Updating Buck

May 08 2023

Hey there! A shorter post today, but I wanted to continue my series on Buck by going over some things that have changed since this series started.

A series

This post is part of a series:

- Using buck to build Rust projects (using-buck-to-build-rust-projects)
- Using Crates.io with Buck (using-cratesio-with-buck)
- Updating Buck (you are here)

Updating Buck

Lots of great stuff has been happening since the initial release of buck2, and we'd like to take advantage of that. If we try and update things, though, we'll get (well, you might not, but I did) an error:

```
~> cargo +nightly-2023-03-07 install --git https://github.com
<snipped>
   Replacing C:\Users\steve\.cargo\bin\buck2.exe
error: failed to move `C:\Users\steve\.cargo\bin\cargo-instal

Caused by:
  Access is denied. (os error 5)
```

(Why yes, my console prompt has changed...)

See on Windows, we can't replace a program that's running, while it's running. On other systems, this may work for you. However, it's probably a good idea to not take advantage of this particular feature, because of the underlying cause: buck runs a daemon in the background. This daemon is still going. Trying to replace it while it's running means you'll still have the old one going around in the back, and while that's fine (is it? I don't actually know), best to cleanly shut down first. So do this:

```
~\Documents\GitHub\buck-rust-hello> buck2 killall
Killed buck2.exe (4788). C:\Users\steve\.cargo\bin\buck2.exe
Killed buck2.exe (40200). C:\Users\steve\.cargo\bin\buck2.exe
```

`buck2 killall` will kill every instance of the daemon. As you can see, it found two of mine, and shut them down. As you can see, you want to run this command from within one of your projects.

And now an upgrade works.

Update Reindeer

If you're building Rust code and using Reindeer, like we talked about in the last post, go ahead and grab that too. Lots of good stuff in there:

```
~> cargo install --git https://github.com/facebookincubator/r
```

Now that we're all updated, let's fix up our project to make use of the latest changes that are relevant.

Update the prelude

I no longer need my weird toolchain hacks to get MSVC Rust working. Thank you dtolnay! 🙏

Let's pull in changes to the prelude:

```
~\Documents\GitHub\buck-rust-hello> cd prelude
~\Documents\GitHub\buck-rust-hello\prelude> git fetch origin
remote: Enumerating objects: 28, done.
remote: Counting objects: 100% (28/28), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 17 (delta 12), reused 15 (delta 10), pack-reuse
Unpacking objects: 100% (17/17), 3.60 KiB | 102.00 KiB/s, don
From https://github.com/facebook/buck2-prelude
   920d3f2..370cd4d  main       -> origin/main
~\Documents\GitHub\buck-rust-hello\prelude> git reset --hard
HEAD is now at 370cd4d Http_archive execution platform hack
~\Documents\GitHub\buck-rust-hello\prelude> cd ..
~\Documents\GitHub\buck-rust-hello>
```

And remove the hack (if you had to do it too) in `toolchains\BUCK`:

```
system_cxx_toolchain(
    name = "cxx",
    visibility = ["PUBLIC"],
)
```

No more `link.exe` stuff!

Fix some configuration

Some good changes have landed in the `buck2 init` subcommand that we'll want to add ourselves.

First up, we should add an empty file named `.buckroot` to the root of our repository (https://github.com/facebook/buck2/commit/54e1eb5e29eca8a021579b859d0a34c73754eb2a). Because buck works with your filesystem hierarchy, it can and will traverse upwards looking for things at times. Adding this file ensures that if it does so, it will stop before it starts trying to traverse even higher. There's no need for anything in the file, as the contents are ignored.

I think this kind of change is an interesting way to look at the usability of various systems. Adding an empty file here is *sort of* "more complex" than say, Cargo. But it's also more explicit. Which means it can be more… I want to say "legible", if you also read insufferable books like I do sometimes. And therefore, easier. Anyway, more thoughts on all of this in the future.

Next, we have a change that is demonstrated by this example. Can you guess what it is?

```
~\Documents\GitHub\buck-rust-hello> buck2 run //src/bin:hello
File changed: root//.git/modules/prelude
Build ID: 1a24b418-acdb-4d23-a5e1-6e9b644c01e6
Jobs completed: 83. Time elapsed: 1.9s. Cache hits: 0%. Comma
```

There's no reason for buck to be watching our `.git` directory for changes. And that's why it is now put in the default configuration (https://github.com/facebook/buck2/commit/2083261102e867bce2b6385b78b598e758b424a8) when you `buck2 init --git`. But we've gotta make that change ourselves. Open up your `.buckconfig` and add this at the bottom:

```
[project]
ignore = .git
```

We want it to ignore the `.git` directory. Seems good.

... I lied though, there's one more improvement we want to make: we also don't want buck to bother listening to the `target` directory either, as those files are for Cargo's output. So what we *actually* want is this:

```
[project]
ignore = .git, target
```

After doing that we'll want to `buck2 kill` to shut the daemon down, so that it can pick up our new configuration on the next boot.

Regenerating reindeer

Since we've got new bugfixes in Reindeer too, let's regenerate our config for our dependencies:

```
~\Documents\GitHub\buck-rust-hello> reindeer --third-party-di
[WARN  reindeer::fixups] semver-1.0.17 has a build script, bu
```

We still have to deal with the build script! We didn't talk about the contents of `third-party\BUCK` last time, and we won't this time either. If you want to see what's changed, you can take a peek though. One change that we didn't explicitly talk about before, but you may end up noticing, is that it did not generate a target to try and build our build script.

Let's try it out now:

```
~\Documents\GitHub\buck-rust-hello> buck2 run //src/bin:hello
File changed: root//BUCK
File changed: root//third-party/BUCK
File changed: root//third-party
Build ID: 2eb0c121-d1fb-43d2-b8a4-f923d8dda657
Jobs completed: 20. Time elapsed: 1.1s. Cache hits: 0%. Comma
```

Nice. Also note that we aren’t seeing any more `.git` or `target` changes, not that we’ve run anything that would inherently change those files, but go ahead, invoke Cargo or git, and then build again. You shouldn’t see notifications about those directories anymore.

Fixing some breakage

Speaking of invoking Cargo, remember how I said this in the last post?

> *“We do have two different Cargo.tomls now. That is a bit of a bummer. But at least it is easy to determine if there’s a problem: dependency failures are loud, and if you’re building with both in CI, you’ll notice if stuff goes wrong. There also may be a solution to this I’m just not aware of.”*

Well…

```
~\Documents\GitHub\buck-rust-hello> cargo run
   Compiling hello_world v0.1.0 (C:\Users\steve\Documents\Git
error[E0432]: unresolved import `semver`
 --> src\bin\main.rs:1:5
  |
1 | use semver::{BuildMetadata, Prerelease, Version, VersionR
  |     ^^^^^^ use of undeclared crate or module `semver`

For more information about this error, try `rustc --explain E
error: could not compile `hello_world` due to previous error
```

Going back and re-reading my last post, I did have a `cargo add` `semver` in there, so maybe I just forgot to commit that in my last post. Just in case, we'll fix that:

```
~\Documents\GitHub\buck-rust-hello> cargo add semver
    Updating crates.io index
      Adding semver v1.0.17 to dependencies.
             Features:
             + std
             - serde
```

With that, `cargo build` and `cargo run` are back in business.

We also have... well "breakage" isn't exactly right, but we have a buck configuration issue. Let's try to build every target:

```
~\Documents\GitHub\buck-rust-hello> buck2 build ...
Error running analysis for `root//:build (prelude//platforms:

Caused by:
    0: Error looking up configured node root//:build (prelude
    1: `root//src/bin:hello_world` is not visible to `root//:
Build ID: 51657a07-112c-46b4-a2eb-91d60a4b0aed
Jobs completed: 8. Time elapsed: 0.0s.
BUILD FAILED
```

We didn't declare that our binary was visible anywhere, and so when we try and build it, it isn't happy. We do want this to be public, so change `src\bin\BUCK` by adding this `visibility` line near the end. It should look like this:

```
rust_binary(
    name = "hello_world",
    srcs = ["main.rs"],
    crate_root = "main.rs",
    deps = [
        "//third-party:semver",
    ],
    visibility = ["PUBLIC"],
)
```

And now that will work:

```
~\Documents\GitHub\buck-rust-hello> buck2 build ...
File changed: root//src/bin/BUCK
Build ID: 9da23da4-3f99-4e78-8c58-ae5e2f1facaa
Jobs completed: 25. Time elapsed: 0.6s. Cache hits: 0%. Comma
BUILD SUCCEEDED
```

Okay, now that we’ve fixed both Cargo *and* buck… let’s make sure that isn’t gonna happen again. We aren’t testing any of this. So it broke. Just like I said it was easy to not let break. Sigh.

We’re going to use GitHub Actions because this is already on GitHub. I’m sure you can adapt it to your setup of choice.

Put this in `.github\workflows\ci.yml`:

```
name: CI

on:
  pull_request:
  push:
    branches: [main]

jobs:
  test:
    name: ${{matrix.name || format('Rust {0}', matrix.rust)}}
```

```yaml
    runs-on: ${{matrix.os || 'ubuntu'}}-latest
    strategy:
      fail-fast: false
      matrix:
        include:
          - rust: nightly
          - rust: beta
          - rust: stable
          - rust: 1.69.0
          - name: Cargo on macOS
            rust: nightly
            os: macos
          - name: Cargo on Windows (msvc)
            rust: nightly-x86_64-pc-windows-msvc
            os: windows
    steps:
      - uses: actions/checkout@v3
      - uses: dtolnay/rust-toolchain@master
        with:
          toolchain: ${{matrix.rust}}
      - run: cargo check
      - run: cargo run

  buck:
    name: Buck2 on ${{matrix.os == 'ubuntu' && 'Linux' || mat
    runs-on: ${{matrix.os}}-latest
    if: github.event_name != 'pull_request'
    strategy:
      matrix:
        os: [ubuntu, macos, windows]
    steps:
      - uses: actions/checkout@v3
        with:
          submodules: true
      - uses: dtolnay/rust-toolchain@stable
        with:
          components: rust-src
      - uses: dtolnay/install-buck2@latest
      - run: buck2 build ...
      - run: buck2 run //src/bin:hello_world

      - uses: actions/cache/restore@v3
        id: cache
        with:
          path: ~/.cargo/bin/reindeer${{matrix.os == 'windows
          key: ${{matrix.os}}-reindeer

      - run: cargo install --git https://github.com/facebooki
        if: steps.cache.outputs.cache-hit != 'true'

      - uses: actions/cache/save@v3
        if: steps.cache.outputs.cache-hit != 'true'
        with:
          path: ~/.cargo/bin/reindeer${{matrix.os == 'windows
          key: ${{steps.cache.outputs.cache-primary-key}}

      - run: reindeer buckify
        working-directory: third-party
        if: matrix.os == 'ubuntu'
      - name: Check reindeer-generated BUCK file up to date
        run: git diff --exit-code
        if: matrix.os == 'ubuntu'
```

This is... you guessed it, based off of dtolnay's CI for cxx (https://github.com/dtolnay/cxx/blob/master/.github/workflows/ci.yml). What can I say, he writes good code. This version is a bit stripped down, since this is primarily a project for showing off a build system, rather than a regular project. This has:

- Builds with Cargo on Linux against nightly, beta, stable, and 1.69.0 as a sort of MSRV check.
- Builds with Cargo on MacOS and Windows against nightly Rust.
- Buck on Linux, MacOS and Windows against stable Rust.

- On Linux, checks that we have comitted any relevant changes to dependencies.

This does not have:

- A “fail if there are any warnings” build
- A clippy build
- A full matrix of every os/language/build system combination

or other things you may want out of a build. Again, you’ll probably want to customize this heavily, but this is what I’m going to do here.

Wrapping up

And with that, we are done! Next post we’re going to deal with that build script. And use some crates that have more intense dependencies, and get them all working.

As always, you can check out the code (https://github.com/steveklabnik/buck-rust-hello/commit/914e3aa7cdc269b2536f8118255eae12bf552f8d) at this point if you’d like.