I found my self in this area by accident, I was working on harvey and the builds were very slow, so I voulenteered to rewrite the build system.

I thought, how hard could this be?

It was hard in all the ways I couldn't think of and a lot easier in pretty much the same unpredictable ways.

Here are some of the things I've learned:

Most people think they are able to build a build-system. And they aren't wrong.

Building a build-system isn't hard. As a matter of fact, as a matter of fact, you probably built a few your self.

Recall that all build-systems do the exact same thing; map source files to intermediate files and reduce intermediate files to a single output file.

So if you've ever written a shell script that runs gcc, congratulations! You've built a build-system. Not a generic build-system, a bespoke one, but a build-system nevertheless.

I'll take this claim a bit further and say that if you don't know how to build a build-system, you probably don't know how to write software, cause if you're writing software you can't build, what are you even doing?

But don't take my word for it, here are some very successful build-systems, written JS, alone

- webpack
- babel
- rollup
- parcel

These are not unsuccessful build-systems, infact I would say they are more successful and have better ergonmoics than the blazeclass of build systems (bazel, pants, buck...).

The one thing blaze class of systems has going for it is that it is a distributed build system, which the aforementioned don't.

So if you want to build a build-system that will only be building on a single machine, stop reading this, go build your build system, don't even think about using blaze. You'll be fine.

If at some point, your project will need to horizontally scale, just switching to bazel alone will not be enough. As a matter of fact by distribuding your build you'll be adding an overhed to your build, which will be the transfer of intermeriary artifacts from one machine to another. So care needs to be taken, attention needs to be paid when breaking builds in to smaller pieces, as it may have a concealed adverse effect.

In the next post, I’ll write down a strategyI discovered migrating large scale projects.