Dependency Management for Continuous Release

Chuck Karish
chuck.karish@gmail.com

Continuous release

1. Automate everything.
2. Tests
    a. run early, run often
    b. enough coverage to ensure release success
3. Release frequently, a few changes at a time
4. Be prepared to roll back.
5. Allow for multiple versions live at once.

Continuous release for a startup?

- Project is fast when code base is small
- As the project matures all tasks take more time, more structure is needed
- Slowness makes developers less productive, less confident
  - Speedup makes code better

Testing: Fast and complete

- Shorter developer work cycle
- Prompt feedback: developer confidence
- Predict success of release
- Run all needed tests for every change

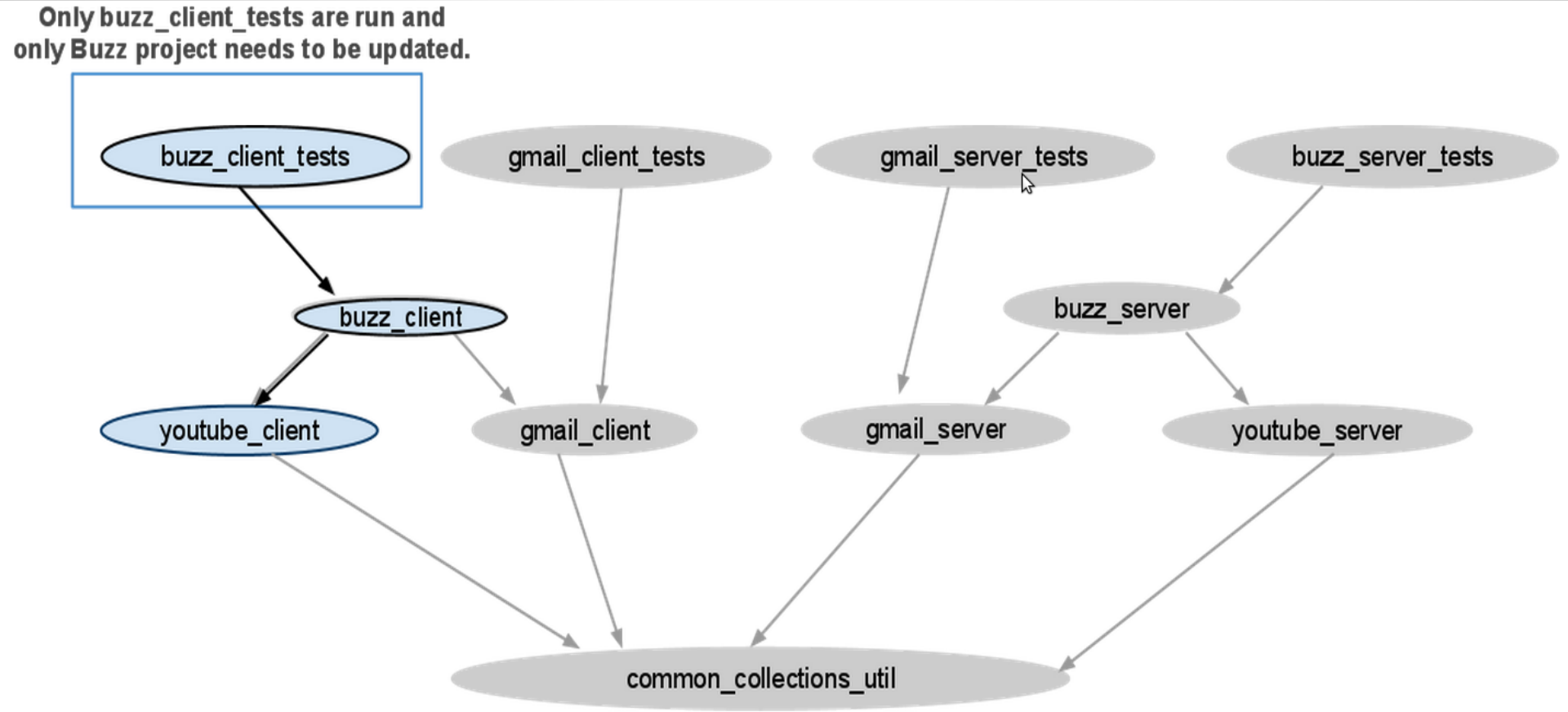
Big company dev workflow (Google)

http://google-engtools.blogspot.com/

- Central source control, access through FS
- Aggressive caching of build artifacts, reliable incremental build
- Dependency server chooses tests to run
- Release and deployment framework holds configs, customizations, results

Why is it fast?

- Build tool works like an IDE, has metadata and list of modified files in memory
- Test framework knows reverse dependencies for each file, runs only relevant tests
- Brute force: build and test caches use lots of compute machines, lots of storage, lots of net bandwidth



(taken from google-engtools.blogspot.com)

What about the little guys?

- git: devs may not see each other's work right away
- Conventional build tools
- Run all tests to be safe
- Open source frameworks for CI, release, cloud system management
- Maven and Jenkins: deps at package granularity

Easy fixes

- Frequent merges to a shared repository, so the CI server stays up to date
- Incremental compiles
- Testing is still slow

Why dependencies matter

- Minimize side effects of changes
- Code is easier to understand
- Unit tests can be more focused
- Fewer tests need to be run for each change

An open-source dependency server?

Why?

- To calculate reverse dependencies so only the tests that depend on a modified file need to be run.
- Potential for substantial speedup

An open-source dependency server?

How?

- Put all deps in build config files
- Extract reverse dependencies from test targets
  - Specified dependencies must be precise
- Incremental updates
- On-the-fly updates for presubmit
- Deliver through a Jenkins plugin?

An open-source dependency server?

Extra credit: work at file granularity (it’s messy)

- Java: instrumented classloader for tests
- C, C++: mine debug metadata
- Static analysis using modified compilers