“Best Free Option #1: GitHub Actions ⭐ (RECOMMENDED)” **give me step by step guide**

Perfect! Now I have all the information needed. Let me create a comprehensive step-by-step guide for you.

Complete Step-by-Step Guide: Build Android APK with GitHub Actions

PHASE 1: Connect Replit to GitHub

Step 1: Open Your Replit Project

1. Go to Replit.com and open your Android project
2. Look for the **Version Control** or **Git** icon on the left sidebar (looks like a branch symbol)

Step 2: Connect to GitHub

1. Click on the **"Connect to GitHub"** button
2. A new tab will open asking you to authorize Replit
3. Select your GitHub account and authorize
4. Choose whether to:
    - **Create a new repository** (recommended if you don't have one)
    - **Connect to existing repository**

Step 3: Initial Push to GitHub[1] [2]

1. In the Git panel, you'll see all your project files
2. Add a commit message like: `"Initial Android app commit"`
3. Click **"Stage and commit all changes"**
4. Click the **"Push"** button to upload to GitHub

✅ **Checkpoint:** Visit your GitHub repository URL to confirm all files are there.

PHASE 2: Create Signing Keystore (For Release APK)

Step 4: Generate Keystore File[3] [4]

Open **Replit Shell** and run this command:

```
keytool -genkey -v -keystore my-release-key.jks -alias my-key-alias -keyalg RSA -keysize
```

Fill in the prompts:

- **Keystore password:** (Choose a strong password - save it!)
- **Key password:** (Can be same as keystore password)
- **First and last name:** Your name or company name
- **Organizational unit:** (e.g., Development)
- **Organization:** Your company/app name
- **City, State, Country:** Your location details

⚠ **IMPORTANT:** Save these passwords somewhere secure! You'll need them.

Step 5: Encode Keystore to Base64[5] [3]

In Replit Shell, run:

```
openssl base64 -in my-release-key.jks -out keystore_base64.txt
```

This creates a text file with encoded keystore.

Step 6: Copy the Encoded Keystore

```
cat keystore_base64.txt
```

Select and copy ALL the text output (it will be very long).

PHASE 3: Add Secrets to GitHub

Step 7: Go to GitHub Repository Settings[4] [3]

1. Open your repository on GitHub
2. Click **Settings** (top right of repository)
3. In the left sidebar, click **Secrets and variables** → **Actions**
4. Click **"New repository secret"**

Step 8: Add Four Secrets[5]

Create these secrets one by one:

Secret Name	Value	Where to Get It
`KEYSTORE`	Paste the entire base64 text from Step 6	From `keystore_base64.txt`
`KEYSTORE_PASSWORD`	Your keystore password	From Step 4
`KEY_ALIAS`	`my-key-alias`	From Step 4 command
`KEY_PASSWORD`	Your key password	From Step 4

How to add each secret:

1. Click **"New repository secret"**
2. Enter **Name** (e.g., `KEYSTORE`)
3. Paste **Value**
4. Click **"Add secret"**
5. Repeat for all 4 secrets

□ PHASE 4: Configure Gradle for Signing

Step 9: Update `app/build.gradle.kts`[3]

Add this code inside the `android` block (before `buildTypes`):

```
android {
    // ... existing code ...

    signingConfigs {
        create("release") {
            // Read from environment variables (GitHub secrets)
            storeFile = file(System.getenv("KEYSTORE_FILE") ?: "my-release-key.jks")
            storePassword = System.getenv("KEYSTORE_PASSWORD")
            keyAlias = System.getenv("KEY_ALIAS")
            keyPassword = System.getenv("KEY_PASSWORD")
        }
    }

    buildTypes {
        getByName("release") {
            isMinifyEnabled = false
            signingConfig = signingConfigs.getByName("release")
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    // ... rest of code ...
}
```

Step 10: Commit and Push Changes

Back in Replit:

1. Go to Git panel
2. Add commit message: `"Configure signing for release builds"`
3. Stage and commit
4. Push to GitHub

▯ PHASE 5: Create GitHub Actions Workflow

Step 11: Create Workflow File in Replit[6] [7]

Create this folder structure in your project:

```
.github/
  └── workflows/
      └── build-apk.yml
```

In Replit:

1. Click **"Add folder"** → name it `.github`
2. Inside `.github`, create folder `workflows`
3. Inside `workflows`, create file `build-apk.yml`

Step 12: Add Workflow Code

Paste this complete workflow into `build-apk.yml`:

```
name: Build Android APK

on:
  push:
    branches: [ main, master ]
  pull_request:
    branches: [ main, master ]
  workflow_dispatch:  # Allows manual trigger

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
```

```
    # Step 1: Checkout code
    - name: Checkout repository
      uses: actions/checkout@v4

    # Step 2: Set up JDK
    - name: Set up JDK 17
      uses: actions/setup-java@v4
      with:
        distribution: 'temurin'
        java-version: '17'
        cache: 'gradle'

    # Step 3: Grant execute permission for gradlew
    - name: Grant execute permission for gradlew
      run: chmod +x gradlew

    # Step 4: Decode keystore from base64
    - name: Decode Keystore
      run: |
        echo "${{ secrets.KEYSTORE }}" | base64 --decode > app/my-release-key.jks

    # Step 5: Build Debug APK (for testing)
    - name: Build Debug APK
      run: ./gradlew assembleDebug

    # Step 6: Build Release APK (signed)
    - name: Build Release APK
      env:
        KEYSTORE_FILE: app/my-release-key.jks
        KEYSTORE_PASSWORD: ${{ secrets.KEYSTORE_PASSWORD }}
        KEY_ALIAS: ${{ secrets.KEY_ALIAS }}
        KEY_PASSWORD: ${{ secrets.KEY_PASSWORD }}
      run: ./gradlew assembleRelease

    # Step 7: Upload Debug APK
    - name: Upload Debug APK
      uses: actions/upload-artifact@v4
      with:
        name: app-debug
        path: app/build/outputs/apk/debug/app-debug.apk

    # Step 8: Upload Release APK
    - name: Upload Release APK
      uses: actions/upload-artifact@v4
      with:
        name: app-release
        path: app/build/outputs/apk/release/app-release.apk

    # Step 9: Create Release (Optional - on version tags)
    - name: Create GitHub Release
      if: startsWith(github.ref, 'refs/tags/v')
      uses: softprops/action-gh-release@v1
      with:
        files: |
          app/build/outputs/apk/release/app-release.apk
          app/build/outputs/apk/debug/app-debug.apk
      env:
        GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

Step 13: Commit and Push Workflow

In Replit Git panel:

1. Commit message: `"Add GitHub Actions workflow for APK build"`
2. Stage and commit
3. Push to GitHub

▯ PHASE 6: Trigger and Download APK

Step 14: Trigger the Build[7]

The workflow triggers automatically when you push to `main` or `master` branch.

To manually trigger:

1. Go to your GitHub repository
2. Click **Actions** tab (top menu)
3. Click **"Build Android APK"** workflow (left sidebar)
4. Click **"Run workflow"** button (right side)
5. Select branch and click **"Run workflow"**

Step 15: Monitor Build Progress[6]

1. In **Actions** tab, you'll see your workflow running
2. Click on the workflow run to see details
3. Watch each step complete (takes 5-10 minutes)
4. Green checkmarks = success ✅

Step 16: Download Your APK[8] [7]

Once build completes:

1. Scroll down to **"Artifacts"** section
2. You'll see:
    - **app-debug** - for testing (unsigned)
    - **app-release** - for Play Store (signed)
3. Click to download the ZIP file
4. Extract and install on your Android device

BONUS: Auto-Release on Version Tags

Step 17: Create Version Releases

To automatically create GitHub releases with APKs:

```
# In Replit Shell
git tag -a v1.0.0 -m "First release"
git push origin v1.0.0
```

This will:

- Build APK automatically
- Create a GitHub Release
- Attach APK files to the release
- Generate release notes

Quick Reference Checklist

- [ ] Connected Replit to GitHub
- [ ] Pushed initial code to GitHub
- [ ] Generated keystore file
- [ ] Encoded keystore to base64
- [ ] Added 4 secrets to GitHub (KEYSTORE, KEYSTORE_PASSWORD, KEY_ALIAS, KEY_PASSWORD)
- [ ] Updated `app/build.gradle.kts` with signing config
- [ ] Created `.github/workflows/build-apk.yml` file
- [ ] Pushed workflow to GitHub
- [ ] Triggered build in Actions tab
- [ ] Downloaded APK from Artifacts

Troubleshooting Common Issues

Build fails with "Permission denied":

```
# In workflow, ensure this step runs:
chmod +x gradlew
```

Keystore not found:

- Check that Step 4 in workflow correctly decodes to `app/my-release-key.jks`
- Verify path matches your `build.gradle.kts` configuration

Gradle version issues:

- Update to Java 17 in workflow (already included above)
- Check your `gradle/wrapper/gradle-wrapper.properties`

Secrets not working:

- Secret names must match EXACTLY (case-sensitive)
- Re-create secrets if unsure

▯ Next Steps: Publishing to Google Play Store

Once you have your signed `app-release.apk`:

1. Go to Google Play Console
2. Create a new app
3. Upload your APK to Internal Testing track
4. Complete store listing
5. Submit for review

Would you like help with any specific step, or shall I guide you through publishing to Google Play Store?

⁂

1. https://weblianz.com/blog/how-to-connect-replit-to-github
2. https://www.youtube.com/watch?v=tlR1l952HDg
3. https://proandroiddev.com/how-to-securely-build-and-sign-your-android-app-with-github-actions-ad5323452ce
4. https://www.kodeco.com/33542617-deploying-android-apps-using-github-actions/lessons/4
5. https://github.com/marketplace/actions/github-action-release-apk
6. https://www.p4innovation.com/blog/2024/4/25/building-android-using-github-actions
7. https://www.devopsschool.com/blog/android-app-build-and-publishing-workflow-using-github-actions/
8. https://github.com/marketplace/actions/automated-build-android-app-with-github-action
9. https://github.com/marketplace/actions/build-and-release-debug-apk
10. https://www.reddit.com/r/replit/comments/1jbf4wa/how_to_push_from_replit_github_to_back_up_app/