

dog food

in the Gradle Team

About Me

- Szczepan Faber
- lives in Krakow/Poland
- supports TS Podbeskidzie (sadly)
- lead at Mockito
- core dev at Gradle

Gradle

What do you know about Gradle?

- very compelling feature set
- very active development
- always open source and free

About this talk

- shows how the Gradle Team dogfoods the automation
- inspires
- impresses (hopefully ;)

Inside Gradle

- ~ 6 very distributed full time engineers
- Frequent, regular releases (~ 6 weeks)
- ~500k LOC (Java, Groovy, includes all tests & infrastructure code)
- ~2k test classes
- ~40 modules
- Contains documentation and release notes

Automation

Critical for high performing teams.

- Allows us to achieve high quality
- Continuous investment
- Persistent problem
- Everyone's responsibility

Development

Automating the day-to-day

IDE/Editor

- Automate the “import” process
- Prerequisites are java & IDE but no Gradle (thanks to Wrapper)
- Provide a consistent setup
- Make it possible to run frequent tasks via the IDE

Testing

- Unit
 - we are fond of tests, TDD and high coverage
 - groovy tests (spock) for java code

Integration testing

- every integration test can be ran in different modes:
 - embedded (speed, fast dev cycles, easy debugging)
 - forking (slower but more realistic)
 - daemon (excellent stress test and coverage test for the daemon)
 - parallel (excellent stress test and coverage test for parallel feature)

Integration testing

- cross version integration tests (backward compatibility)
 - run test against a set of different versions of a 3rd party tool (e.g. groovy compilation)
 - run test against a set of Gradle versions
 - full suite VS recent version only
- tooling api cross version tests (backward and forward compatibility)
 - consumer & provider

Testing

- Distribution
- Performance

Testable documentation

- Automatically tested documentation
 - samples (in user guide or in distribution)
 - evaluation
 - running arbitrary tasks and comparison of output
 - using samples content for integration tests
 - build script snippets in the javadoc - evaluation (dsl reference, javadocs)
 - java code snippets in the javadoc - compilation attempt (javadocs)

Documentation

- Automatically deployed documentation
 - User guide
 - DSL reference
 - API reference
 - Release notes

Code Quality

- Ensure public API is documented
- License headers
- Cyclical package dependencies
- Static analysis (Checkstyle & Codenarc)

Build pipeline

- Runs on every push
- Identical on both branches (master & release)
- TeamCity

Pipeline structure

- Static analysis
- Quick, less accurate tests (small number of platforms)
- Build production like distributions
- Platform tests
- Execution mode tests
- Performance test
- Promote

Reusing binaries

- Production like binaries built at start of pipeline
- CI server pushes binaries to downstream builds

Release cycle

A cycle is ~ 6 weeks.

- Move to release branch after ~ 4 weeks
- Plan new release ~ 4 weeks
- Promote RC ~ 5 weeks
- Promote final (end)
- Merge release branch back to master (final)

Promotion

- Separate “build” for promotion
- Automatically checks out specific revision
- Programatically rebuilds Gradle
- Imposes the version number
- Triggered by 1 click @ CI server

Delivery

- Build, smoke test
- Decorate docs with Google Analytics JS
- Upload to `repo.gradle.org` (some bits)
- Upload dists and decorated docs to Amazon S3
- Checkout website repo, update data, push
- Trigger pull new docs
- Smoke test delivered distributions
- Send notification email to team
- Finish with manual processes

Gradle.org

- Data driven
- Push deploy
- Updated by non technical staff
- Post “release” test

Defect tracking/planning

Three levels:

- forums.gradle.org -> entry point
- issues.gradle.org -> acknowledged defects
- Pivotal Labs -> team planning

Automated move/sync data between systems.

Forums

Some interesting automations/tweaks:

- Markdown/syntax highlighting
- Issue number linking
- Documentation linking
- News
- Roadmap

Thanks!

- gradle.org
- gradleware.com
- Questions?