Programmatic Initialization of Firebase Authentication in GCP

Background and Problem

Firebase Authentication usually requires an initial configuration step (often done by clicking **“Get Started”** in the Firebase console). Without this, API calls can fail with errors like `CONFIGURATION_NOT_FOUND`, meaning the project’s auth config is missing (Getting "CONFIGURATION_NOT_FOUND"). In a CI/CD setup, we need to automate this setup. The goal is to enable Firebase Authentication (specifically email/password sign-in) **fully programmatically**, using CLI or API calls – no manual console steps.

Solution Overview

To automate the “Get Started” initialization, you have two primary approaches:

- **Option A: Add Firebase to the GCP project** – Use the Firebase Management API (or CLI) to **add Firebase services** to your existing project. This creates the default Firebase Auth configuration (among other Firebase setup tasks) similar to the console’s “Get Started”.

- **Option B: Initialize Identity Platform for Auth** – Use the Identity Toolkit (Identity Platform) REST API to explicitly **initialize Firebase Auth** for the project. This creates the auth config without using the Firebase console (this essentially enables Google Cloud Identity Platform for the project) (Method: projects.identityPlatform.initializeAuth | Identity Platform Documentation | Google Cloud).

After the project is initialized by either method, you can **enable the Email/Password provider** via an API call. We’ll detail required API enablements, IAM permissions, and example commands for each step.

Implementation Steps

1. Enable Required APIs

Ensure the necessary service APIs are enabled on your GCP project (this can be done via gcloud or the Service Usage API):

- **Identity Toolkit API** (`identitytoolkit.googleapis.com`) – Provides Firebase Authentication/Identity Platform functionality (Get started using Firebase with an existing Google Cloud project | Firebase Documentation).

- **Firebase Management API** (`firebase.googleapis.com`) – Only needed for Option A (to programmatically add Firebase).

For example, using gcloud CLI:

gcloud services enable identitytoolkit.googleapis.com firebase.googleapis.com

This ensures the project can use Firebase Auth services. (If you plan to use other Firebase features like Firestore, Storage, etc., enable those APIs as well.)

2. Set Up IAM Permissions

Your CI/CD service account or user needs appropriate IAM permissions to configure Firebase Auth:

- To **add Firebase to a project** (Option A), you need the `firebase.projects.update` permission on that project (which is included in roles like **Project Editor** or **Owner**). You'll also need permissions to enable APIs (`serviceusage.services.enable`) and read the project (`resourcemanager.projects.get`). Being a Project **Editor/Owner** satisfies these requirements.

- To **initialize Auth via Identity Platform** (Option B), you need the Firebase Authentication admin permission to create configs: `firebaseauth.configs.create` (Method: projects.identityPlatform.initializeAuth | Identity Platform Documentation | Google Cloud). This is included in **Editor** as well as the **Firebase Authentication Admin** role (Firebase product-level predefined roles | Firebase Documentation) (Permissions Reference for Firebase Authentication). (Editor also covers enabling services, etc.)

- To **enable auth providers** (like email/password), you'll need `firebaseauth.configs.update` rights, which the above roles also include (Firebase product-level predefined roles | Firebase Documentation).

In summary, using a service account with **Editor** role on the project is the simplest way, or you can assign a combination of more specific roles (e.g. **Firebase Auth Admin** + **Service Usage Admin**). Make sure no organization policy blocks enabling services or creating service accounts (since adding Firebase will create service accounts (Get started using Firebase with an existing Google Cloud project | Firebase Documentation)).

3. Initialize Firebase Authentication Configuration

Next, perform the project initialization using **one** of the two options below. Both will create the underlying Authentication config for your project (resolving the `CONFIGURATION_NOT_FOUND` issue):

Option A: Add Firebase to the Project (Firebase Management API)

This approach fully enables Firebase on your existing GCP project, which includes setting up Firebase Auth. You can do this via the Firebase Management REST API or the Firebase CLI:

Using gcloud CLI: Run the command to add Firebase services. For example:

```
 gcloud firebase service-projects add <PROJECT_ID>
```

- This command wraps the REST call to `projects.addFirebase` (Set up and manage a Firebase project using the Management REST API | Firebase Documentation). It will initiate an operation to add Firebase; you should wait for the operation to complete (the CLI will handle this). On success, your project becomes a Firebase project.

Using Firebase Management API (REST): Invoke the REST endpoint directly. For example:

```
 ACCESS_TOKEN=$(gcloud auth print-access-token)
curl -X POST -H "Authorization: Bearer $ACCESS_TOKEN" \
    https://firebase.googleapis.com/v1beta1/projects/<PROJECT_ID>:addFirebase
```

- (The request body is empty (Set up and manage a Firebase project using the Management REST API | Firebase Documentation).) This returns an long-running **Operation**. You can poll the `operations/<name>` to check when `done:true` and a `FirebaseProject` is in the response (Set up and manage a Firebase project using the Management REST API | Firebase Documentation). Only after it's done can you use Firebase features.

When you "add Firebase", the necessary backend resources are created: for example, a default web API key (restricted to Firebase APIs), and service accounts for Firebase are set up (Get started using Firebase with an existing Google Cloud project | Firebase Documentation). Crucially, the **Identity Toolkit API is enabled** automatically as part of this (Get started using Firebase with an existing Google Cloud project | Firebase Documentation), and a default Auth config is created (even if no providers are enabled yet).

Option B: Initialize via Identity Platform API (Identity Toolkit)

This approach directly creates the Firebase Auth config using Identity Platform's initialization call, without enabling all Firebase services. Use the Identity Toolkit **v2 Admin API** endpoint `identityPlatform:initializeAuth`:

Using REST (curl):

```
 ACCESS_TOKEN=$(gcloud auth print-access-token)
curl -X POST -H "Authorization: Bearer $ACCESS_TOKEN" \
    -H "Content-Type: application/json" \
    -H "X-Goog-User-Project: <PROJECT_ID>" \

"https://identitytoolkit.googleapis.com/v2/projects/<PROJECT_ID>/identityPlatform:initializeAuth"
```

- This call requires no body (it's an empty POST) (Method: projects.identityPlatform.initializeAuth | Identity Platform Documentation | Google Cloud). A successful response is empty with HTTP 200. It initializes Identity Platform for your project (Method: projects.identityPlatform.initializeAuth | Identity Platform Documentation | Google Cloud). (The `X-Goog-User-Project` header is used when using a service account to specify the target project for billing/quota.) For example, a similar approach was shown on StackOverflow (firebase - Identity Toolkit API config returns "CONFIGURATION_NOT_FOUND" - Stack Overflow).

After this call, your project is enabled for Identity Platform (which is basically Firebase Auth under the hood). Note: This API is only available for **billing-enabled projects** (Method: projects.identityPlatform.initializeAuth | Identity Platform Documentation | Google Cloud). In other words, you need to have a Blaze (pay-as-you-go) plan or otherwise attached billing account for the project. Using this route effectively "upgrades" your project's auth to Identity Platform's model (which has some pricing differences, as noted by the community (firebase - Identity Toolkit API config returns "CONFIGURATION_NOT_FOUND" - Stack Overflow)). In practice, for standard email/password auth usage the costs on Blaze may remain in the free tier for low volume, but be aware of this requirement.

Result: Either Option A or B will create the authentication configuration on the project. At this point, the Firebase Authentication service recognizes your project (no more `CONFIGURATION_NOT_FOUND` errors). However, by default **no sign-in providers are enabled yet** – so you still need to enable the Email/Password provider via an update.

4. Enable the Email/Password Sign-In Provider

With the auth config in place, you can programmatically enable the email/password authentication method. This is done by updating the Auth **Config** resource via the Identity Toolkit **Admin API**:

Use the `projects.updateConfig` method to set the email provider to enabled. For example, a HTTP PATCH call:

```
ACCESS_TOKEN=$(gcloud auth print-access-token)
curl -X PATCH -H "Authorization: Bearer $ACCESS_TOKEN" \
    -H "Content-Type: application/json" \
"https://identitytoolkit.googleapis.com/admin/v2/projects/<PROJECT_ID>/config?updateMas
k=signIn.email" \
   -d '{
        "signIn": {
          "email": {
            "enabled": true,
            "passwordRequired": true
          }
        }
      }'
```

In this JSON, we set `"email.enabled": true` and `"email.passwordRequired": true` to require a password (i.e. standard email+password login) (Config | Identity Platform Documentation | Google Cloud). This corresponds to enabling Email/Password in the Firebase console. If you wanted passwordless email link sign-in, you could set `enabled:true` and `passwordRequired:false` (Config | Identity Platform Documentation | Google Cloud), but typically for email/password authentication you require a password. The `updateMask=signIn.email` parameter ensures only the email provider config is changed (Method: projects.updateConfig | Identity Platform Documentation | Google Cloud).

This call will succeed if your identity platform config exists (after steps above) and your credentials have permission. It returns the updated Config object on success (Method: projects.updateConfig | Identity Platform Documentation | Google Cloud). Required permission for this is `firebaseauth.configs.update` (included in Editor or Firebase Auth Admin) (Firebase product-level predefined roles | Firebase Documentation).

> **Note:** The Identity Toolkit Admin API (v2) is used here. In the past, there was no public API to enable providers, but as of v2 it's possible (How to enable email and password signin provider for new firebase project using gcloud console or firebase tools CLI? - Stack Overflow). We use the `/admin/v2/projects/.../config` endpoint (the same one the Firebase console uses under the hood) as shown above.

At this point, Firebase Authentication with Email/Password is fully enabled in your project. You can verify by calling the GET config endpoint or by attempting to create a user with email/password via the API/Admin SDK.

Summary of Required Permissions, APIs, and Steps

Below is a summary of the key steps with the associated IAM permissions and API services involved:

Step	Description	Key Permissions / Roles Required	APIs/Services Involved
1. Enable APIs	Enable Identity Toolkit API (Firebase Auth) (and Firebase Management API if using Option A).	`serviceusage.services.enable` on the project (included in **Project Editor/Owner**).	**Service Usage API** (to enable:`identitytoolkit.googleapis.com`, `firebase.googleapis.com`) ([Get started using Firebase with an existing Google Cloud project
2. IAM Setup	Grant permissions to run auth setup.	**Project Editor/Owner** role is simplest (includes all needed perms). Or:`firebase.projects.update` (to add Firebase)`firebaseauth.configs.create` (to init Auth) ([Method: projects.identityPlatform.initializeAuth	Identity Platform Documentation
3A. Initialize (Option A)	*Add Firebase to project.* Programmatically link Firebase to the GCP project, which creates default auth config and enables APIs.	`firebase.projects.update` on the project (in **Editor/Owner** role). Also requires `serviceusage.services.enable` (Editor covers this).	**Firebase Management API** – call `projects.addFirebase` ([Set up and manage a Firebase project using the Management REST API
3B. Initialize (Option B)	*Initialize Identity Platform.* Directly create the auth config via API	`firebaseauth.configs.create` permission (included in **Firebase Auth Admin** or **Editor**) ([Method: projects.identityPlatform.initializeAuth	Identity Platform Documentation

	without full Firebase add.		
4. Enable Email/ Pass	Update auth config to turn on Email/Password provider.	`firebaseauth.configs.update` permission (included in **Firebase Auth Admin** or **Editor**) ([Firebase product-level predefined roles	Firebase Documentation](https://firebase.google.com/docs/projects/iam/roles-predefined-product#:~:text=firebaseauth.get)).

Notes: In the table above, **Editor** refers to the GCP project role, which encompasses all required permissions for simplicity (Permissions Reference for Firebase Authentication). For principle of least privilege, you could use a combination of narrower roles (such as **Firebase Authentication Admin** for auth configs (Firebase product-level predefined roles | Firebase Documentation), plus roles to enable services). Also ensure the **Firebase Management API** is enabled if using Option A, and a billing account is attached if using Option B (Method: projects.identityPlatform.initializeAuth | Identity Platform Documentation | Google Cloud).

Additional Considerations and Limitations

- **Billing Requirement (Option B):** The direct Identity Platform initialization (Option B) only works on projects with billing enabled (Method: projects.identityPlatform.initializeAuth | Identity Platform Documentation | Google Cloud). If your project is on a free Spark plan with no billing, you'll need to use Option A (adding Firebase) instead, as it doesn't require upfront billing (though certain Firebase services may prompt for upgrade if used beyond free limits).

- **Firebase Terms of Service:** Adding Firebase to a project the first time may require accepting Firebase's terms. This **cannot be done via API or CLI** (Get started using Firebase with an existing Google Cloud project | Firebase Documentation). Ensure that the Google account or organization has already accepted Firebase terms (for example, by manually adding Firebase to any project once, or via an organization policy agreement) before automating this in CI/CD. Otherwise, the `projects.addFirebase` call might fail.

- **Service Account Creation Policy:** When you add Firebase (Option A), it will create service accounts in the project (like the Firebase Admin SDK service account) (Get started using Firebase with an existing Google Cloud project | Firebase Documentation). If your organization has a policy blocking service account creation, you may need to exempt this project or temporarily allow it. The IAM permission `iam.serviceAccounts.create` is used internally during addFirebase. Make sure your CI service principal isn't restricted from creating these.

- **API Key for Client SDKs:** Firebase Auth (client SDK usage) requires an API key. Option A will automatically generate a **Browser API Key** in your project and restrict it to Firebase APIs (Get started using Firebase with an existing Google Cloud project | Firebase Documentation). Option B (initializeAuth) does **not** create an API key, so if you plan to use Firebase client SDKs, you must create an API key in Google Cloud console or via gcloud (`gcloud alpha services api-keys create`) and enable the "Identity Toolkit API" on it. This API key is needed for client-side calls (like web or mobile app sign-in); server-side Admin SDK usage doesn't require an API key.

- **Verification:** After running these steps in CI/CD, you can verify setup by calling `GET https://identitytoolkit.googleapis.com/admin/v2/projects/<PROJECT_ID>/config` with your token to fetch the Auth config. It should show `"email": {"enabled": true, ...}` in `signIn` settings. Alternatively, try creating a user via Admin SDK or REST; if it succeeds or if you can see the user in the Firebase Console, everything is configured.

- **No Terraform Required:** The above methods use gcloud and REST calls. (Terraform has a resource `google_firebase_project` that wraps addFirebase, but since you wish to avoid Terraform, the direct approaches above are recommended (Firebase Auth + Api Gateway + Cloud Function | by Frank DS | Medium).) Both approaches are suitable for script automation in CI/CD pipelines.

By following the steps above, you can fully automate Firebase Authentication setup – from enabling the service to configuring email/password sign-in – without any manual console interaction. This ensures that when your CI/CD creates a new environment or project, the Firebase Auth backend is initialized and ready for use.

References: The solution draws on official Google documentation and proven community solutions. For instance, Google's docs on using the Firebase Management API to add Firebase to a project (Set up and manage a Firebase project using the Management REST API | Firebase Documentation) and the Identity Platform admin API reference for initializing auth (Method: projects.identityPlatform.initializeAuth | Identity Platform Documentation | Google Cloud) (Method: projects.identityPlatform.initializeAuth | Identity Platform Documentation | Google Cloud) were used. A Stack Overflow answer demonstrated the `identityPlatform:initializeAuth` approach in practice (firebase - Identity Toolkit API config returns "CONFIGURATION_NOT_FOUND" - Stack Overflow), and the ability to enable the Email/Password provider via the REST API was confirmed in 2023 (How to enable email and password signin provider for new firebase project using gcloud console or firebase tools CLI? - Stack Overflow). These sources and the Firebase IAM documentation (Firebase product-level predefined roles | Firebase Documentation) (Firebase product-level predefined roles | Firebase Documentation) guided the required permissions and steps outlined. By leveraging these methods, you can reliably script the "Get Started with Firebase Auth" initialization as part of your deployment pipeline.