Synthesized Targets

Synthetic targets to tie up loose ends

Background

Pants uses dependencies between targets to track relationships between source files and packaged artifacts. These dependencies form a digraph that is consulted by Pants in order to determine when an artifact needs to be repackaged as a result of a changed input source file (or other dependency).

Problem Statement

For Pants to be able to correctly deduce when an artifact has been invalidated, the digraph with all dependencies must be complete, as any omissions may result in an artifact to not be rebuilt when it ought to have been when using invalidation (such as the changed since feature).

Due to dependency inference and injection, Pants have a good view of the world with the notable exception of configuration files and lock files. These are included in the cache key, so when Pants is asked to actually package an artifact the correct action is taken, but the invalidation is not (i.e. using `--changed-since` will not result in the artifact to be rebuilt on config file changes when it should be).

Proposed Solution

Introduce a new feature that can be used to populate the remaining missing pieces for configuration files and lock files that Pants may then infer dependencies upon for relevant targets. These would be “synthetic targets” synthesized from known sources such as the Pants configuration.

Design

Goals

- Unobtrusive (don’t break stuff in case of conflicts)
- Overridable/customizable (let’s not assume anything)
- Opt-in/Opt-out(?)

non-Goals

- TBD..

The implementation for synthesized targets uses unions to request synthetic target adaptors for injection during BUILD file parsing. That implies that synthetic target adaptors live in the same space as regular target adaptors. In case of conflicting addresses between a synthetic target adaptor and one declared in a BUILD file, the declared target adaptor wins (unobtrusive). When a declared target adaptor wins, the field values from the synthetic target adaptor are used as defaults for undefined fields in case the declared target provided a `_extend_synthetic=True` field (customizable).
Any `__defaults__` are applicable also to synthetic targets.

In order to help differentiate between a target that is instantiated from a synthesized source, there may be an `is_synthetic` flag on the target address that can be used for filtering and presentation purposes.

Alternatives considered

New dedicated target types that can be manually added or automatically added by tailor to own the configuration and lock files, with the undesirable effect of having more boilerplate and noise in the BUILD files. The one up-side being transparency; less magic going on.

Lessons learned from Pants v1

It would be beneficial to reiterate the design, pros and cons of how this worked in Pants v1 to help guide this effort.

Benjy: Pants v1 had synthetic targets. They were used primarily during codegen. A python_library would be synthesized to own the generated .py files, and then stitched into the dependency graph at the same place as the proto_library from which they were generated. This caused issues with the dependencies not necessarily making sense. We've eliminated this entire issue in v2, because codegen support is significantly more robust