities

Trending

- ☐  Spotlight
- ☐  Popular

Firebase products

- ☐  Authentication
- ☐  Cloud Firestore
- ☐  Cloud Storage
- ☐  Realtime Database



Search with Algolia

Made by [Algolia](#) 

Enables full text search of your Cloud Firestore data with Algolia.

 Search

Install

[View details](#)



Search with Elastic App Search

Made by [Elastic](#) 

Syncs documents from a Firestore collection to Elastic App Search to enable full-text search.

 Search

Install

[View details](#)



Delete User Data

Made by [Firebase](#) 

Deletes data keyed on a userId from Cloud Firestore, Realtime Database, or Cloud Storage when a user deletes the...

 Utilities

Install

[View details](#)



Stream Collections to BigQuery

Made by [Firebase](#) 

Sends realtime, incremental updates from a specified Cloud Firestore collection to BigQuery.

 Utilities

Install

[View details](#)



Distributed Counter

Made by [Firebase](#) 

Records event counters at scale to accommodate high-velocity writes to Cloud Firestore.

 Utilities

Install


[View details](#)



Trigger Email

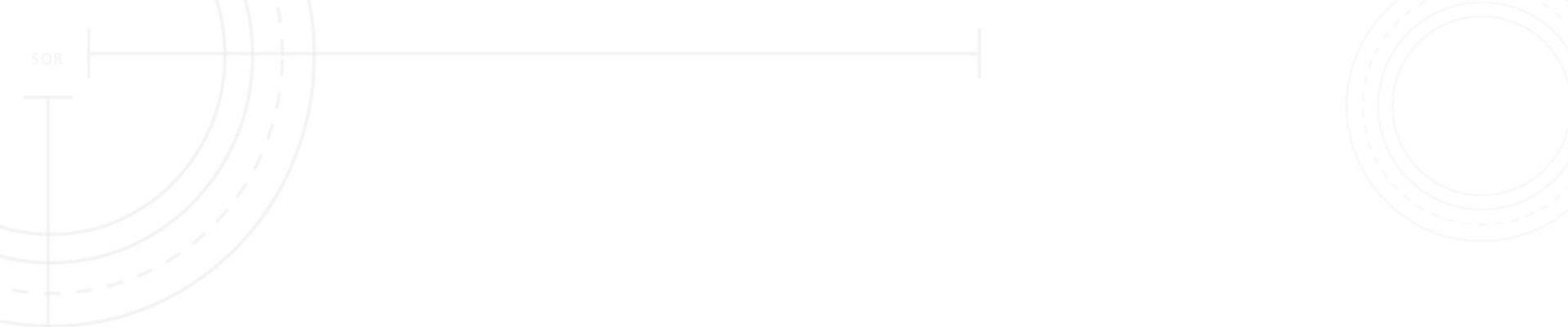
Made by [Firebase](#) 

Composes and sends an email based on the contents of a document written to a specified Cloud Firestore...

 Messaging

Install

[View details](#)



Release & Monitor

Crashlytics

Crashlytics

- Crash reporting from Flutter
- Crash reporting from native code (iOS/Android) and Android NDK
- Deobfuscation handling
- Metrics synced with Analytics

```
void main() async {  
  WidgetsFlutterBinding.ensureInitialized();  
  
  await Firebase.initializeApp();  
  
  // Pass all uncaught errors from the framework to Crashlytics.  
  FlutterError.onError = FirebaseCrashlytics.instance.recordFlutterError;  
  
  runApp(MyApp());  
}
```


Performance Monitoring

Performance Monitoring

- Startup times
- Rendering per screen
- HTTP requests
- Foreground / background activity
- Performance metrics



Test Lab

Test Lab

- Test on real devices and emulators
- Both Android and iOS
- Integrations with CLI / IDEs / CI systems
- *integration_test* package works with Test Lab



App Distribution

App Distribution

- Distribute artifacts
- Tester groups but only per-project
- No auto-provisioning
- No auto-update alerts
- Check out appcenter.ms

Engage



Analytics

Analytics

- Quantity analytics
- Free up to 500 events
- Events and user props
- Audience segmentation

```
await widget.analytics.logEvent(  
  name: 'test_event',  
  parameters: <String, dynamic>{  
    'string': 'string',  
    'int': 42,  
    'long': 12345678910,  
    'double': 42.0,  
    'bool': true,  
  },  
);
```

Remote Config

Remote Config

- Config server-side parameters for users
- Auto-synchronization every 24 hours
- Connected to Analytics audiences

```
Future<RemoteConfig> setupRemoteConfig() async {  
  await Firebase.initializeApp();  
  final RemoteConfig remoteConfig = RemoteConfig.instance;  
  await remoteConfig.setConfigSettings(RemoteConfigSettings(  
    fetchTimeout: const Duration(seconds: 10),  
    minimumFetchInterval: const Duration(hours: 1),  
  ));  
  await remoteConfig.setDefaults(<String, dynamic>{  
    'welcome': 'default welcome',  
    'hello': 'default hello',  
  });  
  RemoteConfigValue(null, ValueSource.valueStatic);  
  return remoteConfig;  
}
```

Predictions

Predictions

- ML connected to Analytics data predicts specific events
- Defaults to churn and spend predictions

A/B Testing

A/B Testing

- Experiments with Remote Config
- Connected to Predictions therefore also Analytics
- Combines with Notifications to notify about changes
- Allows “safe rollout” mechanisms

Cloud Messaging (FCM)

Cloud Messaging

- Send push notifications to Android / iOS / Web
- Use specific devices or pub/sub model
- Send acknowledgements from clients

```
FirebaseMessaging.onMessage.listen((RemoteMessage message) {  
  print('Got a message whilst in the foreground!');  
  print('Message data: ${message.data}');  
  
  if (message.notification != null) {  
    print('Message also contained a notification: ${message.notification}');  
  }  
});
```

In-App Messaging (β)

In-App Messaging

- Send engagement messages without instant arrival
- Use segmentation from Analytics / Predictions
- Implement custom alerts in your code

Google AdMob

Google AdMob

- Just ads :)
- Can be connected to Firebase Analytics
- Flutter support


```
InterstitialAd.load(  
  adUnitId: '<ad unit id>',  
  request: AdRequest(),  
  adLoadCallback: InterstitialAdLoadCallback(  
    onAdLoaded: (InterstitialAd ad) {  
      // Keep a reference to the ad so you can show it later.  
      this._interstitialAd = ad;  
    },  
    onAdFailedToLoad: (LoadAdError error) {  
      print('InterstitialAd failed to load: $error');  
    },  
  ));
```

Emulator

Firebase Emulator Suite

