# Pants 2.9: Alpha support for Java and Scala, improvements for Docker and Go, and more

We're pleased to announce [Pants 2.9.0](https://www.pantsbuild.org/v2.9/docs), the latest release of Pants, the scalable and ergonomic build system.

To update, set `pants_version = "2.9.0"` in your `pants.toml`. See [upgrade tips](https://www.pantsbuild.org/docs/upgrade-tips) for more information.

## Alpha Support for Java and Scala

We're very happy to announce that support for Java and Scala has reached alpha quality in the `2.9.0` release!

Pants 1.x had a long history of support for Java and Scala, going back to when it was first created at Twitter. In fact: they were the first supported languages! Consequently, (re-)adding support for these popular JVM languages has been high on our list ever since [the 2.x release](https://blog.pantsbuild.org/introducing-pants-v2/) in late 2020.

### Improvements over Pants 1.x

In the last few years, we've learned a lot about how best to deal with more-slowly-compiling, high level languages like Scala.

#### Dependency inference and per-file compilation

As described in our recent blog post, [dependency inference for Java and Scala](https://blog.pantsbuild.org/automatically-unlocking-concurrent-builds-and-fine-grained-caching-on-the-jvm-with-dependency-inference/) removes a ton of boilerplate.

But very fine-grained, always-accurate dependencies also enable per-file compilation, reducing the number of files that Pants needs to feed to `scalac`, and allowing for automatic file-level parallelization and the most accurate cache keys possible. From a correctness perspective, that means that unlike tools which use [compilation libraries like Zinc](https://github.com/sbt/zinc) (SBT, Bloop, Mill, optionally Bazel, and others) Pants `2.9.0` supports sandboxed, minimal incremental Java and Scala compilation, while preventing the under-compilation bugs that have historically troubled Scala developers.

For more information on how dependency inference works for the JVM, [check out that post](https://blog.pantsbuild.org/automatically-unlocking-concurrent-builds-and-fine-grained-caching-on-the-jvm-with-dependency-inference/)!

## Multiple resolves of third party dependencies

Another significant improvement for the JVM over 1.x is that Pants `2.9` implements a monorepo-friendly multiple-resolve/lockfile strategy for third party dependencies, allowing for correctness, flexibility, _and_ performance.
Build tools for the JVM tend to either resolve dependencies globally (for an entire repository), or locally (on a project-by-project basis). Global resolves (as in Bazel) remove flexibility, because teams working within a repository cannot diverge from the single blessed versions of any library: if they try, they are nearly guaranteed to encounter classpath incompatibilities. On the other hand, local/per-project resolves (as in SBT, Gradle, Maven) within a monorepo allow for local flexibility, but reduce the performance and compatibility of any particular build by executing one unique resolve per project.

Rather than forcing global or per-project resolves, Pants `2.9` supports a unique hybrid approach: third party resolves are named and first-class, and can be used on a target by target basis. This allows a monorepo to operate with the minimum number of resolves required to support their conflicting library versions, without necessarily going to the costly extreme of per-project resolves.

Having the minimum number of resolves improves performance, but it doesn't come with a correctness cost! To ensure reproducible builds, Pants generates a lockfile per resolve, which is then efficiently consumed to fetch the precise, fingerprinted dependencies of any particular file.

## Trying it out

Although this is an alpha release, the features that are implemented so far are expected to give teams enough to work with to validate using Pants with a JVM codebase:

- Scalatest, Junit
- scalafmt, google_java_format
- Scala Repl
- Protobuf code generation with ScalaPB
- Debugging support
- Scala compiler plugins
- Support for cycles between Java and Scala
- Multiple resolves with independent lockfiles
- Bootstraps a configured JVM using coursier

We'd love to help you try out Pants with your JVM codebase: you can start by checking out the [initial documentation](https://www.pantsbuild.org/v2.9/docs/jvm-overview) and [example JVM repository](https://github.com/pantsbuild/example-jvm). If you see anything missing that prevents you from evaluating Pants for Java and Scala, please let us know by [opening a Github issue](https://github.com/pantsbuild/pants/issues/new/choose), or [visiting Slack](https://www.pantsbuild.org/v2.9/docs/community)!

## Better visibility into runtime and cache hits for tests

Thanks to a great change from a new contributor, Pants `2.9` now renders test runtime and cache status (whether a process ran, hit a cache, or was memoized in memory by `pantsd`) in the test summary for all supported languages!

```
❯ ./pants test src/python/pants/util:
…
✓ src/python/pants/util/dirutil_test.py:tests succeeded in 1.21s (cached locally).
✓ src/python/pants/util/osutil_test.py:tests succeeded in 0.72s (memoized).
✓ src/python/pants/util/strutil_test.py:tests succeeded in 0.98s (cached remotely).
```

No more thinking to yourself: “Gee, that was even faster than usual! I wonder why?”

## Improvements to Docker support

Among a number of bug fixes and documentation enhancements, here are the noteworthy improvements to the Docker backend:

- Introduce a new `target_stage` field for `docker_image` as well as the `[docker].build_target_stage` option.
- Add `instructions` field to `docker_image` to support generating the Dockerfile.
- Logs warning about the docker build context, and what files where referenced and not, and possible renames to get more matches in case of docker build failure.
- New option for the `[docker]` scope, which allows passing additional options when executing `docker run [OPTIONS] <image>`, in addition to the `--run-args` which are passed to the image entrypoint.
- Add new `secrets` field to `docker_image`.
- Include `shell_source(s)` in `docker_image` build context.
- Support interpolating Docker build args into the `repository` field of `docker_image` targets.
- Allow tailor to create `docker_image` targets for any files with “Dockerfile” in the file name.

Check out the updated [`docker` documentation](https://www.pantsbuild.org/v2.9/docs/docker) for more information!

## Changes to Go project layouts

After [adding experimental support for Go](https://blog.pantsbuild.org/golang-support-pants-28/) in Pants `2.8`, we decided that a few changes to how Go targets are laid out in BUILD files would help to future proof the support.

To that end, in Pants `2.9`, each `go` package now needs its own `go_package` declaration in a `BUILD` file. Thanks to dependency inference and `./pants tailor`, these BUILD files are very simple, and rarely require adjustments. But when they do, they [follow the 1:1:1 principle](https://github.com/pantsbuild/pants/issues/13488): metadata about your code should live near the code itself.

In particular, the changes to target layouts make it easier to use the new Go features in `2.9`. `go_package` targets now support:

1. Setting a `test_timeout` for the package (in seconds), which is propagated down to the Go test runner:

```
go_package(
    test_timeout=120,
)
```

2. Embedding `resource` files in a binary for use at runtime:

```
go_package(name='pkg', dependencies=[":resources"])
resources(
    name="resources",
    sources=["*.txt"],
)
```

3. Adding `files` to the working directory of your tests (i.e. `testdata`):

```
go_package(dependencies=[":testdata", "//:root"])
file(
    name="testdata",
    source="testdata/f.txt"
)
```

To update your `BUILD` files for the new layout in `2.9`, run `./pants tailor`. Thanks for your patience, and all of your feedback on the new Go support! Please continue to let us know how it can be most useful to you.

## Thanks

Thanks to all the contributors to 2.9, including everyone who shared feedback on changes and who tested release candidates!

## Notes

- Tell the (hi)story of JVM support
  - How v1 served the JVM community
    - Discuss our relationship with the Scala community
  - Problems with our v1 JVM support, and how v2 fixes them
- Highlighted features
  - JVM Alpha
    - Scalatest, Junit
    - Scalafmt, google_java_format
    - Scala Repl
    - Debugging support
    - Scala compiler plugins
    - Support for cycles between Java and Scala
    - Multiple resolves with independent lockfiles
    - Bootstraps a JVM via coursier
  - Duration and cache source for tests
  - Docker improvements
    - https://github.com/pantsbuild/pants/pull/13997
    - https://github.com/pantsbuild/pants/pull/13953
    - https://github.com/pantsbuild/pants/pull/13818
    - https://github.com/pantsbuild/pants/pull/13830
    - https://github.com/pantsbuild/pants/pull/13761
    - https://github.com/pantsbuild/pants/pull/13721
    - https://github.com/pantsbuild/pants/pull/13601
    - https://github.com/pantsbuild/pants/pull/13386
  - Go changes
    - Move to 1:1:1:
      - https://github.com/pantsbuild/pants/pull/13702
      - https://github.com/pantsbuild/pants/pull/13681
      - https://github.com/pantsbuild/pants/pull/13707
    - https://github.com/pantsbuild/pants/pull/13781
    - https://github.com/pantsbuild/pants/pull/13743
  - Support for using `tailor` continuously in a repo
    - https://github.com/pantsbuild/pants/pull/13432
    - https://github.com/pantsbuild/pants/pull/13422
    - https://github.com/pantsbuild/pants/pull/13420
  - Performance improvements
    - https://github.com/pantsbuild/pants/pull/13370
    - https://github.com/pantsbuild/pants/pull/13559

        - Performance improvements for test runs via subsetting improvements
            - NB: Only the `hash` removal ended up being enabled by default, but probably still significant.
- Future work, call to action