Automate and finish the setup of the Jenkins WASM tinderbox

The goal is to have a tinderbox, which builds the feature/wasm branch, deploys a package to the "daily builds" directory and serves the latest WASM build statically. This webserver can easily be separate from the TB.

Now that LODE is patched to setup the WASM build environment, a few steps are needed for an automated setup via Salt:

Lode changes for WASM

- lode/init.sls
  - lode-pkg eventually needs wget
  - No libreoffice build dependencies are needed
  - Default branch is feature/wasm
  - setup needs the additional option --wasm-local

Static webserver setup

- nginx/*
  - Two additional headers must be served for thread support:
    - add_header Cross-Origin-Opener-Policy same-origin
    - add_header Cross-Origin-Embedder-Policy require-corp
  - https is needed for the browser clipboard API
  - mime.types should contain an entry for application/wasm wasm; to enable browser caching.

For the actual build we need to:

- source $LODE_HOME/emsdk/emsdk_env.sh
- export QT5DIR=$LODE_HOME/qt5/qtbase
- Run autogen.sh --with-distro LibreOfficeWASM32 (AKA --host=wasm64-local-emscripten)
- Run make

All the required files can be bundled via:

```
tar -chf wasm.tar --xform 's/.*program/lo-wasm/' instdir/program/soffice.* instdir/program/qt*
```

FYI: this includes the file `qtlogo.svg`, which is shown while downloading the data before running the WASM app. Obviously this can be easily be replaced with whatever "branding" we want to display on load ;-) 

For a debug build xz saves ~1/3 compared to gzip.

Eventually one might rename qt_soffice.html to index.html

History

#1 - 2021-11-01 16:00 - Jan-Marek Glogowski
- Description updated

#2 - 2021-11-22 15:21 - Florian Effenberger
- Assignee set to Christian Lohmaier
Seems there was some confusion around this ticket. I've talked to Guilhem and Cloph, let me try to summarize (and Cloph, please correct me if I write something wrong :).

Happy to provide a box or two, of course - sorry for the delay on this.
From what I understand, you currently need a box that you can manually tweak in production and configure, change settings etc., so nothing that (yet) can go into SaltStack for deployment, as any of your changes would be quickly overwritten by SaltStack.

Buildbots/tinderboxes are behind the VPN and not publicly accessible, so they can stay out of deployment and can be tweaked manually as you wish, but they are only accessible internally with the VPN.
If the box is public-facing, we need some sort of deployment in the long run, to manage the machines, which limits the amount of manual tweaks.

Proposal therefore: We can provide a box (either VM or Hetzner-hosted) you can play with, when it is fully tweaked, we then reinstall and turn into a salted machine, and/or split into two different machines (one salted public box, one non-salted behind VPN).

Would that work for you?

#3 - 2021-11-22 16:50 - Jan-Marek Glogowski
- Description updated

Florian Effenberger wrote in #note-2:

From what I understand, you currently need a box that you can manually tweak in production and configure, change settings etc., so nothing that (yet) can go into SaltStack for deployment, as any of your changes would be quickly overwritten by SaltStack.

FWIW: the described setup should work right now with SaltStack. It should be an automated TB, installed with the described Salt modification with the LODE setup and without any manual tweaks. We know we can build LO WASM via LODE. And actually I would want to have two TBs at some point: one for a debug build, one a production build. If one is working, the 2nd is basically a clone. The difference is just a single flag to autogen.input: --enable-dbgutil. This box can be completely internal (AKA just reachable via SSH jumpserver).

But as a start / currently: we need to update LODE more or less often. I found that Qt WASM in any branch is much buggier then any binary Qt build, even basic stuff :-). So a developer (and any TB) will need to patch Qt "somehow". Also Emscripten doesn't have something like a stable branch. I don't know how often LODE is updated on TDF boxes, so for the time being the already provided box seems better. I currently patch Qt WASM daily and even Emscripten -> Windows ARM64 is much more stable...

Buildbots/tinderboxes are behind the VPN and not publicly accessible, so they can stay out of deployment and can be tweaked manually as you wish, but they are only accessible internally with the VPN.

That is fine. As I wrote we just need a static web server to deploy the result, which means my provided packaging command. Then copy that package to the daily repo and unpack it to the static web server. I've described the needed nginx setup in the ticket text. We just need to define, how that data transfer will happen.

If the box is public-facing, we need some sort of deployment in the long run, to manage the machines, which limits the amount of manual tweaks.

The public-facing web server is just that: a web server with https and two additional headers for responses. It just serves static downloads from a single directory (or two directories for release and debug builds).

Proposal therefore: We can provide a box (either VM or Hetzner-hosted) you can play with, when it is fully tweaked, we then reinstall and turn into a salted machine, and/or split into two different machines (one salted public box, one non-salted behind VPN).

Would that work for you?

Yup. Separate boxes, like your 2nd proposal for web server (public, Salt'ed) and the TB (hidden, currently manual) would generally be my preference too. I don't know if a real, extra web server box is needed or you can expand some existing box, like the download sever (if it's not a CDN), but that is completely up to infra. There is no need to expose the TB to the web.

So in addition: the Salt'ed web server automatically updates it's deployment / serving directory from the daily build directory once an hour. TB's already fill an upload directory, so some mechanism must already exist and work. This just needs some coordination. On that already provided box, we can run a cron'ed TB job on for the time being.

#5 - 2021-11-25 11:50 - Christian Lohmaier
FYI: setup cannot be deployed on the centOS baseline, the upstream binaries require glibc 2.18, while CentOS 7 only has 2.17 - so will keep in on the ubuntu machine for now.

Also there's another problem that the build fails with --enable-werror (which is set by configure.ac when LODE_HOME is set, which is the case for the regular jenkins configuration) in the added lockfile tool (and maybe other places). Can add a workaround for the build job though, but ideally should be
fixed in the branch so it can build with --enable-werror

#6 - 2021-11-29 11:28 - Christian Lohmaier
test-webhost at https://wasm-test.libreoffice.org/ however there's another build issue, namely that its not possible to build in builddir != srcdir

static/CustomTarget_wasm_fs_image.mk simply disregards the prefix when creating the filelist for the packaging command:

```bash
$(wasm_fs_image_WORKDIR)/soffice.data.filelist: $(gb_wasm_image_filelist) | $(wasm_fs_image_WORKDIR)/.dir

TEMPFILE=$(call gb_var2file,$(shell $(gb_MKTEMP)),1,$(subst $(SRCDIR)/,,$(subst $(BUILDDIR)/,,$^))) &&
\    mv $$TEMPFILE $@.tmp
$(call gb_Helper_replace_if_different_and_touch,$@.tmp,$@)
```

and thus it fails since the sample document (android/default-document/example3.odt) is in SRCDIR, not in BUILDDIR.

#7 - 2021-12-09 11:36 - Florian Effenberger
Jan-Marek: Anything else needed from our side, is all working for you?

#8 - 2022-01-13 11:52 - Florian Effenberger
- Status changed from New to Closed

Seems to work fine, just talked to Cloph - closing here

#9 - 2022-02-09 16:54 - Jan-Marek Glogowski
- Description updated

Firefox has a setting to show additional information about WASM processing in the developer tools console in about:config:
javascript.options.wasm_verbose

This should show at best WebAssembly verbose: async instantiate succeeded (loaded from cache). For The wasm-test side, at least the (loaded from cache) is missing. That is happening, because WASM files should return application/wasm as a mime type; currently it's the default binary blob application/octet-stream.

#10 - 2022-02-09 16:55 - Jan-Marek Glogowski
- Description updated

#11 - 2022-02-10 11:18 - Florian Effenberger
- Target version changed from Q4/2021 to Q1/2022

Cloph, something for you or hand over to Guilhem for web server configuration?

#12 - 2022-02-14 11:41 - Christian Lohmaier
added the mime type (FYI: chrome still doesn't cache the file, but works for firefox now)