Maintaining the Neovim project

Notes on maintaining the Neovim project.

General guidelines

- Decide by cost-benefit
- Write down what was decided
- Constraints are good
- Use automation to solve problems
- Never break the API... but sometimes break the UI

Ticket triage

In practice we haven't found a way to forecast more precisely than "next" and "after next". So there are usually one or two (at most) planned milestones:

- Next bugfix-release (1.0.x)
- Next feature-release (1.x.0)

The forecasting problem might be solved with an explicit priority system (like Bram's todo.txt). Meanwhile the Neovim priority system is defined by:

- PRs nearing completion.
- Issue labels. E.g. the `+plan` label increases the ticket's priority merely for having a plan written down: it is *closer to completion* than tickets without a plan.
- Comment activity or new information.

Anything that isn't in the next milestone, and doesn't have a finished PR—is just not something you care very much about, by construction. Post-release you can review open issues, but chances are your next milestone is already getting full... :)

Release policy

Release "often", but not "early".

The (unreleased) `master` branch is the "early" channel; it should not be released if it's not stable. High-risk changes may be merged to `master` if the next release is not imminent.

For maintenance releases, create a `release-x.y` branch. If the current release has a major bug:

1. Fix the bug on `master`.
2. Cherry-pick the fix to `release-x.y`.
3. Cut a release from `release-x.y`.
    - Run `./scripts/release.sh`
    - Update (force-push) the remote `stable` tag.

- The nightly job will update the release assets based on the `stable` tag.

The neovim repository includes a backport github action. In order to trigger the action, a PR must be labeled with a label matching the form `backport release-0.X`. If the label is applied before the PR is merged, the backport will be filed automatically against the target branch. Otherwise, comment `\backport` on the merged PR *after* the label has been applied to trigger a backport. Note, the PR must have a description in the issue body, or the backport will fail.

Third-party dependencies

These "bundled" dependencies can be updated by bumping their versions in `third-party/CMakeLists.txt`: - Lua - LuaJIT - Luv - libtermkey - libuv - libvterm - lua-compat - tree-sitter

`scripts/bump-dep.sh` is a script that can automate this process for `LuaJIT`, `Luv`, `libuv` & `tree-sitter`. See usage guide: - Run `./scripts/bump-deps.sh --dep Luv --version 1.43.0-0` to update a dependency. See `./scripts/bump-deps.sh -h` for more detailed usage - Run `./scripts/bump-deps.sh --pr` to create a pr To generate the default PR title and body, the script uses the most recent commit (not in `master`) with prefix `build(deps):`

These dependencies are "vendored" (inlined), we need to update the sources manually: - libmpack - xdiff - lua-cjson - Klib

We also maintain some forks, particularly for Windows, if we are waiting on upstream changes: https://github.com/neovim/neovim/wiki/Deps

See also

- https://github.com/neovim/neovim/issues/862
- https://github.com/git/git/blob/master/Documentation/howto/maintain-git.txt