# Creating new Electronite Release

**A. Notes:**

a. Before starting In the steps below:
   i. Replace the version `22.0.0` with the new Electron version you want to build on (see https://github.com/electron/electron/releases)
   ii. Also replace `v21.2.0` with latest release version

b. you need to build on Intel computer:
   i. building on Arm has problems - building for Intel targets would require emulation and run slowly. And currently no way to emulate MacOS Intel on Arm.

c. for each OS, it will take a day or two to build all the targets. This can be sped up by using multiple machines to build each target in parallel.

d. It's best to have a dedicated build PC or VM (maybe cloud VM which enables setting up multiple machines in parallel):
   i. Using a fast internet connection and fast drive it can take as little as 45 minutes to download and patch source files. Otherwise it can take over 4 hours on a clean PC.
   ii. doing the builds can use up to 100GB of disk space (particularly for arm64 builds). So it's best to have at least 160GB of primary disk space (80GB free after old builds are deleted) or have a second drive with 100GB of storage.
   iii. It's best to have at least 4 cores and a fast disk.
   iv. Best to have at least 16GB RAM and fast disk or the final link phase can take many hours with disk swapping.
   v. It looks like **Oracle Cloud Compute** is cheapest and fastest for doing Windows and Linux builds (no Mac). It's only about $10 a build for Windows and Linux architectures.
      - The best minimal configuration I tried was (each build takes 3 to 4 hrs):
        **Shape: VM.Standard.E3.Flex**
        **OCPU count:** 6
        CPU Model: AMD EPYC 7742 64-Core
        CPU Core: 12
        **Network bandwidth (Gbps):** 6
        **Memory (GB):** 32

   vi. **AWS Compute** looks much more expensive, but has MacOS support.

e. Tried using the same CircleCI automated build system that Electron uses, but it only works for MacOS and Linus, not WIndows. Plus it uses the GOMA build system to speed up the build process so builds don't time out. We can use GOMA setting cache-only, but it doesn't have the same level of performance. We could use higher level performance machines, but it would require a yearly license.

f. Note that the build process is always changing for Electron, so if it doesn't build you should check the build instructions at `` `docs/development` `` in the latest release.

**B. Create a new Electronite beta release:**

a. Creating new Electronite branch

1. If you haven't done this yet, you will need to clone https://github.com/unfoldingWord/electronite
2. Now cd into the Electronite folder

3. Create new electron branch:

```
# Get all upstream tags
git fetch --tags upstream
# get the source electron tag
git checkout v22.0.0-beta
# create a new electron branch from electron sources
git checkout -b v22.0.0-beta-electron
# push it up
git push origin v22.0.0-beta-electron
```

4. Create a new Electronite branch:

```
# create a new Electronite branch from electron sources
git checkout -b electronite-v22.0.0-beta
```

5. Copy files from Electronite branch of last release (such as `v21.2.0`):
    - copy files from `` `./electronite/docs/development/Electronite` ``
        - And update the version references from `v21.2.0` to `v22.0.0`

6. Copy `` `./electronite/patches/chromium/add_graphite.patch` `` from older Electronite branch
    - this may need to be updated later due to chromium changes

7. Copy the `` `electron.d.ts` `` from the `v22.0.0` electron version release assets to `` `./electronite/docs/development/Electronite` ``
    - In file rename instances of `'electron` with `'electronite`

8. Go through electronite.patch tweaking line numbers to match line numbers in new files.
    - Manually apply the patch to the files except for creating the file `` `./electronite/patches/chromium/add_graphite.patch` ``):
        edit `./DEPS`
        edit `./patches/chromium/.patches`
    - Copy `` `./electronite/patches/chromium/add_graphite.patch` `` from older build. This may have to be tweaked later if the step of fetching and patching sources fails.

9. search and replace in the source of old version number (such as `v21.2.0`) and replace with `v22.0.0`

10. Test by building using a Intel computer (currently Arm is not supported by build tools):
    - *Note: if this is a test rebuild, you will have to delete the cached Electronite repo:*
        - look in `` `git_cache` `` and delete the folder and lock file that contain `` `unfoldingWord/electronite` ``. Otherwise it will just use the older commits.
    - use build instructions in `` `./docs/development/Electronite` ``. Use appropriate instructions `` `MacBuildNotes.md` ``, `` `LinuxBuildNotes.md` ``, or `` `WindowsBuildNotes.md` ``. Follow steps for setting up the build environment (if not up to date). And follow the steps up to the fetching latest sources.

    - If fetching fails at `add_graphite.patch`:
        - Update/fix `add_graphite.patch` and push up to branch.
        - Delete source folder (e.g. `~/Develop/Electronite-Build/src`)
        - Start over again with test
    - If build fails on graphite due to cpp deprecations, then you will have to update our custom patch `add_graphite_cpp_std_iterator.patch`.
        - fix the compile error in the cpp source code in `.\src\third_party\graphite\graphite2`. *Note - We hope that eventually SIL will update their code, and then we can remove this patch from build scripts/batch.*
        - Once it compiles past graphite make an updated patch) by doing:
            - `cd src/third_party/graphite/graphite2 && git diff HEAD > ~/graphite.patch`
            - make copy of patch for Electronite: `cp ~/graphite.patch ~/add_graphite_cpp_std_iterator.patch`
            - in `~/add_graphite_cpp_std_iterator.patch` do search for instances of `a/src` and replace with `a/third_party/graphite/graphite2/src`
            - Also create issue in `https://github.com/silnrsi/graphite/issues/78` to share the fix with others and give content of `graphite.patch`
            - update `./electronite/docs/development/Electronite/add_graphite_cpp_std_iterator.patch` in ELectronite branch with contents of `~/add_graphite_cpp_std_iterator.patch` and commit/push
    - Make sure that `.\src\third_party\graphite` has been downloaded. Otherwise fix patches and start over with test
11. Commit and Push up your fixes to the Electronite branch
12. Do builds for Mac, Windows, and Linux using notes `MacBuildNotes.md`, `LinuxBuildNotes.md`, and `WindowsBuildNotes.md`.
    - *Note* - builds take up lots of disk space, so after building each target, copy the `dist.zip` folder from `./src/out/Release-*` and rename it appropriate to version/OS/architecture such as `electronite-v22.0.0-graphite-beta-win32-x64.zip`. Then delete `./src/out` before building for the next architecture.
    - If you run out of space, particularly building for arm64, you can delete the `git_cache` folder which should free up over 20GB of space.
13. At https://github.com/unfoldingWord/electronite/releases, create a draft release on Electronite using the previous release as a template (set it up to create a tag with name `v22.0.0-graphite-beta` when release is made).
    - Attach all the builds renaming the dist.zip file to match the tag name (`v22.0.0-graphite-beta`) and architecture of the build (again using the previous release as a template)
    - Attach the patched `electron.d.ts` from Electronite branch
14. Create a pre-release for testing

C. **Create new electronite-cli beta release:** https://github.com/unfoldingWord-dev/electronite-cli
   a. Checkout the latest release and create a branch `electronite-v22.0.0-beta`
      i. Update version in package.json to be `v22.0.0-graphite-beta`
      ii. Replace `` `electron.d.ts` `` with the patched `` `electron.d.ts` `` from above
      iii. Commit and push the changes
   b. Test package by deleting the dist folder, and then run `npm i --legacy-peer-deps && npm pack`, this should create a file such as `v22.0.0-graphite-beta.tgz`
   c. Create a test branch in translationCore (such as `` `feature-electronite-v22.0.0` ``)
      i. Import the electronite tgz file from step 2: `` `npm i --legacy-peer-deps --save-dev <<path-to-tgz>>` ``
      ii. Make sure translationCore will start up `` `npm i --legacy-peer-deps && npm start` ``
      iii. make sure tCore version is correctly displayed in app.
      iv. Check the log file to make sure it is running the right version of electron
      v. Make sure graphite is working:
         1. Look at this issue for how to test: https://github.com/unfoldingWord/translationCore/issues/6788
         2. Example `ar` project: https://git.door43.org/ElsyLambert/ar_new_phm_book
   d. npm publish the electronite-cli beta branch: `` `npm i --legacy-peer-deps && npm publish` ``
   e. update tCore to use the published package: `` `npm i --legacy-peer-deps --save-dev electronite@v22.0.0-graphite-beta` ``
   f. in package.json remove the ^ before Electronite version.
   g. Create builds and test:
      i. commit and push up changes
      ii. create a PR for this so github actions will try to create all the versions (ensures that all Electronite builds are named correctly)
      iii. test the builds - particularly macos x64 and windows x64:
         1. install and start up tCore builds.
         2. Check the log file to make sure it is running the right version of electron
         3. Make sure graphite is working:
            a. Look at this issue for how to test: https://github.com/unfoldingWord/translationCore/issues/6788
            b. Example `ar` project: https://git.door43.org/ElsyLambert/ar_new_phm_book
   h. At https://github.com/unfoldingWord/electronite-ci/releases, create a prelim release

D. **Creating the full release:**
   a. Make another Electronite draft release (https://github.com/unfoldingWord/electronite/releases) using the same branch as above `` `electronite-v22.0.0-beta` `` and select to create tag `v22.0.0-graphite` on release
      i. Attach the same Electronite builds as the prerelease beta, but rename them by removing beta from name.
      ii. Change description to take beta out of links
      iii. publish as a pre-release for the moment

b. Make another electronite-cli branch: https://github.com/unfoldingWord-dev/electronite-cli
    i. Checkout the latest branch `electronite-v22.0.0-beta` and create a new branch `electronite-v22.0.0`:
        1. Update version in package.json to be `v22.0.0-graphite`
        2. Commit and push the changes
    ii. Test package by deleting the dist folder, and then run `` `npm i --legacy-peer-deps && npm pack` ``, this should create a file such as `v22.0.0-graphite.tgz`
    iii. If that succeeds, commit changes and push up to new branch.
    iv. Make an electronite-cli release selecting to create tag `v22.0.0-graphite` when done
    v. change release description to take beta out.
    vi. Publish branch: `` `npm i --legacy-peer-deps && npm publish` ``

c. Make the Electronite (https://github.com/unfoldingWord/electronite/releases) pre-release for `v22.0.0-graphite` the latest release.

d. *Not sure this is needed anymore now that we directly check out the build branch in the build scripts:* ~~Merge released branch into graphite branch and push - lots of changes typically~~
    i. ~~checkout and update the released branch,~~
    ii. ~~merge in origin/graphite~~
    iii. ~~diff with released branch - download all changes from branch~~
    iv. ~~commit and push~~
    v. ~~make PR for merge to graphite~~
    vi. ~~merge PR~~