Who says you can’t use Gradle for a Monorepo?
Who says you can’t use Gradle for a Monorepo?

Yes, we looked into Bazel, but...
Who says you can't use Gradle for a Monorepo?

Contents

WHO SAYS YOU CAN'T USE GRADLE FOR A MONOREPO?

Backend Development @ DoorDash
Problems we were facing
Exploring Bazel
Exploring Gradle
Augmentations
Takeaways
Who says you can't use Gradle for a Monorepo?

Building a monorepo would help, but it comes with challenges to the build system
Who says you can't use Gradle for a Monorepo?

Backend Development @ DoorDash

Tens of millions of lines of Kotlin code

~700 backend developers working on
~200 backend poly-repos
Who says you can’t use Gradle for a Monorepo?

Problems we were facing

- Build Pipelines & IDE Experience for large repositories can be slow!
- Separate codebases lead to utilities, build standards and pipelines performing a similar function
- Upgrading dependencies becomes a complex endeavour involving multiple organizations
- Propagating best practices & defragmenting tooling is a never-ending task
Exploring Bazel
Who says you can't use Gradle for a Monorepo?

Bazel

**PROS**
- Designed for large monorepos
- Distributed builds
- Remote execution
- Unify build systems across Web, Mobile, Backend

**CONS**
- All of Eng would need to migrate to Bazel
- Learning curve would slow developers down
- Kotlin development experience is subpar
- Lack of specific plugins that are available in Gradle
Who says you can't use Gradle for a Monorepo?

Custom Plugins

Plugins we’ve developed improve developer experience

- Reduce build boilerplate
- Optimized Maven Repository settings
- Apply internal code quality rules
- Standard application settings

Replicating this in Bazel would take work
Developer Empathy

Kotlin ❤ Gradle

Developers can’t pause all other development to migrate build systems.
Is Gradle still viable?
Who says you can’t use Gradle for a Monorepo?

Intro: Gradle Project Structures

**SINGLE PROJECT**
- build.gradle
- settings.gradle
- src

**MULTI-PROJECT**
- project1
  - build.gradle
  - src
- project2
  - build.gradle
  - src
- project3
  - build.gradle
  - src

**COMPOSITE**
- project1
  - build.gradle
  - settings.gradle
  - src
- project2
  - build.gradle
  - settings.gradle
  - src
- project3
  - build.gradle
  - settings.gradle
  - src

Degraded build latency and IDE experience past a certain number of projects
Who says you can't use Gradle for a Monorepo?

## Composite Builds

### PROS
- Familiarity and in-house expertise
- Robust 3P and internal plugin ecosystem
- Significantly better DevEx w/ Kotlin/Java¹
- Better IDE Experience

### CONS
- Distributed execution
  - limited to tests
  - requires Gradle Enterprise²
- Remote execution is not supported²
- Opaque project dependency graph²
- Doesn’t solve our problems for Web/Mobile repos

---

1. [bazelbuild/rules_kotlin](https://bazelbuild/rules_kotlin) lags behind equivalent in Gradle
2. [monorepo.tools](https://monorepo.tools)
Gradle Composite Builds

WITH A FEW KEY AUGMENTATIONS
How to do tasks correctly, fast and efficiently?

<table>
<thead>
<tr>
<th></th>
<th>In School</th>
<th>In Industry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctly</td>
<td>Copy answers from cheatsheet</td>
<td>Single source of truth</td>
</tr>
<tr>
<td>Fast</td>
<td>Do as little work as possible</td>
<td>Minimal set of changed targets</td>
</tr>
<tr>
<td>Efficiently</td>
<td>Ask classmates to do your homework</td>
<td>Distributed execution</td>
</tr>
<tr>
<td></td>
<td>Detention</td>
<td>Promotion</td>
</tr>
</tbody>
</table>
Who says you can't use Gradle for a Monorepo?

Design Principles

- **Correct**: Define dependency versions centrally for easier upgrades
- **Fast**: Build only the projects that are affected by a changeset (directly or transitively)
- **Efficient**: Distribute builds across multiple machines for maximum parallelism
- Empathy for developers
Who says you can't use Gradle for a Monorepo?

Version Catalogs

Monorepo structure

```
libraries
├── lib1
│   ├── lib2
│   └── lib3
│       └── lib4
├── lib2
│   ├── ser1
│   └── ser2
└── ser3
```

```
[versions]
kotlin = "1.8.22"

[libraries]
kotlin-stdlib = { module = "org.jetbrains.kotlin:kotlin-stdlib", version.ref = "kotlin" }
```
Affected Targets Computation

Reversed dependencies graph

What if we make a change on lib3?

```
[[lib3],[lib2, ser2],[ser1]]
```
Who says you can't use Gradle for a Monorepo?

Distributed Builds

Each project can be built and tested independently
Choice of worker containers

- **Ephemeral**
  - Provision at each incoming build request
  - Single build per container
- **Long-running**
  - No boot-up time
  - Isolated gradle daemons per pod for parallel building
  - Local build cache
Takeaway(s)
Who says you can't use Gradle for a Monorepo?

Results

- 🏆 Composite Builds provide a great IDE experience
- 🐌 Cross-cutting changes can run a full CI pipeline quickly
- 🚀 Significant reduction of service boilerplate
- 🚀 Boosted code-sharing and reuse
Who says you can't use Gradle for a Monorepo?

What’s next?

- Explore the remote parallel execution model for other parts of the stack
- Continue onboarding projects into our monorepo
Who says you can't use Gradle for a Monorepo?

Acknowledgements

We would like to specifically thank Jamil Seaidoun, Yazan Mimi, Nico Semko, Claire Han, Francis Tian, Mingmei Zhang, Matthew Anger, Michael Sitter, Aaron Livingstone, Nick Firmani, Haitham Gabr, Arjun Shantharam, Mathieu Courtemanche and Adam Rogal
Thank you