The process of increasing your app version can be both **manual** and **automatic**, depending on how you want to manage version updates. Here’s a breakdown of both approaches for various types of apps:

**1. Manual Version Update**

In most cases, **versioning is manually managed**, especially when you're preparing for a new release or update. However, there are tools and commands that can make it easier.

**How to do it manually:**

1. **React Native (Expo or CLI)**
   * Open app.json or app.config.js.
   * Manually update the version, versionCode (for Android), and buildNumber (for iOS).
2. **Node.js / Express Backend**
   * Open package.json and manually update the "version" field.
3. **Android / iOS Native Apps**
   * For Android, edit android/app/build.gradle (versionCode and versionName).
   * For iOS, edit ios/Info.plist (CFBundleShortVersionString and CFBundleVersion).
4. **React Web Apps**
   * Update the version field in package.json.

**2. Automatic Version Update**

Some tools and commands can **automate version increments**, such as npm version or **Expo CLI**.

**How to do it automatically:**

1. **Using npm version Command**  
   For a **Node.js/Express backend** or **React web app**, use the npm version command to automatically bump the version:
   * npm version patch (for minor updates or bug fixes)
   * npm version minor (for new features)
   * npm version major (for breaking changes)

This command:

* + Automatically updates the version in package.json.
  + Creates a **Git commit** and **Git tag** for the version.

Example:

npm version patch # Increments from 1.0.0 → 1.0.1

npm version minor # Increments from 1.0.0 → 1.1.0

npm version major # Increments from 1.0.0 → 2.0.0

1. **Expo (Managed Workflow)**
   * You can use **Expo CLI** to automate version updates when building and publishing:
   * expo prebuild --clean # Automatically updates the versionCode and buildNumber
2. **CI/CD Pipelines**  
   In professional environments, versioning is often handled automatically through **CI/CD pipelines**. Tools like **GitHub Actions**, **CircleCI**, or **Jenkins** can be configured to automatically update version numbers based on commit messages or tags.
   * **Example:**
     + A pipeline can increment the version automatically based on the git tag or the commit message ([version: patch]).
3. **Custom Scripts**  
   If you're working with a complex setup, you could write custom scripts to increase the version automatically (e.g., a script that increases the version number after each build or pull request).

**3. When Should You Do It Manually vs Automatically?**

**Manual Version Updates:**

* **When you need more control** over versioning, especially for major releases.
* **When managing app store releases** (iOS/Android), as app stores may have specific versioning requirements.
* **When you want to add extra metadata** (e.g., release notes or changelog entries).

**Automatic Version Updates:**

* **For minor updates, patches, or daily builds**, automatic versioning works well.
* **In CI/CD workflows** to ensure consistency and reduce human errors.
* **For projects with frequent deployments** or releases (e.g., weekly updates).

**Conclusion**

* **Manual versioning** is common for major updates and releases.
* **Automatic versioning** (using commands or CI/CD) is useful for minor updates, bug fixes, and continuous deployment.

You can use a combination of both, where **automatic versioning** handles smaller, frequent updates and **manual versioning** is used for significant releases.

Let me know if you need help setting up any automatic versioning!