Initial Project Setup Options

1. React Native CLI (Bare Workflow)

```
npx react-native init ProjectName
```

- Creates a pure React Native project
- Full native code access (iOS/Android)
- Manual configuration required
- Direct native module integration
- Harder to set up, more flexible
- Can create standalone APK/IPA directly

2. Expo (Managed Workflow)

```
npx create-expo-app ProjectName
cd ProjectName
```

- Uses Expo framework
- Easier to start with
- Limited native code access
- Pre-configured features
- Simpler deployment
- Uses EAS for builds

Development Commands

Start Development Server

Expo Projects

```
# Start Expo development server
npx expo start
# -c flag to clean cache too

# Platform-specific starts
npx expo start --android  # Android
npx expo start --ios      # iOS (Mac only)
npx expo start --web      # Web version
```

```
# Install packages needed from files
npx expo install packagename
```

React Native CLI Projects

```
# Start Metro bundler
npm start

# Run on specific platforms
npm run android
npm run ios        # Mac only
```

Building & Device Deployment

Android Deployment

Virtual Device (Expo and RN CLI)

```
npm run android

# Multiple emulator handling
adb devices
npx react-native run-android --deviceId=DEVICE_ID
```

Physical Device (RN CLI)

```
npx react-native run-android
```

Converting Expo to Bare Workflow

```
# Option 1: Prebuild (recommended)
npx expo prebuild

# Clean prebuild (removes existing native folders)
npx expo prebuild --clean
```

Building APK/Release

Expo Projects

```
# Install EAS CLI (if not already installed)
npm install -g eas-cli

# Login to Expo account (if not already logged in)
```

```
eas login

# Initialize EAS in project
eas init

# Configure project for all platforms (creates eas.json automatically)
eas build:configure

# Example eas.json
{
  "cli": {
    "version": ">= 13.3.0",
    "appVersionSource": "remote",
    "requireCommit": true
  },
  "build": {
    "development": {
      "developmentClient": true,
      "distribution": "internal"
    },
    "preview": {
      "distribution": "internal"
    },
    "production": {
      "autoIncrement": true
    }
  },
  "submit": {
    "production": {}
  }
}

# Build APK (remote build)
eas build -p android --profile preview

# Build APK (local build, after prebuild)
eas build -p android --profile preview --local

# Build AAB (Android App Bundle)
eas build -p android --profile preview
```

React Native CLI Projects

```
# Build APK
npx react-native build-android --mode=release

# Build AAB
npx react-native build-android --mode=release
```

Key Differences

1. **React Native CLI (`react-native init`)**:
   - Full native code access
   - More complex setup
   - Direct native module integration
   - Better for apps requiring native customization
   - Can build APK directly using Gradle
2. **Expo (`create-expo-app`)**:
   - Simpler to start
   - Limited native functionality
   - Must use `prebuild` for native code access
   - Better for simpler apps
   - Uses EAS for building APKs
   - Can build locally after prebuild

Common Issues & Solutions

1. Missing Gradle configuration:
   - Check: Gradle Project Configuration Issue
2. SDK location not found:
   - Check: SDK Location Issue
3. EAS Build Issues:
   - Ensure you're logged in (`eas login`)
   - Verify `eas.json` configuration
   - Check Expo account status

Windows-Specific Steps (WORKING)

1. Set up WSL (Windows Subsystem for Linux) and install necessary tools.
2. Navigate to your project directory in WSL.

Prerequisites

- Windows 10/11
- Basic command line knowledge
- Existing Expo/React Native project

Step-by-Step Guide

1. Install WSL

```
# Open PowerShell as Administrator and run:
wsl --install
# After restart, install Ubuntu from Microsoft Store
```

2. Set Up Development Environment in WSL Ubuntu

```
# Install essential tools
sudo apt install curl
sudo apt install openjdk-17-jdk-headless

# Install Node.js via NVM
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.0/install.sh |
bash
source ~/.bashrc
nvm install --lts

# Install Gradle
wget https://services.gradle.org/distributions/gradle-8.5-bin.zip -P /tmp
sudo mkdir /opt/gradle
sudo unzip -d /opt/gradle /tmp/gradle-8.5-bin.zip
```

3. Configure Environment Variables

Add to `~/.bashrc`:

```
export JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64
export ANDROID_HOME=$HOME/android
export ANDROID_SDK_ROOT=${ANDROID_HOME}
export PATH=$PATH:/opt/gradle/gradle-8.5/bin:${ANDROID_HOME}/cmdline-
tools/latest/bin:${ANDROID_HOME}/platform-
tools:${ANDROID_HOME}/tools:${ANDROID_HOME}/tools/bin

source ~/.bashrc
```

4. Build Setup

```
# Install EAS CLI
npm install -g eas-cli

# Navigate to your project (replace with your path)
cd /mnt/c/YourProjectPath

# Initialize EAS
eas init
```

```
# Configure build
eas build:configure
```

5. Build Your App

```
# For AAB (Android App Bundle)
eas build --platform android --local

# For direct APK, first modify eas.json:
{
  "build": {
    "preview": {
      "android": {
        "buildType": "apk"
      }
    }
  }
}

# Then build APK
eas build --platform android --profile preview --local
```

6. Converting AAB to APK (if needed)

```
# Create keystore
keytool -genkey -v -keystore my-release-key.keystore \
  -alias my-key-alias -keyalg RSA -keysize 2048 -validity 10000

# Download bundletool
wget
https://github.com/google/bundletool/releases/download/1.15.6/bundletool-
all-1.15.6.jar \
  -O bundletool.jar

# Convert AAB to APK (single command)
java -jar bundletool.jar build-apks --bundle=your-app.aab --
output=my_app.apks \
  --mode=universal --ks=my-release-key.keystore --ks-pass=pass:your_password
\
  --ks-key-alias=my-key-alias --key-pass=pass:your_password

# Extract APK
unzip my_app.apks -d apk_output
```

(Optional) Converting AAB to APK on Windows

```
# Make sure you're in the correct directory
cd /mnt/c/App/ProjectName

# Create a keystore if you don't have one
keytool -genkey -v -keystore my-release-key.keystore -alias my-key-alias -
keyalg RSA -keysize 2048 -validity 10000

# Generate APK from your AAB
java -jar bundletool.jar build-apks --bundle=build-1732312561837.aab --
output=my_app.apks --mode=universal --ks=my-release-key.keystore --ks-
pass=pass:YOUR_KEYSTORE_PASSWORD --ks-key-alias=my-key-alias --key-
pass=pass:YOUR_KEY_PASSWORD

# Example command with sample password
java -jar bundletool.jar build-apks --bundle=build-1732312561837.aab --
output=my_app.apks --mode=universal --ks=my-release-key.keystore --ks-
pass=pass:gabriel --ks-key-alias=my-key-alias --key-pass=pass:gabriel

# Extract the universal APK
unzip my_app.apks -d apk_output
```

Fuckup commands

```
Remove-Item -Recurse -Force node_modules
Remove-Item package-lock.json
npm install --legacy-peer-deps
```