Firebase + GitHub Actions Deployment Setup Guide

This guide provides step-by-step instructions for setting up a complete CI/CD pipeline for deploying Angular applications to Firebase across multiple environments (Development, Pre-Production, Production).

Table of Contents

Architecture Overview

```
┌─────────────────────────────────────────────────────────────────────┐
│                         GitHub Repository                            │
├─────────────────────────────────────────────────────────────────────┤
│  developer branch ──push──► GitHub Actions ──► Firebase (DEV)       │
│  preprod branch   ──push──► GitHub Actions ──► Firebase (PREPROD)   │
│  main branch      ──push──► GitHub Actions ──► Firebase (PROD)      │
│  Pull Request     ──────────► Preview Channel (DEV project)         │
└─────────────────────────────────────────────────────────────────────┘
```

Environment Mapping

Branch	Firebase Project	URL Pattern
`developer`	`{app}-dev`	`https://{app}-dev.web.app`
`preprod`	`{app}-preprod`	`https://{app}-preprod.web.app`
`main`	`{app}-prod`	`https://{app}-prod.web.app`

Prerequisites

Before starting, ensure you have:

- [ ] Google account with access to Firebase Console
- [ ] GitHub account with admin access to the repository
- [ ] Node.js 18+ installed locally
- [ ] Firebase CLI installed (`npm install -g firebase-tools`)

- [ ] Git installed locally

Firebase Project Setup

Step 1: Create Firebase Projects

1. Go to Firebase Console
2. Click **"Create a project"** (or **"Add project"**)
3. Create **three separate projects**:

Project Name	Project ID	Purpose
`{AppName} Dev`	`{appname}-dev`	Development/Testing
`{AppName} PreProd`	`{appname}-preprod`	Pre-Production/Staging
`{AppName} Prod`	`{appname}-prod`	Production

> ***Note***: *Project IDs must be globally unique and cannot be changed after creation.*

Step 2: Enable Billing (Blaze Plan)

Cloud Functions require the **Blaze (Pay as you go)** plan.

For each project:

1. Go to **Project Settings** → **Usage and billing**
2. Click **"Modify plan"**
3. Select **"Blaze"** plan
4. Add a billing account (or create one)

> ***Cost Note***: *The Blaze plan includes a generous free tier. Small projects typically incur minimal or no charges.*

Step 3: Enable Cloud Billing API (Required)

> ⚠️ ***Manual Step Required***: *The Cloud Billing API must be enabled manually for each project before the first deployment.*

For **each** Firebase project (dev, preprod, prod):

1. Go to Cloud Billing API
2. Select the Firebase project from the dropdown (top of page)
3. Click **"Enable"**
4. Repeat for all three projects

> ***Why?*** *Firebase CLI needs this API to verify your project's billing status before deploying Cloud Functions.*

Step 4: Other APIs (Automatic - No Action Required)

> ✅ ***Good news!*** *Most other APIs are* ***enabled automatically*** *when you first deploy via GitHub Actions.*

When the deployment runs, Firebase CLI automatically enables:

- Cloud Functions API
- Cloud Build API
- Artifact Registry API

- Firebase Hosting API

You'll see messages like this in the deployment logs:

```
⚠  functions: missing required API cloudfunctions.googleapis.com. Enabling now...
⚠  functions: missing required API cloudbuild.googleapis.com. Enabling now...
⚠  artifactregistry: missing required API artifactregistry.googleapis.com. Enabling now...
```

No manual action required for these - proceed to Service Account Creation.

> ***Troubleshooting***: *If APIs fail to enable automatically, see Troubleshooting section for manual steps.*

Service Account Creation

Service accounts allow GitHub Actions to deploy to Firebase without user interaction.

Step 1: Create Service Account

For **each** Firebase project:

1. Go to Google Cloud Console
2. Select the Firebase project
3. Navigate to **IAM & Admin** → **Service Accounts**
4. Click **"+ CREATE SERVICE ACCOUNT"**
5. Fill in:
   - **Name**: `github-actions-deploy`
   - **ID**: `github-actions-deploy`
   - **Description**: `Service account for GitHub Actions CI/CD`
6. Click **"CREATE AND CONTINUE"**

Step 2: Assign Roles

Add the following roles to the service account:

Role	Purpose
`Firebase Admin`	Full Firebase access
`Cloud Functions Admin`	Deploy Cloud Functions
`Service Account User`	Run as service account
`Cloud Build Editor`	Build container images
`Artifact Registry Admin`	Store container images

To add roles:

1. Click **"+ ADD ANOTHER ROLE"** for each role
2. Search and select the role
3. Click **"CONTINUE"** then **"DONE"**

Step 3: Generate JSON Key

1. Find the service account in the list

2. Click the **three dots** (⋮) → **"Manage keys"**
3. Click **"ADD KEY"** → **"Create new key"**
4. Select **"JSON"** format
5. Click **"CREATE"**
6. **Save the downloaded JSON file securely**

> ⚠️ ***Security Warning**: This JSON key provides full access to your Firebase project. Never commit it to version control or share it publicly.*

Step 4: Repeat for All Projects

Repeat Steps 1-3 for each environment:

- `{appname}-dev` → Save as `firebase-sa-dev.json`
- `{appname}-preprod` → Save as `firebase-sa-preprod.json`
- `{appname}-prod` → Save as `firebase-sa-prod.json`

GitHub Repository Setup

Step 1: Create Repository

1. Go to GitHub and create a new repository
2. Initialize with a README or push existing code:

```
git init
git add .
git commit -m "Initial commit"
git branch -M main
git remote add origin https://github.com/{org}/{repo}.git
git push -u origin main
```

Step 2: Create Branches

Create the required branches:

```
# Create developer branch
git checkout -b developer
git push -u origin developer

# Create preprod branch
git checkout -b preprod
git push -u origin preprod

# Return to main
git checkout main
```

Step 3: Branch Protection Rules (Recommended)

1. Go to repository **Settings** → **Branches**
2. Click **"Add branch protection rule"**
3. Configure for each branch:

For `main` (Production):

- Branch name pattern: `main`
- ✅ Require a pull request before merging
- ✅ Require approvals (minimum: 2)
- ✅ Require status checks to pass
- ✅ Require branches to be up to date

For `preprod`:

- Branch name pattern: `preprod`
- ✅ Require a pull request before merging
- ✅ Require approvals (minimum: 1)
- ✅ Require status checks to pass

For `developer`:

- Branch name pattern: `developer`
- ✅ Require status checks to pass (optional)

GitHub Secrets Configuration

Step 1: Add Repository Secrets

1. Go to repository **Settings** → **Secrets and variables** → **Actions**
2. Click **"New repository secret"**
3. Add the following secrets:

Secret Name	Value
`FIREBASE_SA_DEV`	Contents of `firebase-sa-dev.json`
`FIREBASE_SA_PREPROD`	Contents of `firebase-sa-preprod.json`
`FIREBASE_SA_PROD`	Contents of `firebase-sa-prod.json`

Step 2: How to Add JSON Content

1. Open the JSON key file in a text editor
2. **Copy the entire contents** (including curly braces)
3. Paste as the secret value in GitHub

Example JSON structure:

```json
{
  "type": "service_account",
  "project_id": "your-project-id",
  "private_key_id": "...",
  "private_key": "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----\n",
  "client_email": "github-actions-deploy@your-project-id.iam.gserviceaccount.com",
  ...
}
```

GitHub Environments Setup

Environments provide deployment protection rules and environment-specific secrets.

Step 1: Create Environments

1. Go to repository **Settings** → **Environments**
2. Click **"New environment"**
3. Create three environments:

Environment Name	Purpose
`development`	Dev deployments
`preprod`	Pre-production deployments
`production`	Production deployments

Step 2: Configure Protection Rules (Optional)

For **production** environment:

1. Click on `production` environment
2. Enable **"Required reviewers"**
3. Add team members who can approve production deployments
4. Enable **"Wait timer"** (e.g., 5 minutes) for extra safety

For **preprod** environment:

1. Enable **"Required reviewers"** (optional)
2. Add fewer reviewers than production

Deployment Workflow

Create the GitHub Actions workflow file:

File: `.github/workflows/firebase-deploy.yml`

```yaml
name: Deploy to Firebase

on:
  push:
    branches:
      - developer
      - preprod
      - main
  pull_request:
    branches:
      - developer
      - preprod
      - main

env:
  # ============================================
  # UPDATE THESE WITH YOUR FIREBASE PROJECT IDs
  # ============================================
  FIREBASE_PROJECT_DEV: your-app-dev
  FIREBASE_PROJECT_PREPROD: your-app-preprod
  FIREBASE_PROJECT_PROD: your-app-prod

jobs:
  # ============================================
  # BUILD JOB
  # ============================================
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: '20'
          cache: 'npm'

      - name: Install frontend dependencies
        run: npm ci

      - name: Build frontend
        run: npm run build

      - name: Install functions dependencies
        run: cd functions && npm ci

      - name: Build functions
        run: cd functions && npm run build

      - name: Upload build artifacts
        uses: actions/upload-artifact@v4
        with:
          name: build-output
          path: |
            dist/
            functions/lib/
          retention-days: 1

  # ============================================
  # DEVELOPMENT DEPLOYMENT
  # ============================================
  deploy-dev:
    needs: build
    if: github.event_name == 'push' && github.ref == 'refs/heads/developer'
    runs-on: ubuntu-latest
    environment: development
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: '20'

      - name: Download build artifacts
        uses: actions/download-artifact@v4
        with:
          name: build-output

      - name: Install functions dependencies
        run: cd functions && npm ci

      - name: Deploy to DEV
        uses: w9jds/firebase-action@master
        with:
          args: deploy --only hosting,functions --force --project ${{
env.FIREBASE_PROJECT_DEV }}
        env:
          GCP_SA_KEY: ${{ secrets.FIREBASE_SA_DEV }}

  # ============================================
  # PRE-PRODUCTION DEPLOYMENT
  # ============================================
  deploy-preprod:
    needs: build
    if: github.event_name == 'push' && github.ref == 'refs/heads/preprod'
    runs-on: ubuntu-latest
    environment: preprod
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: '20'

      - name: Download build artifacts
        uses: actions/download-artifact@v4
        with:
          name: build-output

      - name: Install functions dependencies
        run: cd functions && npm ci

      - name: Deploy to PREPROD
        uses: w9jds/firebase-action@master
        with:
          args: deploy --only hosting,functions --force --project ${{
env.FIREBASE_PROJECT_PREPROD }}
        env:
          GCP_SA_KEY: ${{ secrets.FIREBASE_SA_PREPROD }}

  # ============================================
  # PRODUCTION DEPLOYMENT
  # ============================================
  deploy-prod:
    needs: build
    if: github.event_name == 'push' && github.ref == 'refs/heads/main'
    runs-on: ubuntu-latest
    environment: production
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: '20'

      - name: Download build artifacts
        uses: actions/download-artifact@v4
        with:
          name: build-output

      - name: Install functions dependencies
        run: cd functions && npm ci

      - name: Deploy to PRODUCTION
        uses: w9jds/firebase-action@master
        with:
          args: deploy --only hosting,functions --force --project ${{
env.FIREBASE_PROJECT_PROD }}
        env:
          GCP_SA_KEY: ${{ secrets.FIREBASE_SA_PROD }}

  # ============================================
  # PR PREVIEW (Deploys to DEV project)
  # ============================================
  deploy-preview:
    needs: build
    if: github.event_name == 'pull_request'
    runs-on: ubuntu-latest
    permissions:
      checks: write
      contents: read
      pull-requests: write
    steps:
      - name: Checkout repository
```

```
        uses: actions/checkout@v4

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: '20'

      - name: Download build artifacts
        uses: actions/download-artifact@v4
        with:
          name: build-output

      - name: Install functions dependencies
        run: cd functions && npm ci

      - name: Deploy PR Preview
        uses: FirebaseExtended/action-hosting-deploy@v0
        with:
          repoToken: '${{ secrets.GITHUB_TOKEN }}'
          firebaseServiceAccount: '${{ secrets.FIREBASE_SA_DEV }}'
          projectId: ${{ env.FIREBASE_PROJECT_DEV }}
          expires: 7d
          channelId: pr-${{ github.event.pull_request.number }}
```

Branch Strategy

Development Flow

```
feature/* ──PR──► developer ──PR──► preprod ──PR──► main
                      │                │             │
                      ▼                ▼             ▼
                   DEV env          PREPROD        PROD
```

Workflow

1. **Feature Development**

```
git checkout developer
git pull origin developer
git checkout -b feature/my-feature
# ... make changes ...
git add .
git commit -m "Add feature"
git push origin feature/my-feature
# Create PR to developer branch
```

2. **Promote to Pre-Production**

```
# Create PR from developer → preprod
# Review and merge
```

3. **Promote to Production**

```
# Create PR from preprod → main
# Review, approve, and merge
```

Verification & Testing

Step 1: Verify Firebase Configuration

```
# Login to Firebase CLI
firebase login

# List projects
firebase projects:list

# Test deployment locally
firebase deploy --only hosting --project your-app-dev --dry-run
```

Step 2: Test GitHub Actions

1. Make a small change to `developer` branch
2. Push and monitor **Actions** tab in GitHub
3. Verify deployment at `https://{app}-dev.web.app`

Step 3: Verify Each Environment

Environment	URL	Status Check
Development	`https://{app}-dev.web.app`	✅
Pre-Production	`https://{app}-preprod.web.app`	✅
Production	`https://{app}-prod.web.app`	✅

Troubleshooting

Common Issues

1. "Permission denied" or "403 Forbidden"

Cause: Service account missing required roles.

Solution:

- Verify all roles are assigned (see Service Account Creation)
- Regenerate the JSON key and update GitHub secret

2. "Cloud Billing API has not been used" or "API not enabled"

Cause: Required Google Cloud APIs are not enabled.

Solution - Manually enable APIs:

1. Go to Google Cloud Console
2. Select your Firebase project from the dropdown
3. Navigate to **APIs & Services** → **Library**
4. Search for and enable each API:

API	Direct Link	Required
Cloud Billing API	Enable	⚠️ **Must enable manually**
Cloud Functions API	Enable	Auto-enabled
Cloud Build API	Enable	Auto-enabled
Artifact Registry API	Enable	Auto-enabled
Firebase Hosting API	Enable	Auto-enabled

5. Click **"Enable"** for each API (especially Cloud Billing API)
6. **Wait 2-3 minutes** for changes to propagate
7. Re-run the GitHub Actions deployment

> ***Note***: *Enable the Cloud Billing API for **all three projects** (dev, preprod, prod) before first deployment.*

3. "Failed to find location of Firebase Functions SDK"

Cause: `node_modules` not installed in functions directory during deployment.

Solution: Ensure workflow includes:

```
- name: Install functions dependencies
  run: cd functions && npm ci
```

4. "Billing account not configured"

Cause: Project is not on Blaze plan.

Solution: Upgrade to Blaze plan in Firebase Console.

5. Environment deployment stuck/waiting

Cause: GitHub Environment requires approval.

Solution:

- Go to **Actions** tab → Click on the waiting job
- Approve the deployment
- Or remove `environment:` line from workflow

6. "Resource not found" during deployment

Cause: Project ID mismatch.

Solution: Verify `FIREBASE_PROJECT_*` values in workflow match actual Firebase project IDs.

Debug Commands

```
# Verify Firebase CLI authentication
firebase login:list

# Test service account locally
export GOOGLE_APPLICATION_CREDENTIALS="path/to/service-account.json"
firebase deploy --only hosting --project your-app-dev

# Check GitHub Actions logs
# Go to: Repository → Actions → Select workflow run → View logs
```

Security Best Practices

1. **Never commit service account keys** to version control
2. **Rotate service account keys** periodically (every 90 days recommended)
3. **Use environment protection rules** for production deployments
4. **Enable branch protection** to prevent direct pushes to main
5. **Review PR changes** before merging to production
6. **Audit GitHub Actions logs** regularly
7. **Use least privilege** - only assign necessary roles to service accounts

---

Quick Reference

URLs

Service	URL
Firebase Console	https://console.firebase.google.com/
Google Cloud Console	https://console.cloud.google.com/
GitHub Actions	https://github.com/{org}/{repo}/actions
Cloud Billing API	https://console.cloud.google.com/apis/library/cloudbilling.googleapis.com

Required Manual Setup (Per Project)

Step	Required For
Create Firebase project	All environments
Enable Blaze billing	Cloud Functions
Enable Cloud Billing API	Deployment verification
Create service account	GitHub Actions auth
Add JSON key to GitHub secrets	CI/CD pipeline

Secret Names

Secret	Description

`FIREBASE_SA_DEV`	Dev service account JSON
`FIREBASE_SA_PREPROD`	PreProd service account JSON
`FIREBASE_SA_PROD`	Prod service account JSON

Branch → Environment Mapping

Branch	Environment	Firebase Project
`developer`	development	`{app}-dev`
`preprod`	preprod	`{app}-preprod`
`main`	production	`{app}-prod`

Support

For issues related to:

- **Firebase**: Firebase Support
- **GitHub Actions**: GitHub Actions Documentation
- **Google Cloud IAM**: IAM Documentation

Last updated: December 2024