Atom Release Build Documentation

Overview

This folder contains build configuration and scripts for automating Atom's release pipeline using Visual Studio Team Services. VSTS allows us to leverage multi-phase jobs to generate Atom installation packages on Windows, macOS, and Linux and then publish a new release automatically once the build completes successfully.

Nightly Release Build

Our scheduled nightly release uses a mutli-phase job to automatically generate Atom Nightly installation packages and then publish them to GitHub and atom.io.

The Atom Nightly build definition is configured with the `nightly-release.yml` file. More information on VSTS' YAML configuration format can be found in their Getting Started documentation.

Versioning Phase

In this phase, we run `script/vsts/generate-version.js` to determine the version of the next Atom Nightly release. This script consults the GitHub v3 API to get the list of releases on the `atom/atom-nightly-releases` repo. We look for the most recent, non-draft release and then parse its version number (e.g. `1.30.0-nightly4`) to extract the base version and the monotonically-increasing nightly release number.

Once we have the version and release number, we compare the base version number (`1.30.0`) against the one in `package.json` of the latest commit in the local repo. If those versions are the same, we increment the release number (`1.30.0-nightly5`). If those versions are different, we use `0` for the release number to start a new series of Nightly releases for the new version (`1.31.0-nightly0`).

Once the release version has been determined, it is set as our custom `ReleaseVersion` output variable by writing out a special string to `stdout` which is recognized by VSTS. This variable will be used in later build steps.

If any part of the build process fails from this point forward, the same version number *should* be chosen in the next build unless the base version number has been changed in `master`.

OS-specific Build Phases

In this part of the build, we use phase templates for Windows, macOS, and Linux to build Atom simultaneously across those platforms and then run the Atom test suite to verify the builds. If build, test, and linting come back clean,

we take the build assets generated in the `out` folder on each OS and then stage them as build artifacts.

For each OS build, we refer to the `ReleaseVersion` variable, set in the previous phase, to configure the `ATOM_RELEASE_VERSION` environment variable to override the version contained in Atom's `package.json`.

Publish Phase

If all three OS builds have completed successfully, the publish phase will launch the `script/publish-release` script to collect the release artifacts created from those builds and then upload them to the S3 bucket from which Atom release assets are served. If the upload process is successful, a new release will be created on the `atom/atom-nightly-releases` repo using the `ReleaseVersion` with a `v` prefix as the tag name. The release assets will also be uploaded to the GitHub release at this time.