Cutting a Deno release

Pre-flight checklist

- [ ] An up to date stable Rust toolchain
- [ ] A binary version of `deno` available (hopefully built from `main`) that is going to be available throughout any local building you might do.
- [ ] Forks and local clones of `denoland/deno`, `denoland/deno_std`, `denoland/dotland`, `denoland/docland`, `denoland/deno_docker` `denoland/manual`
- [ ] Ensure that external dependencies are up-to date in `denoland/deno` (e.g. `rusty_v8`, `serde_v8`, `deno_doc`, `deno_lint`).
- [ ] Ownership access on crates.io for the 19 () crates that you will be publishing. (Don't worry too much though as the main script publishing 18 of the crates allows recovery)
- [ ] Lot's of 

During this process `main` branch (or any other branch that you're creating release from) should be frozen and no commits should land until the release is cut.

Before starting the process write a message in company's #general channel: `:lock: deno and deno_std are now locked`

Updating `deno_std`

1. Go to the "version_bump" workflow in the deno_std repo's actions: https://github.com/denoland/deno/actions/workflows/version_bump.yml
2. Click on the "Run workflow" button.
    1. For the kind of release, select "minor".
    2. Run the workflow.
3. A PR will be automatically created. Follow the checklist in the PR and review it.
4. Merge the PR. While doing so, ensure that the commit name is exactly the version name. Eg. `0.121.0`, not `0.121.0 (#1810)`.
5. Wait for the CI run to complete which will tag the repo and create a draft release. Review the draft release and then publish it.

Updating the main repo

If you are cutting a patch release: First you need to sync commit to the relevant minor branch, so if you are cutting a `v1.17.3` release you need to sync `v1.17` branch.

To do that, you need to cherry-pick commits from the main branch to the `v1.17` branch. For patch releases we want to cherry-pick all commits that are not `feat` commits. Check what was the last commit on `v1.17` branch before the previous release and start cherry-picking newer commits from the `main`.

Once all relevant commits are cherry-picked, push the branch to the upstream and verify on GitHub that everything looks correct.

Phase 1: Bumping versions

1. After releasing deno_std, go to the "version_bump" workflow in the CLI repo's actions: https://github.com/denoland/deno/actions/workflows/version_bump.yml
2. Click on the "Run workflow" button.
    1. In the drop down, select the minor branch if doing a path release or the main branch if doing a minor release.
    2. For the kind of release, select either "patch", "minor", or "major".
    3. Run the workflow.
3. Wait for the workflow to complete and for a pull request to be automatically opened.
4. Review the pull request and make any necessary changes.
5. Merge it.

Phase 2: Publish

1. Go to the "cargo_publish" workflow in the CLI repo's actions: https://github.com/denoland/deno/actions/workflows/cargo_publish.yml
2. Run it on the same branch that you used before and wait for it to complete.
3. This CI run create a tag which triggers a second CI run that publishes the GitHub draft release.

   The CI pipeline will create a release draft on GitHub (https://github.com/denoland/deno/releases). Update the draft with the contents of `Releases.md` that you previously added.
4. Upload Apple M1 build (`deno-aarch64-apple-darwin.zip`) to the release draft and to https://console.cloud.google.com/storage/browser/dl.deno.land

   ```
   cargo build --release
   cd target/release
   zip -r deno-aarch64-apple-darwin.zip deno
   ```
5. Publish the release on Github
6. Update the Deno version on the website by updating https://github.com/denoland/dotland/blob/main/versions.json.
7. Push a new tag to `manual` . The tag must match the CLI tag; you don't need to create dedicated commit for that purpose, it's enough to tag the latest commit in that repo.
8. For minor releases: make sure https://github.com/mdn/browser-compat-data has been updated to reflect Web API changes in this release. Usually done ahead of time by @lucacasonato.
9. **If you are cutting a patch release**: a PR should have been automatically opened that forwards the release commit back to main. If so, merge it. If not and it failed, please manually create one.

Updating `doc.deno.land`

This should occur after the Deno CLI is fully published, as the build script queries the GitHub API to determine what it needs to change and update.

1. Goto the cloned report for `denoland/docland`.
2. Checkout a new branch (e.g. `git checkout -b deno_1.17.0`).
3. Execute `./build.ts` (or `deno run --config deno.jsonc --import-map import-map.json --allow-read=. --allow-write=./static --allow-net build.ts`).
4. Commit changes and raise a PR on `denoland/docland`.
5. Merging the approved PR will trigger deployment to Deploy of the updates.

Updating `deno_docker`

1. Open a PR on the `deno_docker` repo that bumps the Deno version in all Dockerfiles, the README and the example Dockerfile
2. Create a tag with the version number (*without* `v` prefix).

Write a message in company's #general channel: `:unlock: deno and deno_std are now unlocked`.