Who Moved My Module?

About Me

- Yoav Landman
  - JFrog's CTO and Founder
  - Creator of the Artifactory Project
  - 10+ years experience in commercial enterprise build and development environments

Agenda

- Modular software development
- Why manage modules
- Using modules in release management
  - Release strategies and models
  - Common release issues

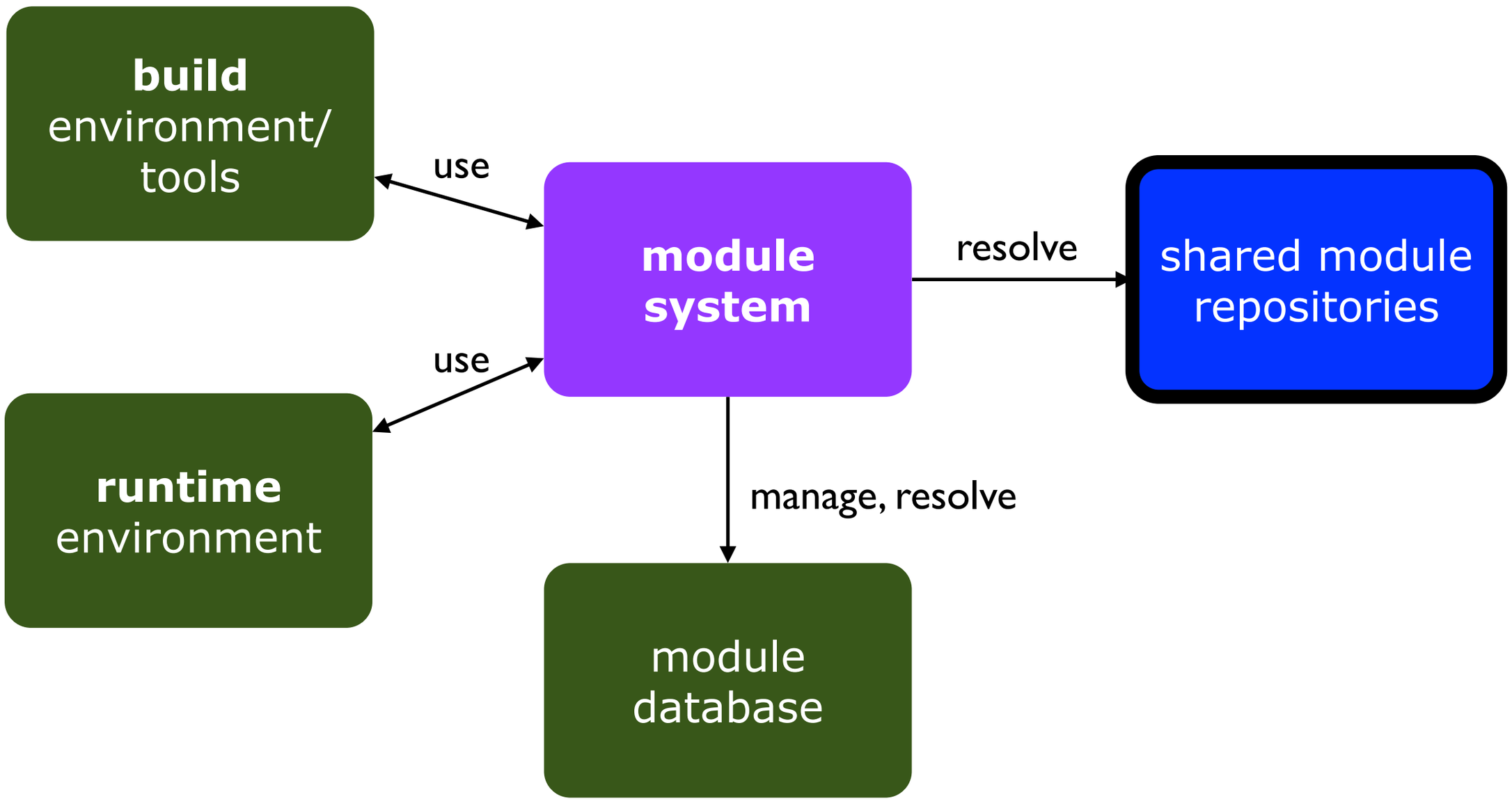
Going modular

http://www.flickr.com/photos/jemsweb/4363548805

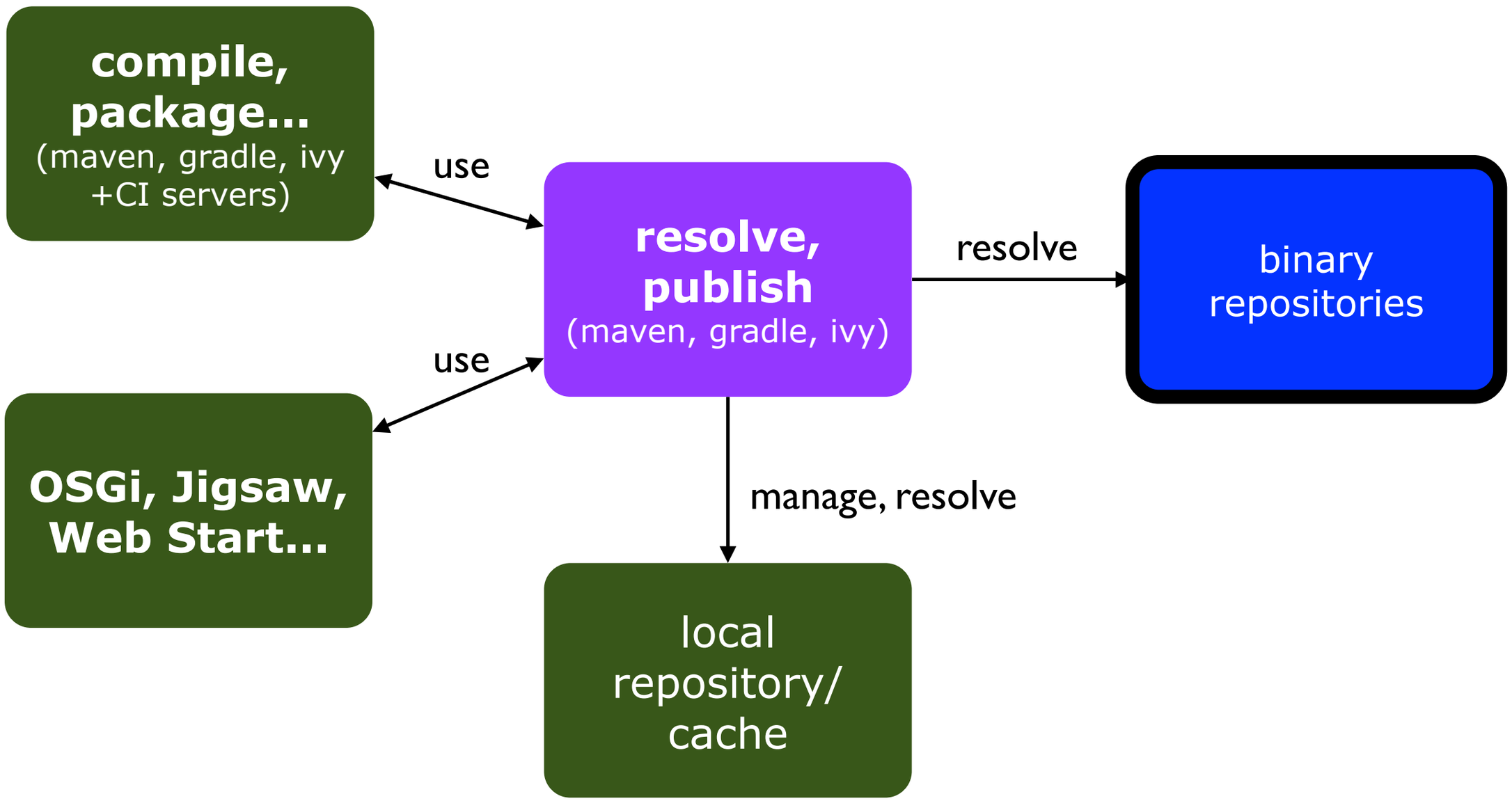
Java modules

- Java IDEs - project modules introduced circa 2004/2005
- Maven, Gradle, Ivy, Jigsaw, OSGi...
- CI Servers
  - Cascading builds according to dependencies
  - Per module results
- A reality

Module key role players



key role players - real world



Demo Time

Meet Artifactory

A Binary Repository

- A shared place for binaries
  - 3rd party and local artifacts
- Much more than a passive storage
- Critical for CI and ALM

What's wrong with my...

- Subversion
- Apache file server/webdav
- Shared file system
- DVD
- Disk-on-key, tape, drawer...

Can your shared file system/VCS

- Download and cache remote artifacts?
- Promote artifacts during release?
- Control licenses used in your software?
- Track and preserve artifacts used by CI builds?
- Enforce module CRUD security easily?
- Track changes made to artifacts?
- Automatically clean up integration garbage?
- Manage artifacts with powerful searches?
- Expose powerful REST API?

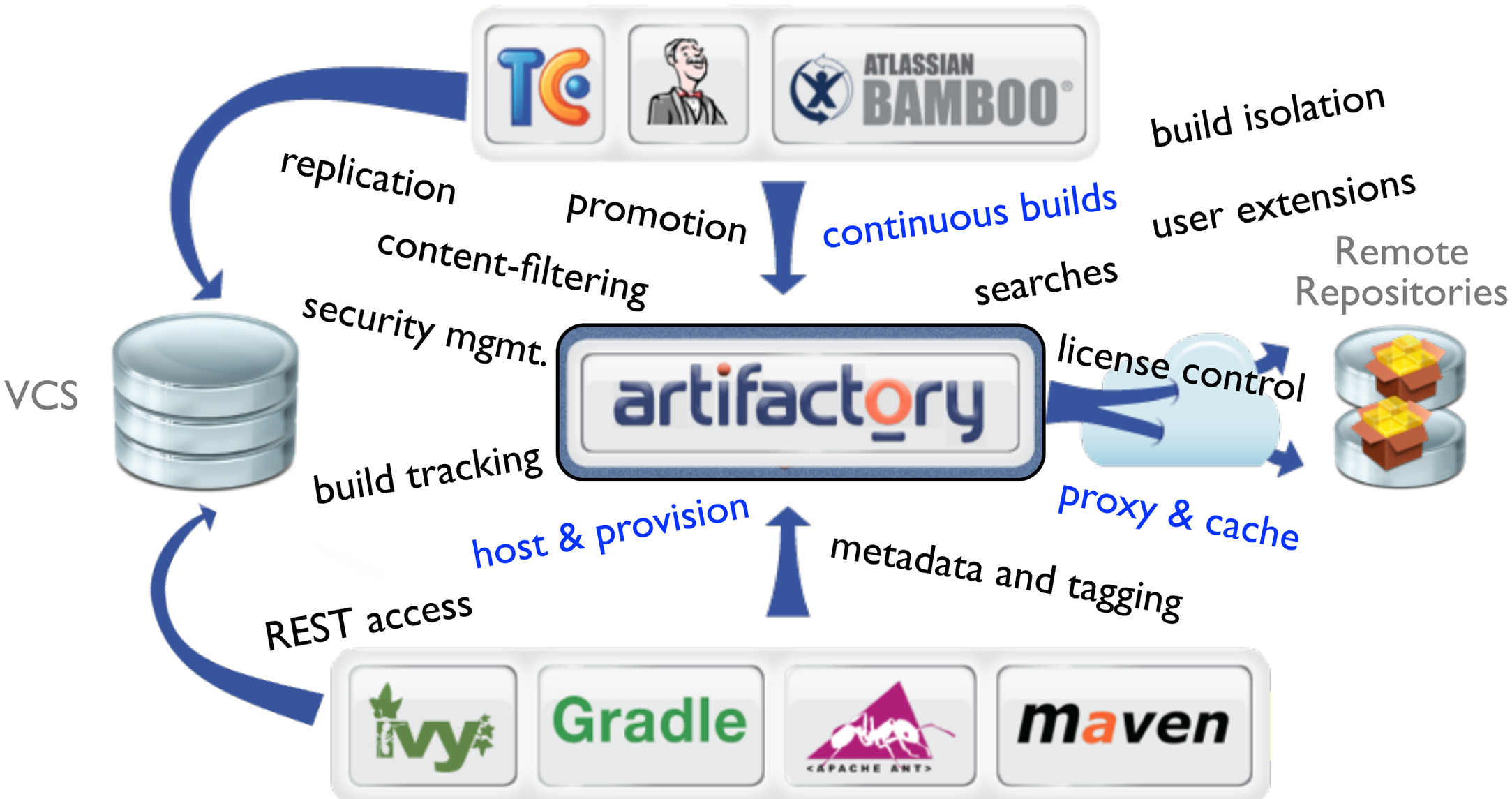
But I can always rebuild my modules from source!

http://www.flickr.com/photos/a-culinary-photo-journal/3134396770

Rebuilding from VCS

- Expensive & time consuming
  - Sometimes non-practical (resources)
  - Disaster recovery time
- Dynamic
  - Properties, version ranges, etc.
  - Reliance on a specific environment

The Artifact Repository Ecosystem



Artifactory

- Advanced Binary Repository Manager
- First searchable, web-driven repository manager (2006)
- Over120,000 downloads (Feb 2011)
- OSS, Pro & Cloud versions
  - jfrog.org | jfrog.com | artifactoryonline.com
- Shaping the binary repository arena

Never look back!

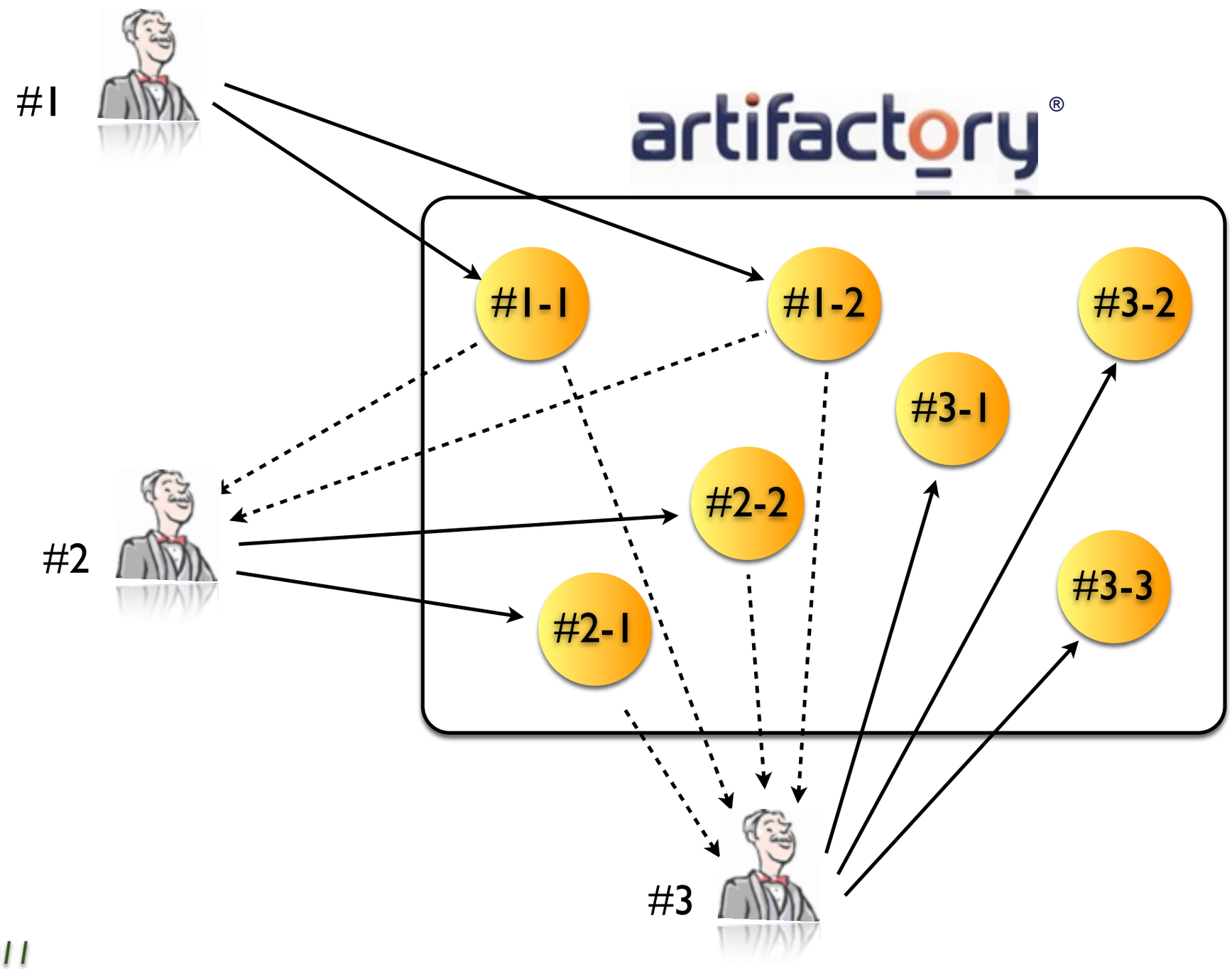
Demo Time

on-boarding & searching

Artifact Build Integration Platform

http://www.flickr.com/photos/skrb/1326663872

Artifact Build Integration Platform



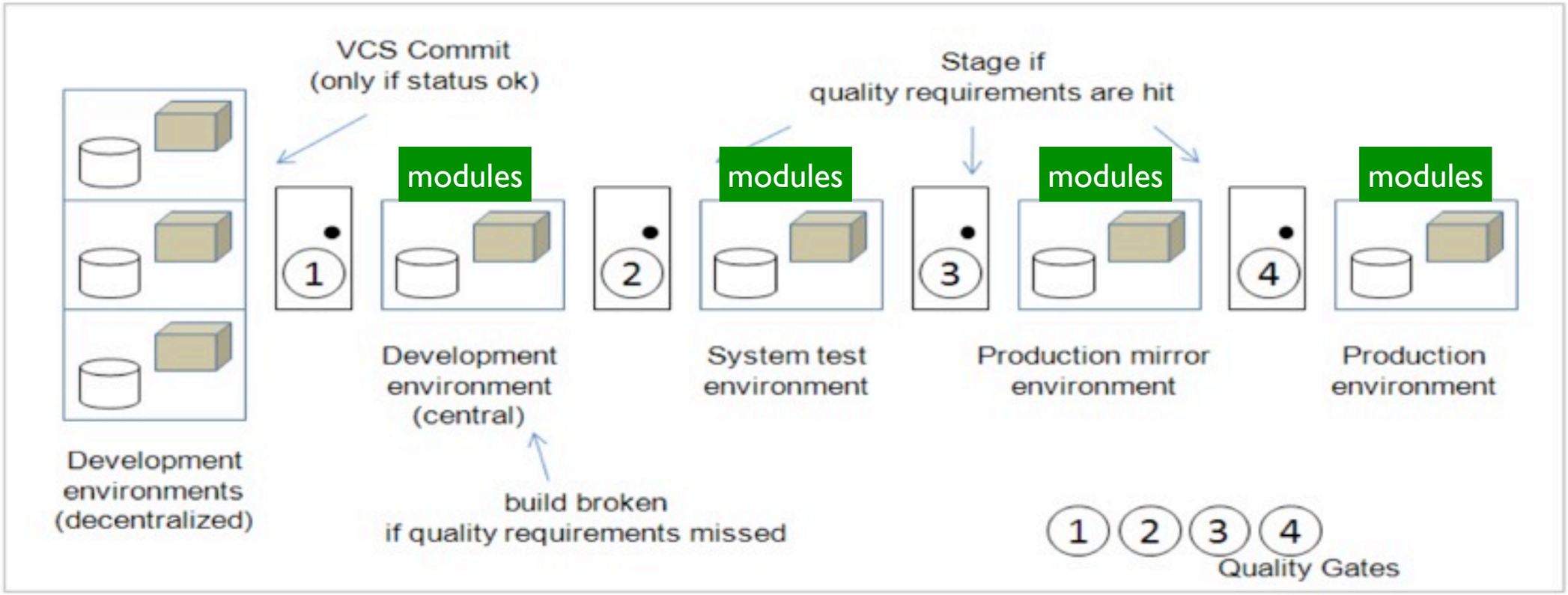
Across-the-board integration

Demo Time

Build integration

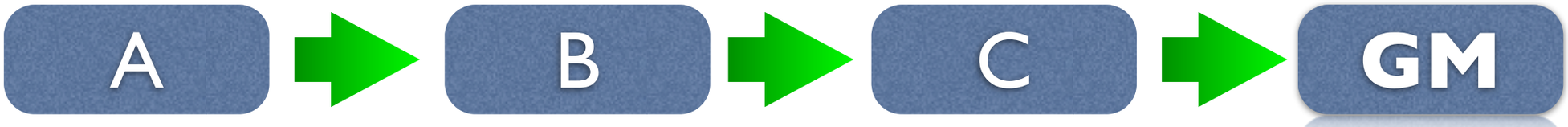
The release pipeline

- Multiple steps towards declaring a release
- Check points for advancing the release flow



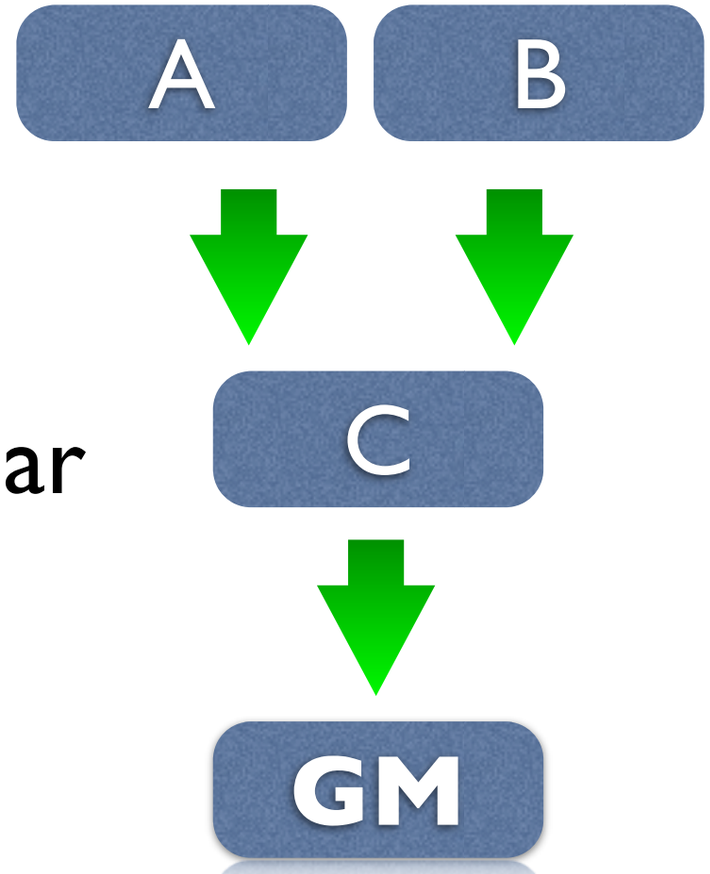
Source: Agile ALM, Michael Hüttermann, Manning Publications Co.

The release pipeline model



Linear

Non-linear



The release pipeline - YMMV

http://www.flickr.com/photos/jaxxon/3335409285

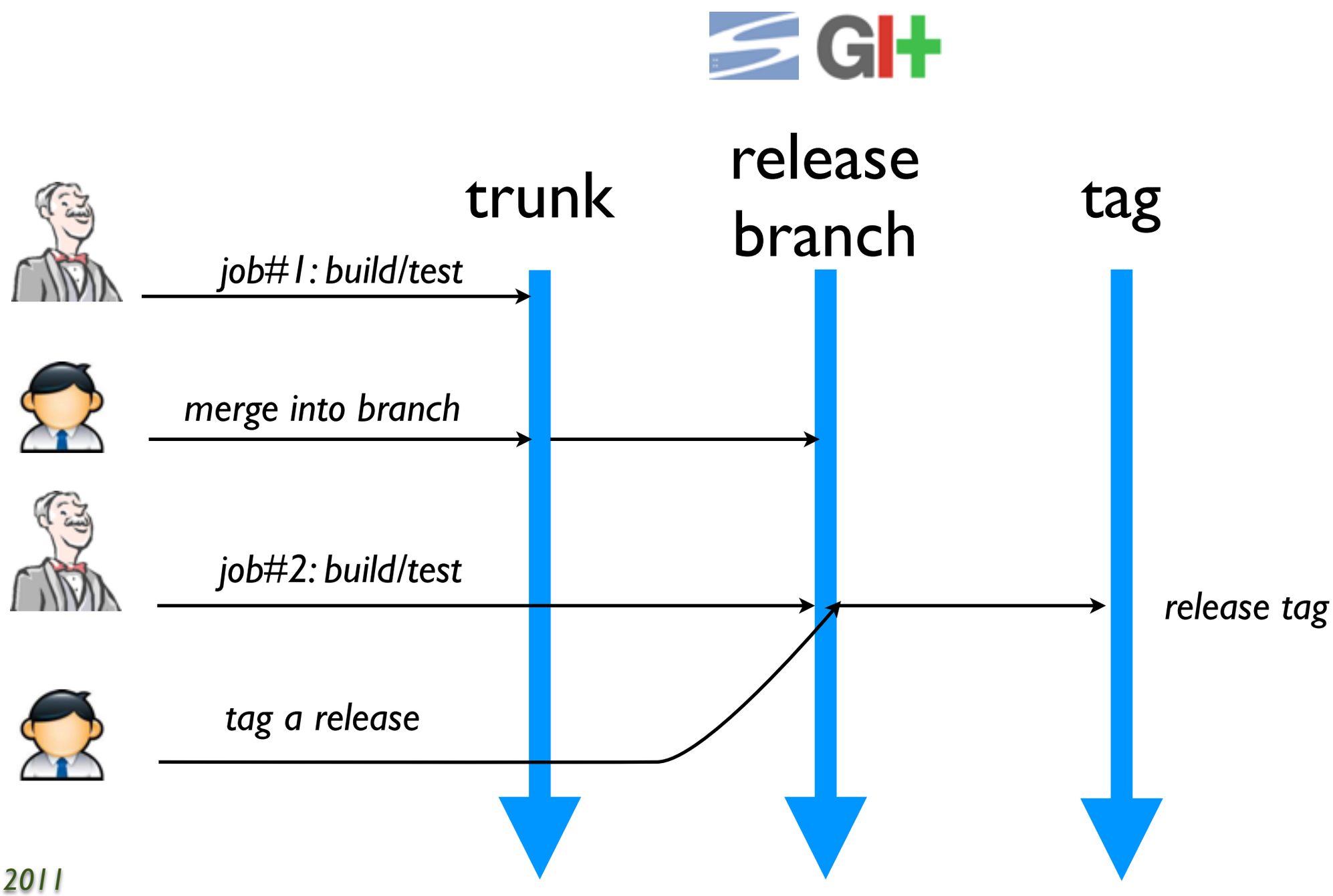
Typical release issues

- Picking a source/release strategy
- Release build Isolation
- Change release status without recompilation
  - Dynamic module descriptors
- Avoiding license “creep”

Source/release strategies

- Release-to-branch
- M2 Release plugin
- Artifactory build-integration release

Merge-to-branch



Jenkins M2 release plugin

1. Check out latest VCS revision
2. Compile & test
3. Change POMs to next release version
4. Compile & test
5. Commit new POMs (!)
6. Create a VCS tag from WC
7. Change POMs to next dev version (++-SNAPSHOT)
8. Commit new POMs
9. Check out previously created VCS tag
10. Compile & test
11. Publish binaries to repository

http://www.flickr.com/photos/rsdio/3642425935

Jenkins M2 release plugin

- Attempts to reuse a user-operated plugin in a CI environment
  - Uses separate external Maven process
  - Uses separate external VCS (svn auth)
- No good rollback
  - End users may like CI's VCS permissions

Artifactory plugin

1. Change POMs to next release version
2. Compile & test
3. Publish binaries to repository
4. Create a VCS tag from WC
5. Change POMs to next dev version (version++-SNAPSHOT)
6. Commit new POMs

Releasing with the Artifactory plugin

- Fast
- Easy automatic rollback
  - E.g. tag removal
- Closely integrated with CI server
  - Reuses Maven and VCS definitions
- Support for Ivy and Gradle + other CI servers in the works

Demo Time

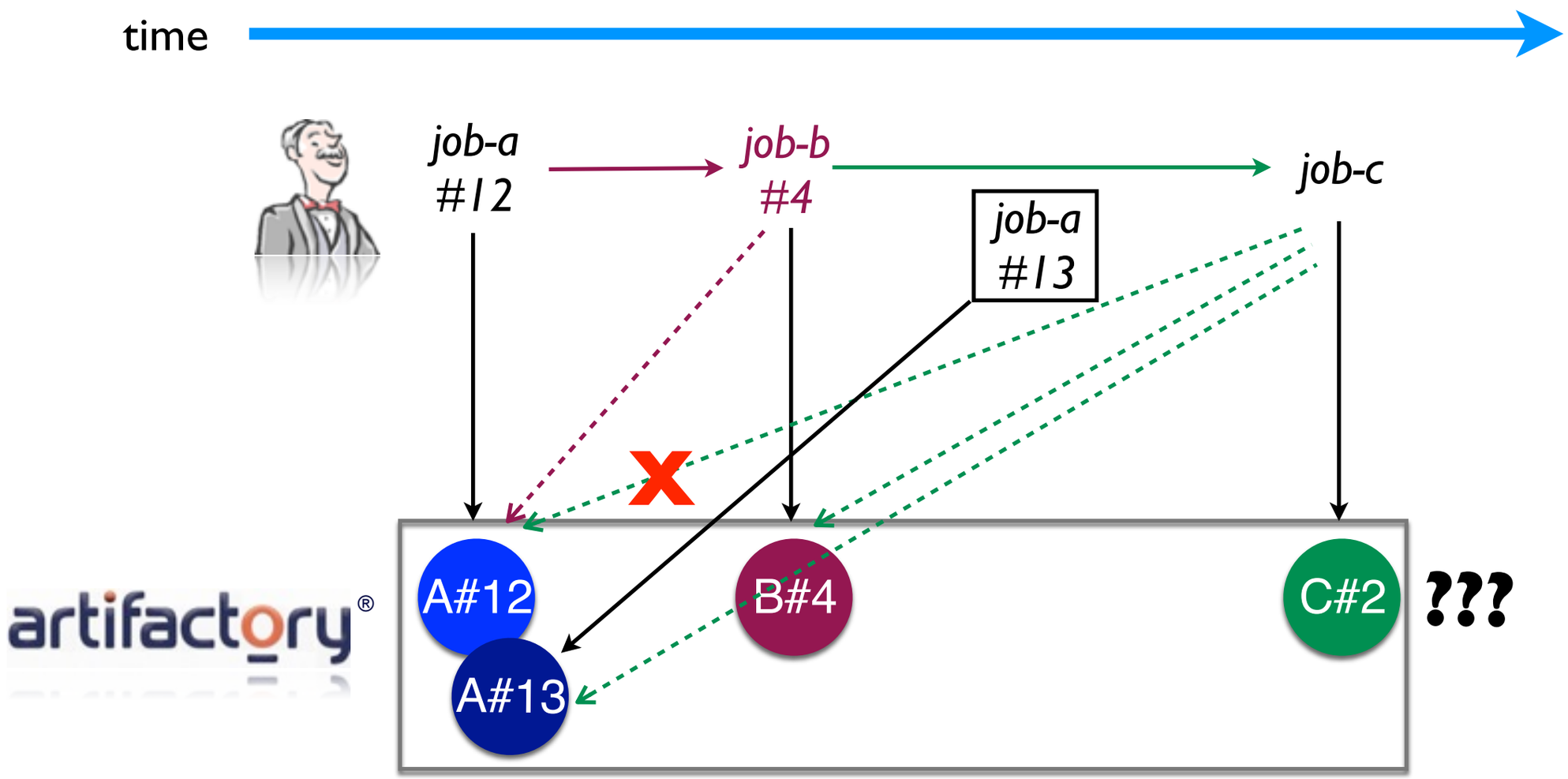
Release & promotion

Release build isolation

- Cascading jobs - tests run longer
- A downstream job may take wrong artifacts put by an upstream job
- No isolation - the build integration silo is broken

http://www.flickr.com/photos/sharynmorrow/5961475

Keeping integration private



Isolating resolution

- Deploy as many integration revs in parallel
- Each build resolves only the artifacts in the build chain it is part of
- Achieved with CI artifact painting and matrix params for resolution:

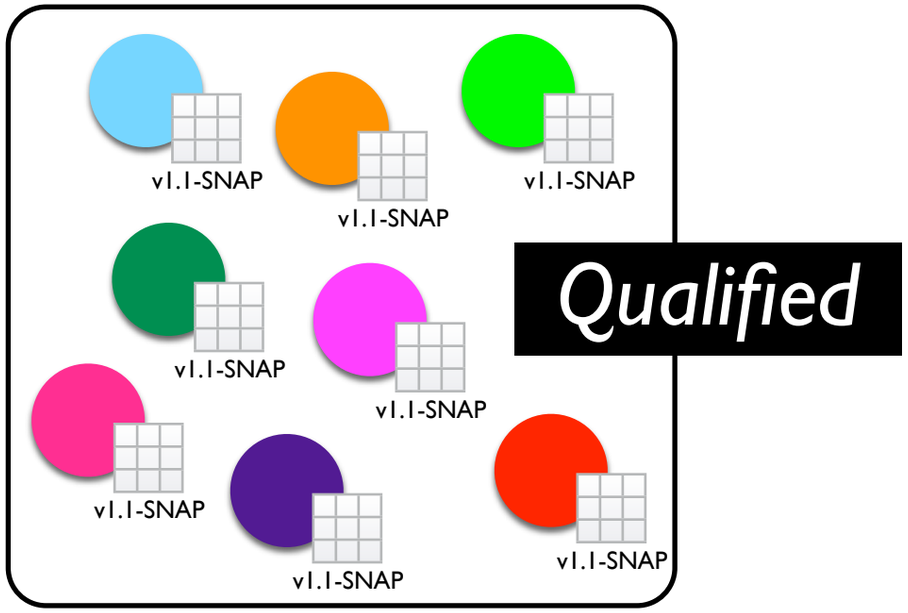
```
http://.../artifactory/jfrog/app-1.0-SNAPSHOT.jar;build.root=j-1025
```

Demo Time

Release isolation/private builds

Avoiding recompilation on release

Product Binaries



- A collection of release-qualified integration packages
- Lack of release status
  - No way to express that without changing pom/ivy descriptors
- Resolution for packaging relies on descriptors

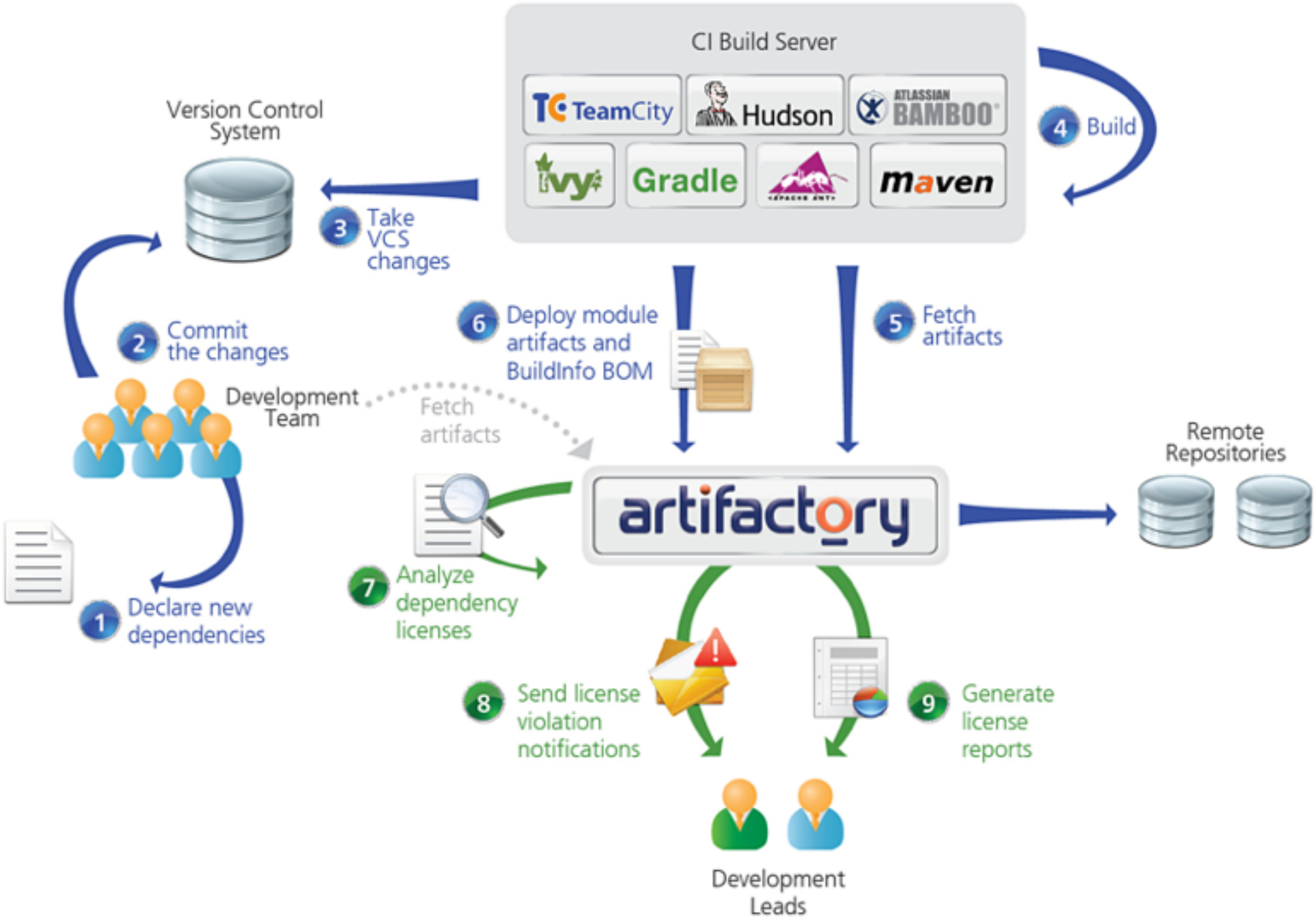
Resolution with integration versions

- Ivy/Gradle
  - Integration revisions same as any release revision (1.0-782)
  - By default all dynamic versions are resolved and replaced in delivered modules
  - Can be used in a release package
- Maven
  - Published artifacts can contain dynamic dependencies (SNAPSHOT or ranges) + unresolved properties
  - Very hard to keep reproducibility

License “creep”

- Verify no unwanted 3rd party is packaged into the release
- Discover and apply license information continuously!
  - Information is there
  - Discover and act early

License violation detection



Demo Time

Detecting license violations

Summary

- Why do we need a binary repository
- Smart modules hub that plays key roles in
  - Continuous integration and delivery
  - Release management
- Really changes how we work and think about binaries
- Constantly evolving

Thank you! Questions?