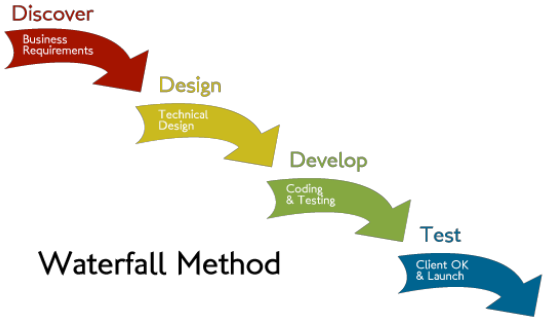
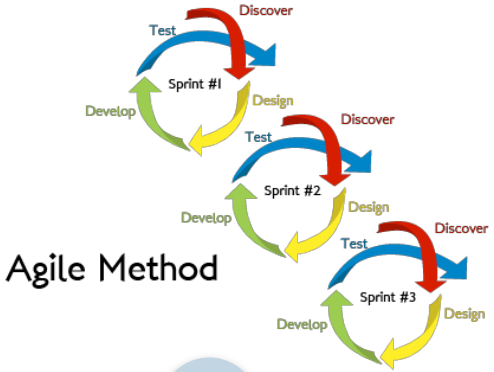
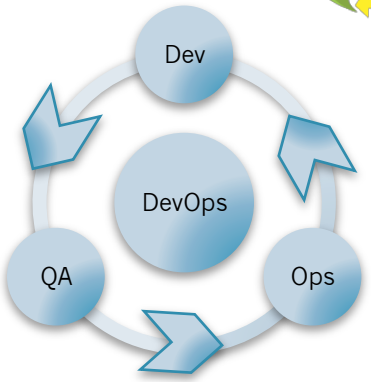
Practicing Continuous Delivery using Hudson

Winston Prakash
Oracle Corporation

Development Lifecycle



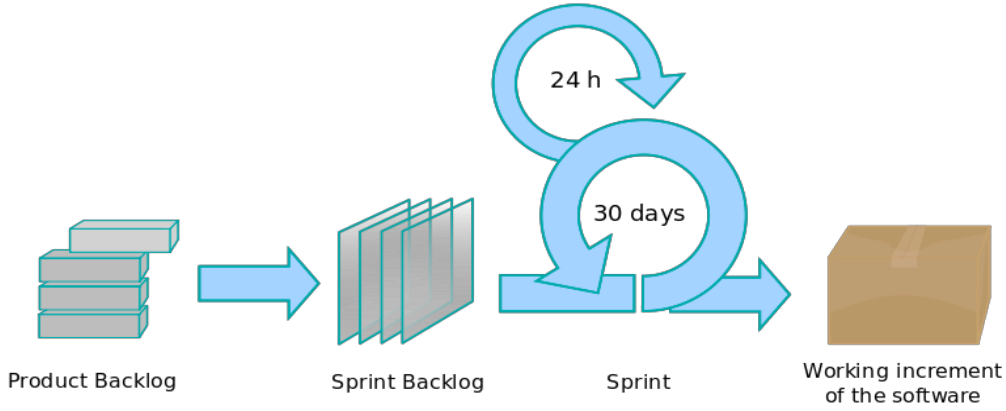


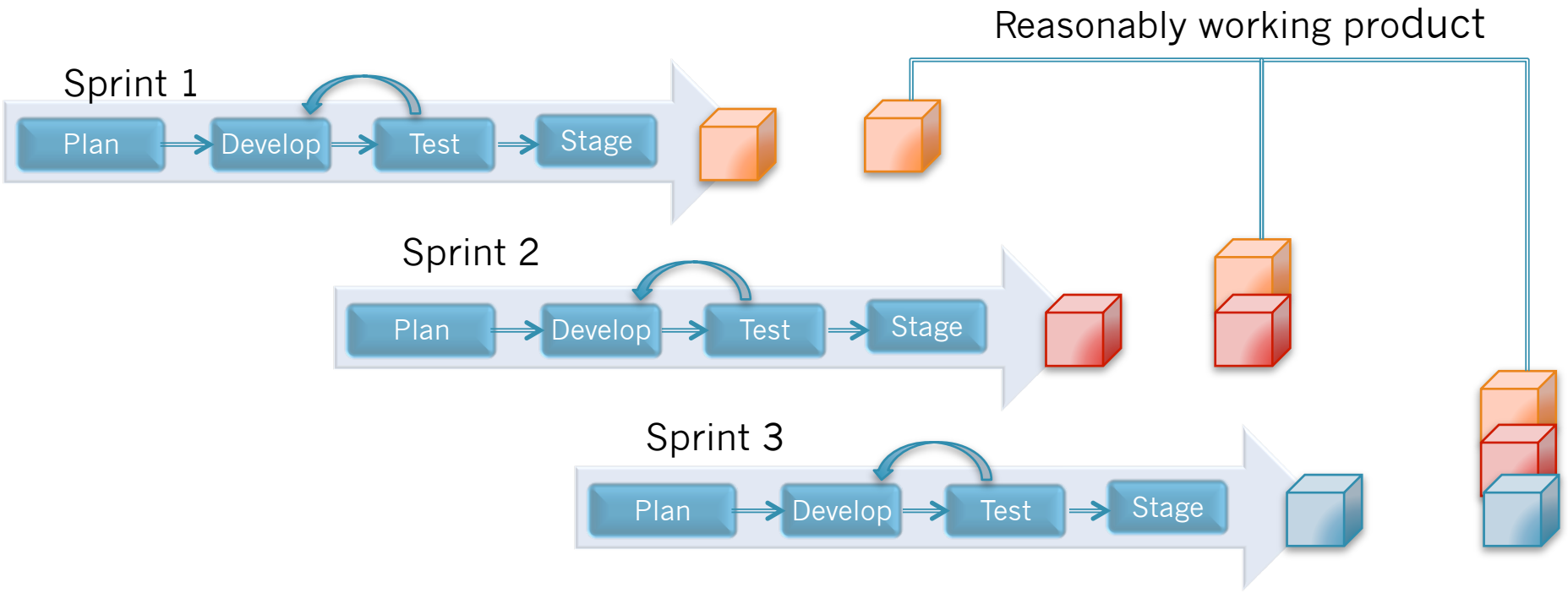


Typical turn around time is
6 months to 1 year

Sprint cycle is typically
2 weeks to 4 weeks

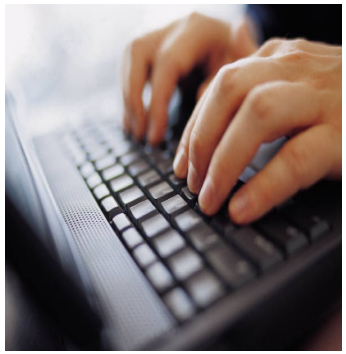
Typical Sprint Cycle





Further Thought Process

- There are some greedy people out there
- They can't wait until the end of the sprint cycle to get a working product
- They want a working product on every commit



Commit

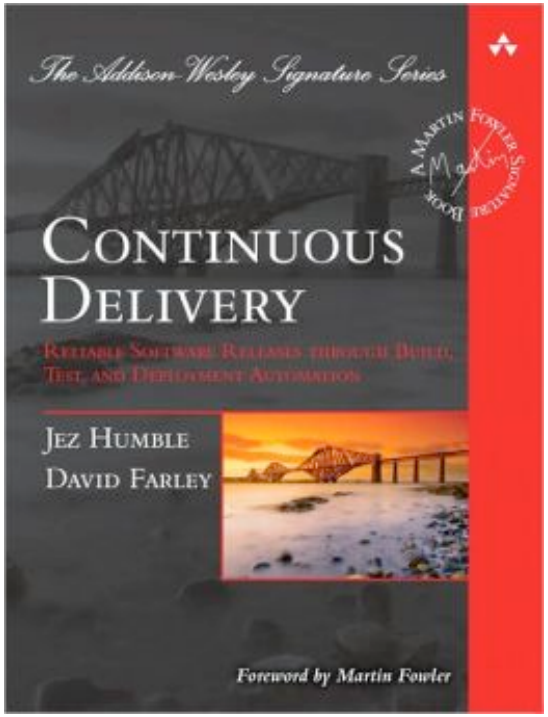
Build Magic

Production Ready

Thus the Continuous Delivery concept was born

What is Continuous Delivery?

- A set of practices and principles aimed at building, testing and releasing software faster and frequently.
- Produce a deployable-to-production build regularly, probably on each commit.
- Every build is a potential release.

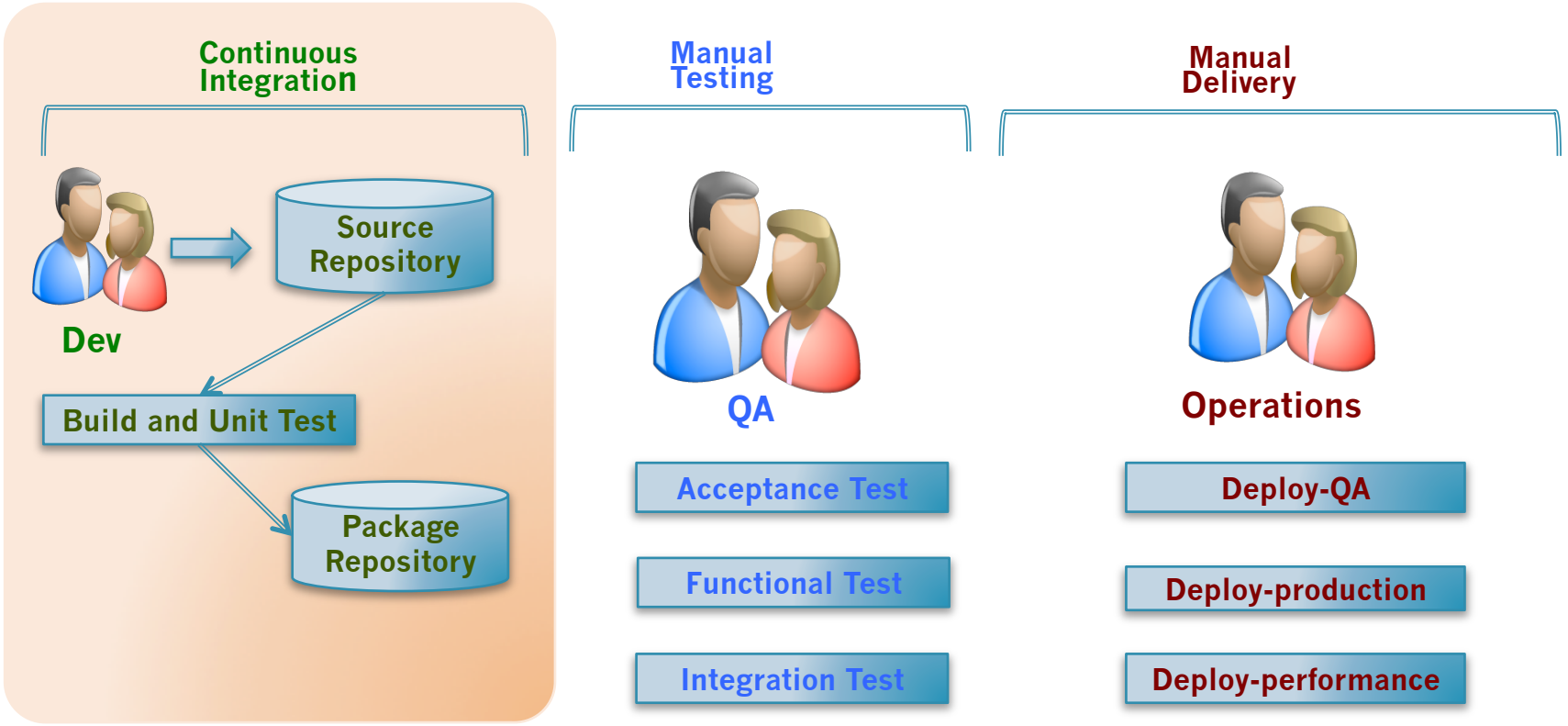


Commit to deploy

In a Test Driven Development build pipeline, Continuous Integration is the first step and the end result is the Continuous Delivery.

While Continuous Delivery promotes the concept of keeping your product in a deliverable state on each commit, Continuous Deployment takes it further. On each commit, the deliverable can be deployed to a production environment.

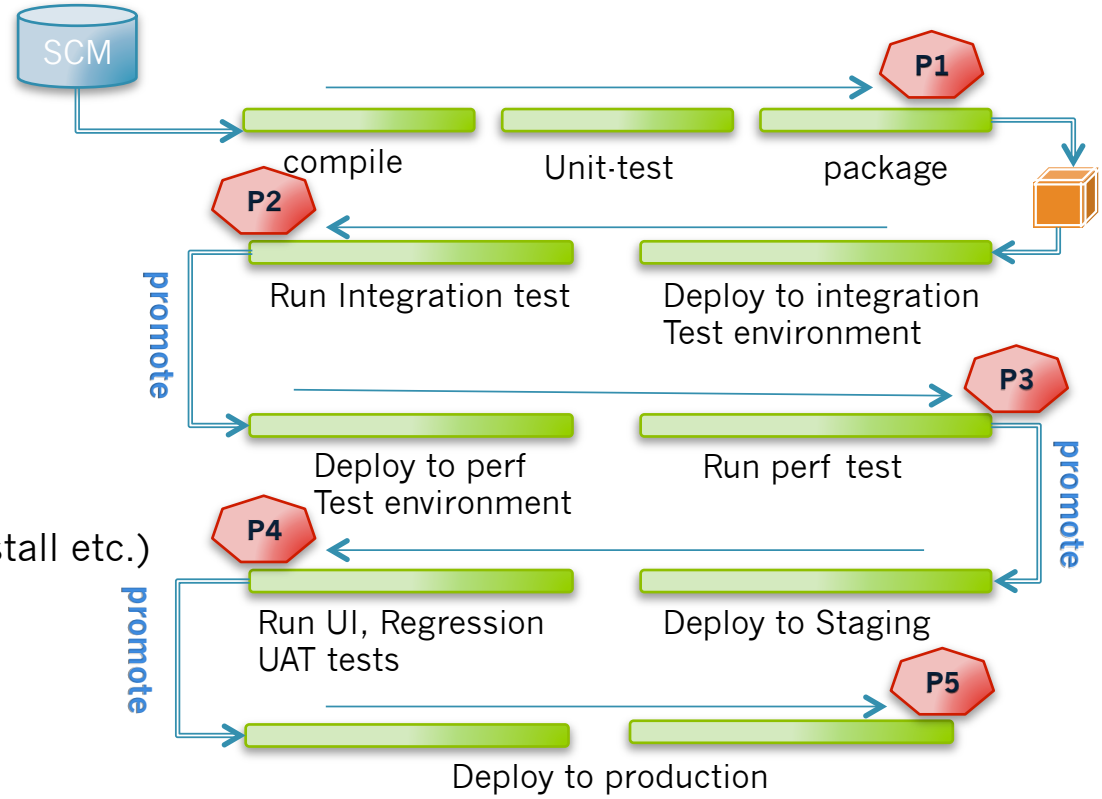
Typical Hudson CI Server Usage



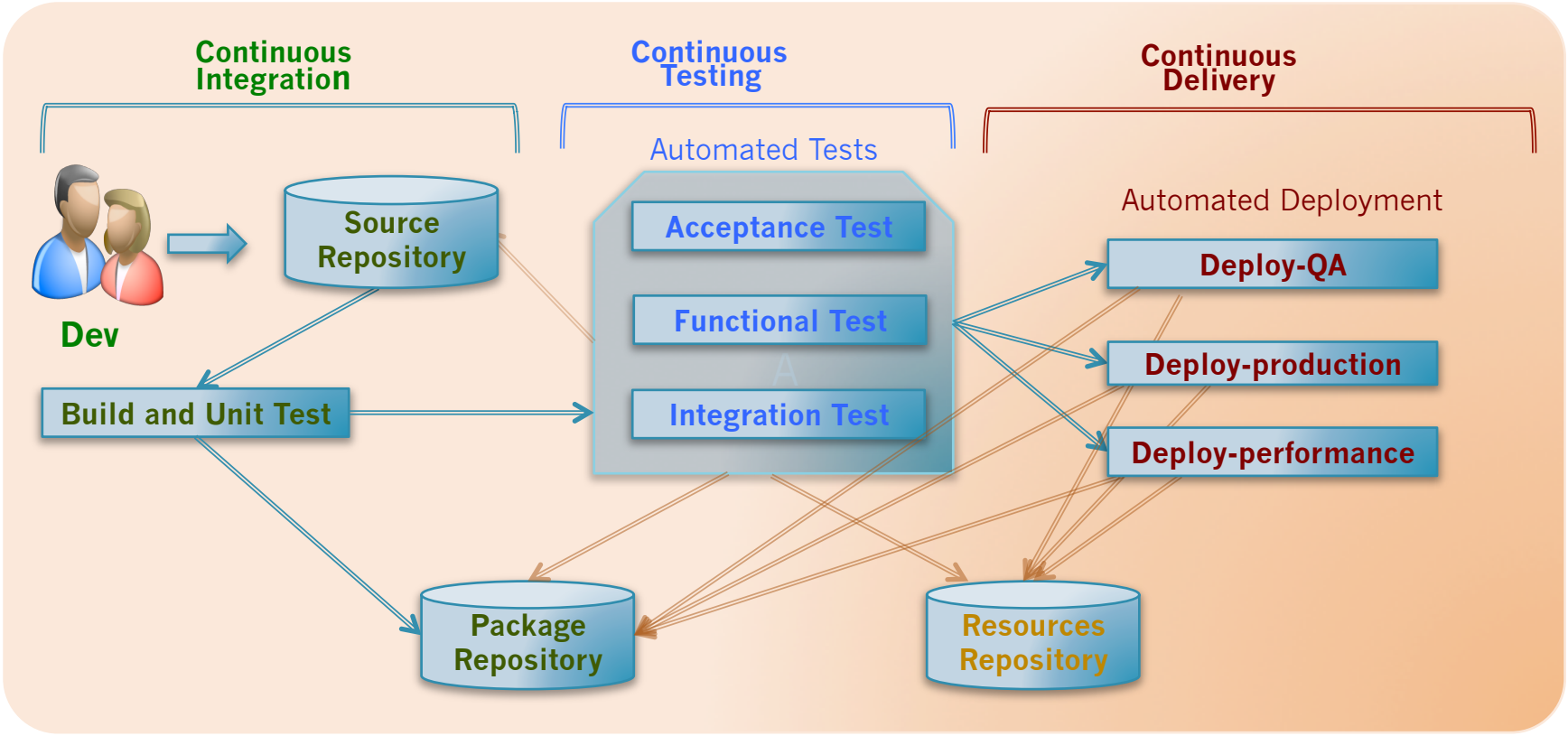
Hudson is mostly tuned to focus on development teams

Build Pipeline

- Commit
- Build and Test
  - Unit tests
  - Static code coverage
  - Packaging
  - Integration tests
  - UI tests
  - Performance tests
  - Regression tests
  - Deployment tests (install, uninstall etc.)
  - Manual exploratory tests
  - Regulatory, compliance checks
  - Clearance from UAT
- Stage and Deploy



Setting up Hudson to do Continuous Delivery

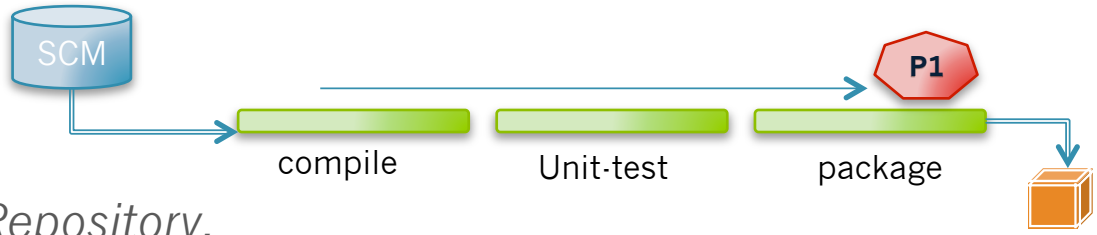


Identifying the relevant plugins and configuring the jobs to participate in the pipeline is critical.

Setting up CI Environment

- A Centralized SCM repositories (Git, SVN, CVS etc)
- Dedicated build servers
- Continuous Integration software (Hudson)
- Unit testing framework (jUnit, nUnit etc)
- Build tool (Maven, Gradle, Ivy, Ant etc)
- Deployment environment (Application Servers)
- Build Dashboard (Hudson UI)
- Communication tool (E-mail, twitter, IRC etc)
- Deployment Tool

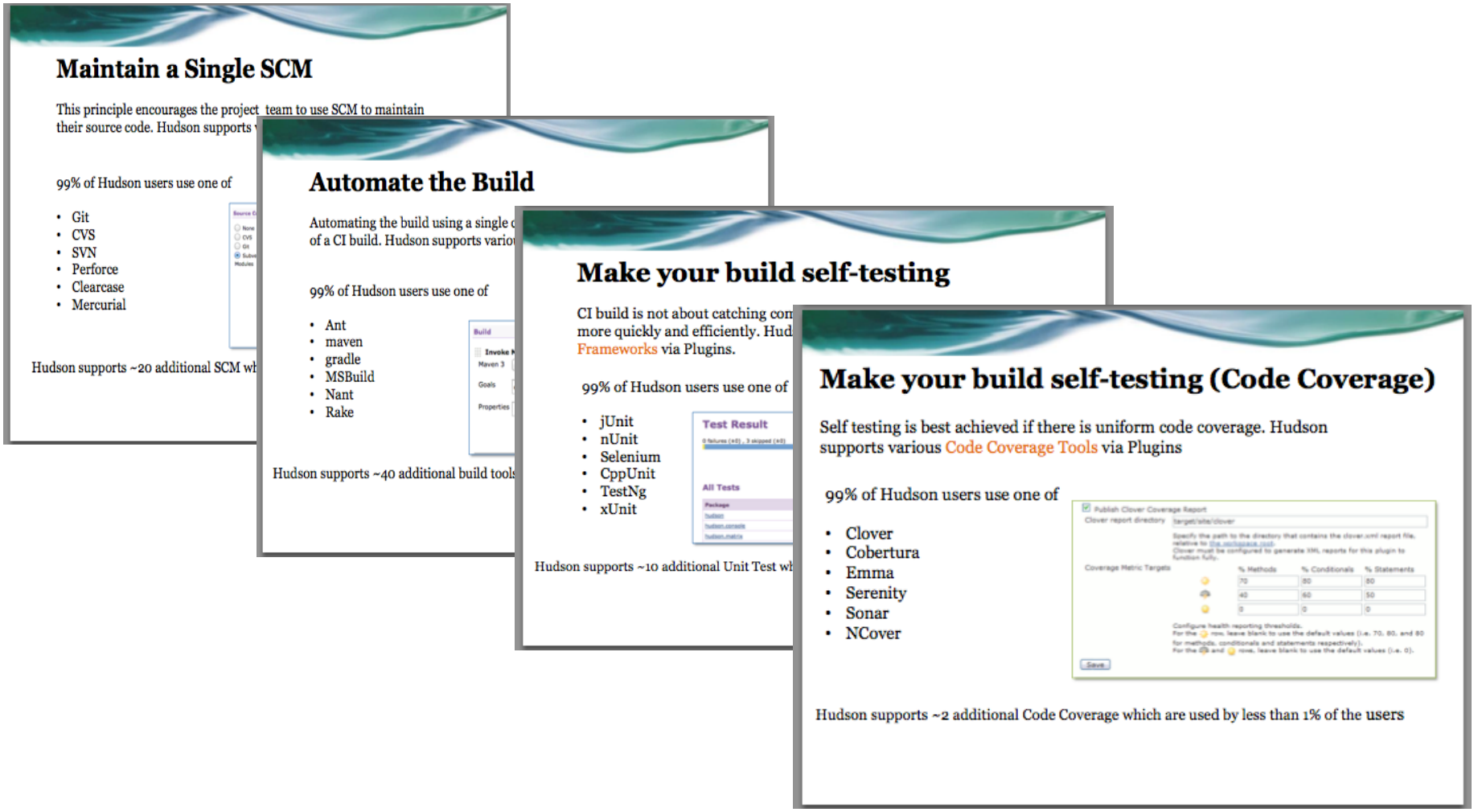
Effective practicing of CI



- *Maintain a Single Source Repository.*
- *Automate the Build*
- *Make Your Build Self-Testing*
- *Everyone Commits To the Mainline Every Day*
- *Every Commit Should Build the Mainline on an Integration Machine*
- *Keep the Build Fast*
- *Test in a Clone of the Production Environment*
- *Make it Easy for Anyone to Get the Latest Executable*
- *Everyone can see what's happening*
- *Automate Deployment*

Choosing Hudson Plugin for Effective practicing of CI

http://wiki.eclipse.org/Hudson-ci#Hudson_Plugins

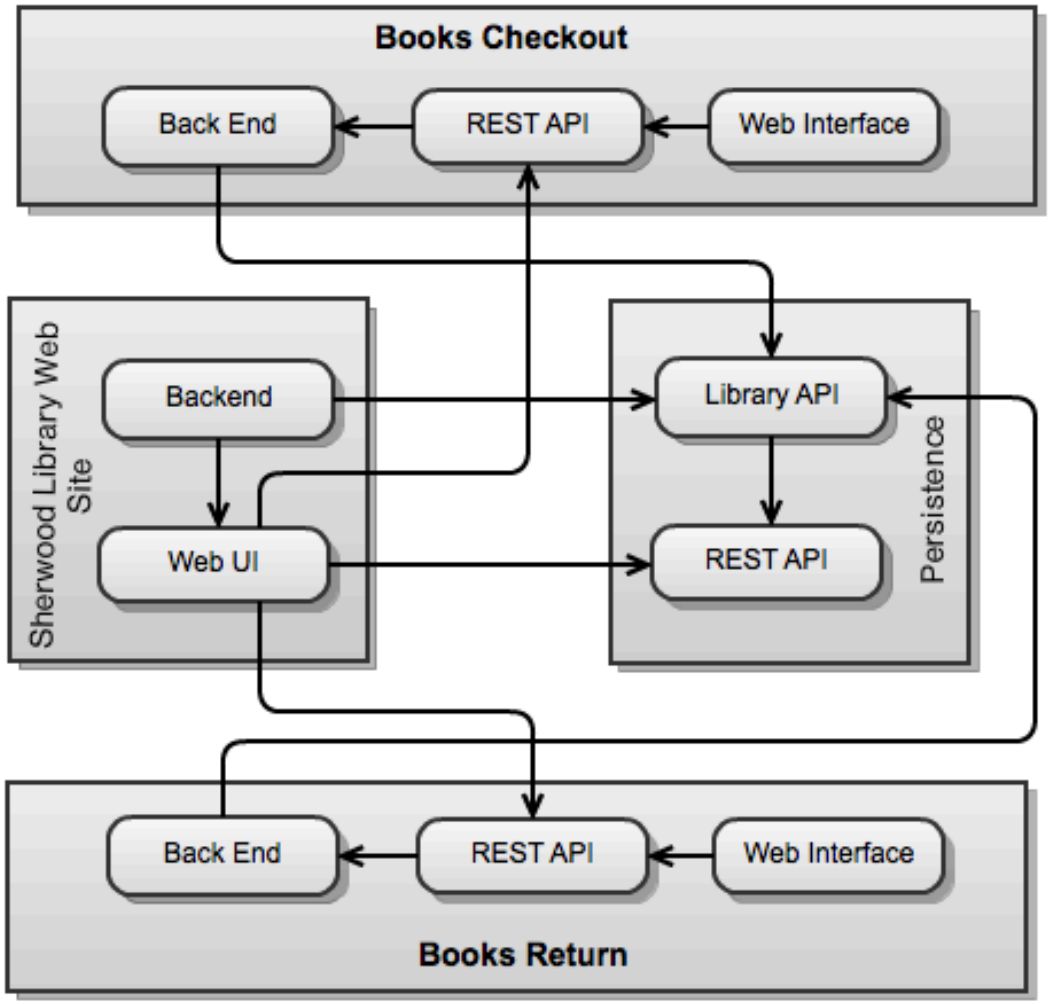


Buildable Units

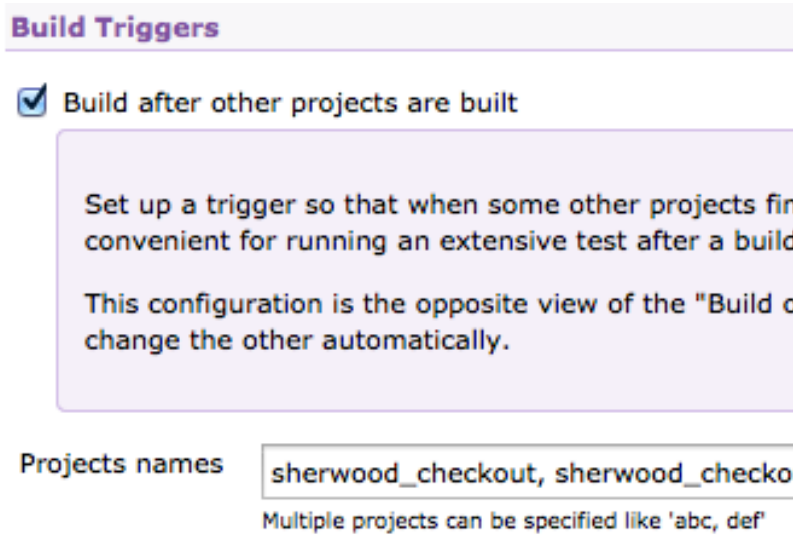
Important guideline of CI is to build fast and give back feedback quickly. To achieve this

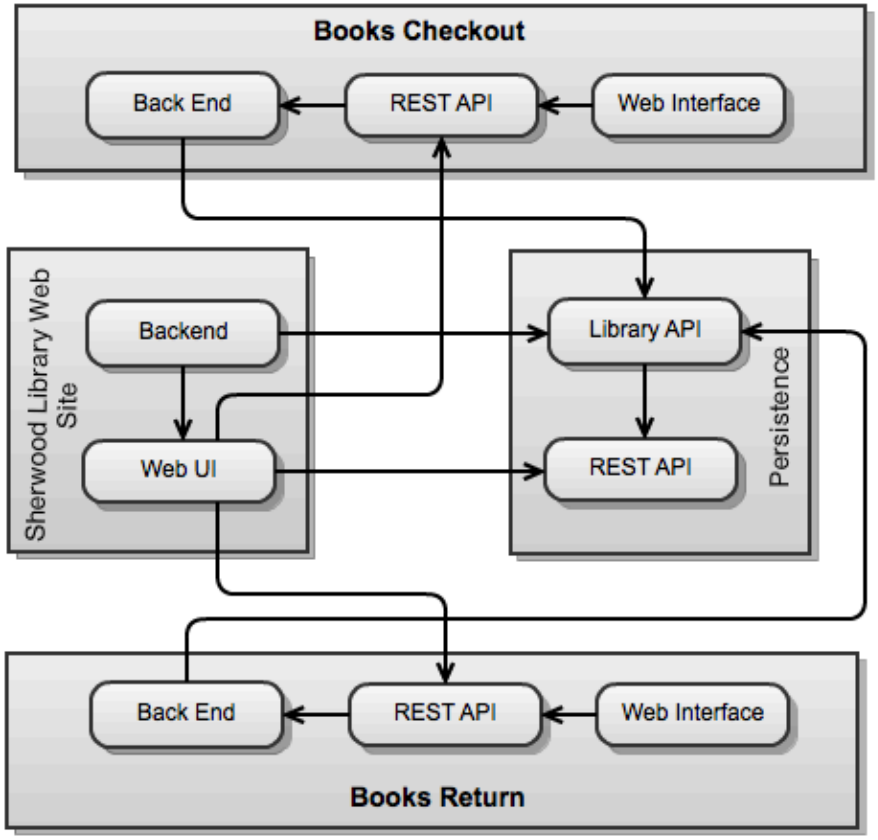
- Rather than building the entire source in one single job, divide the project sources into buildable chunks. Each chunk of software must be able to build independent of each other.
- The dependent chunks must be built separately and stored in an artifact repository manager for other software chunks to use them as dependencies.
- Each of the software chunks is a buildable unit and is built by a single Hudson job.

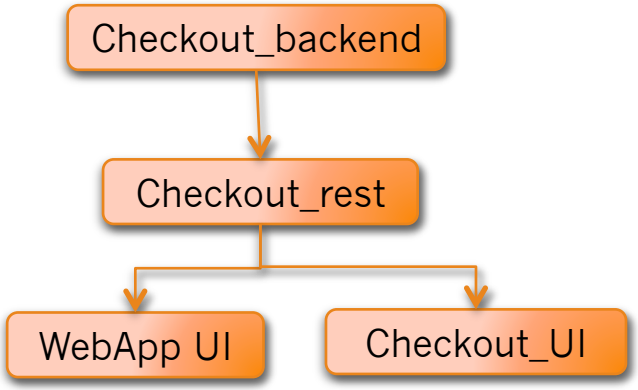
Sherwood Library Build Architecture



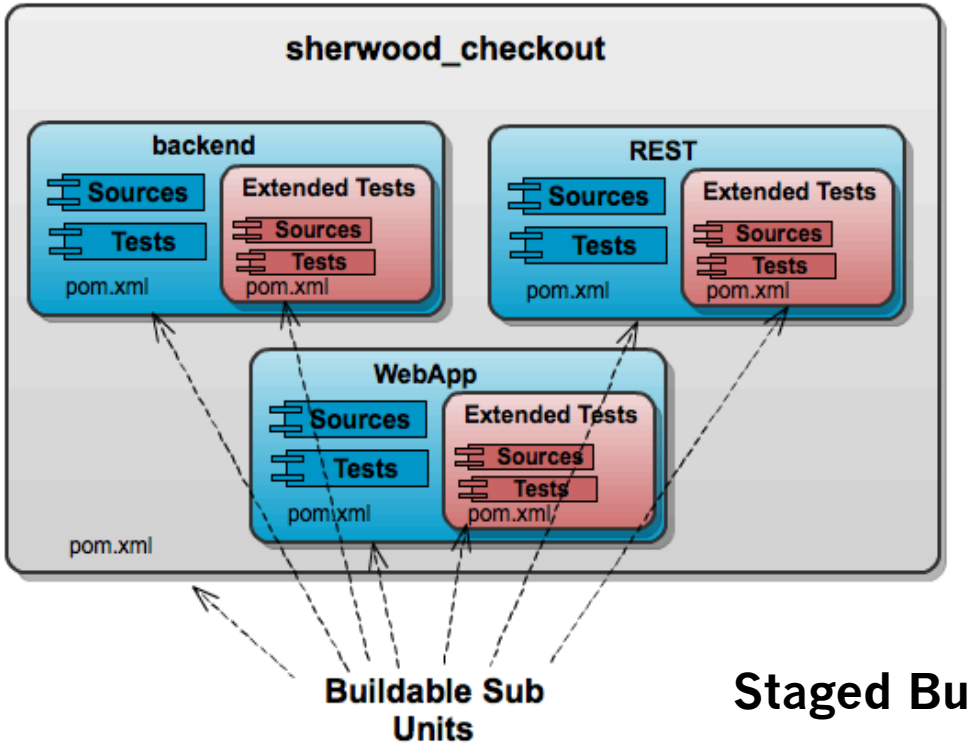
Setting up upstream-downstream builds

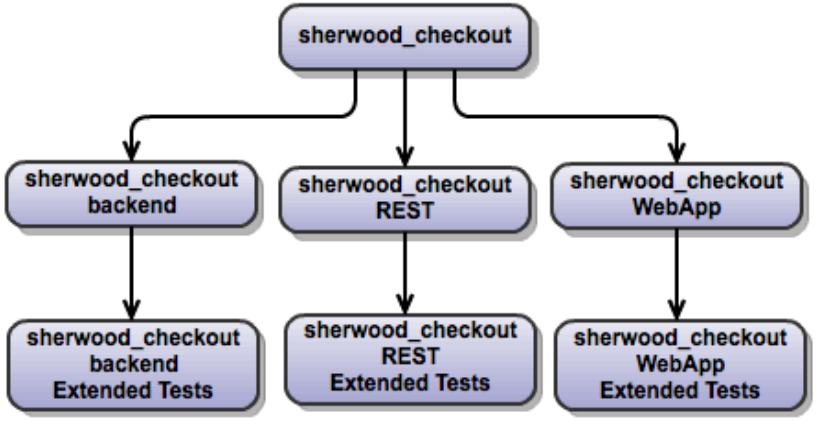






Speeding up CI Builds





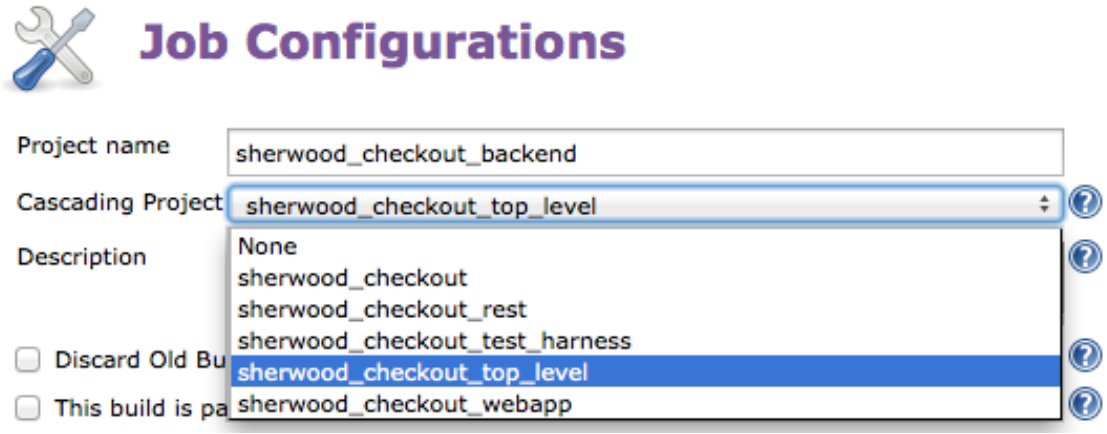
Staged Builds

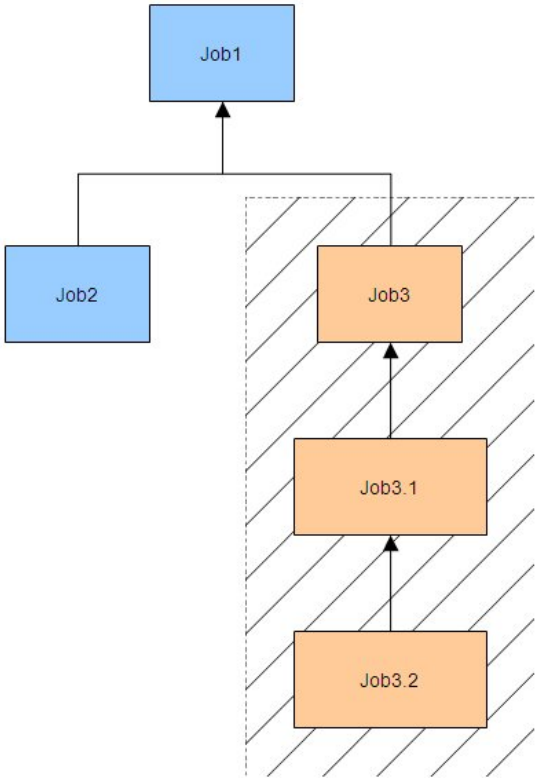
- The first stage would do the compilation and localized unit tests. The unit tests may be created with out any real time database or server connections to keep it fast. (Mockito, Powermock)

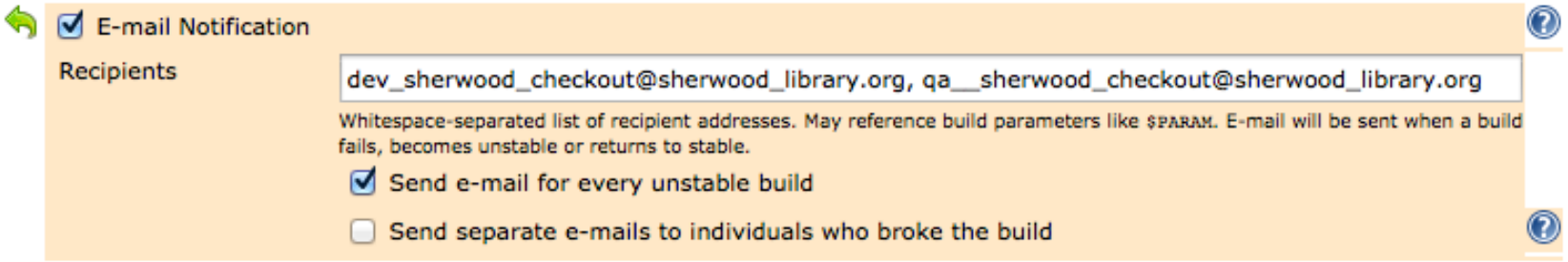
- In the second stage, the extended builds run different suits of tests, may be with real time server and database connections.

Build pipeline plugin

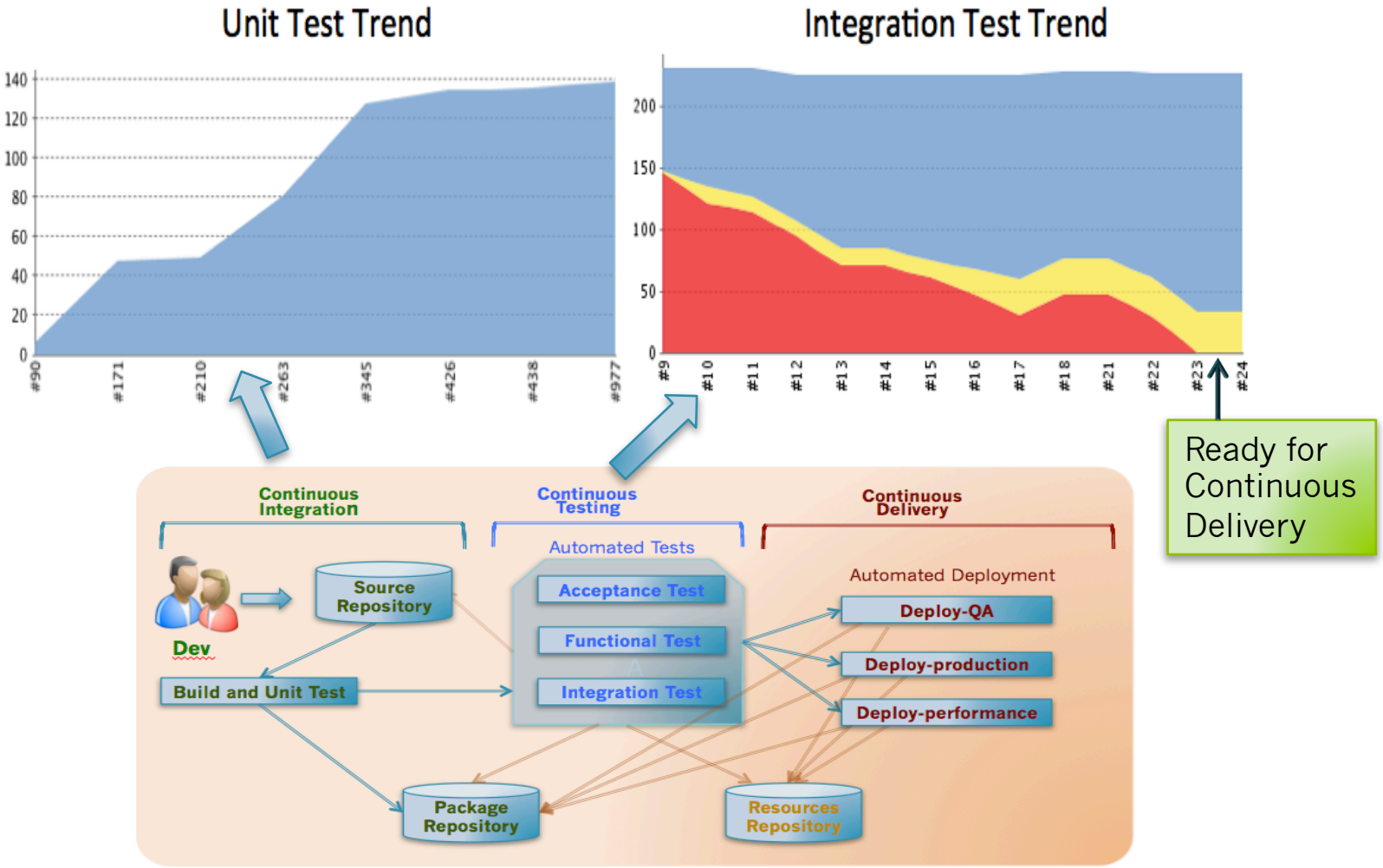
Cascading jobs





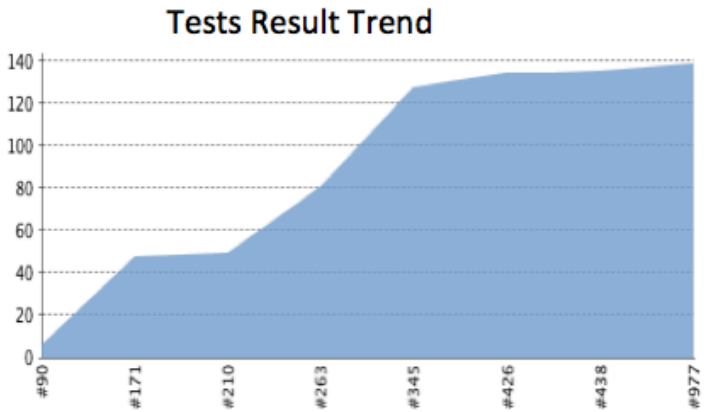


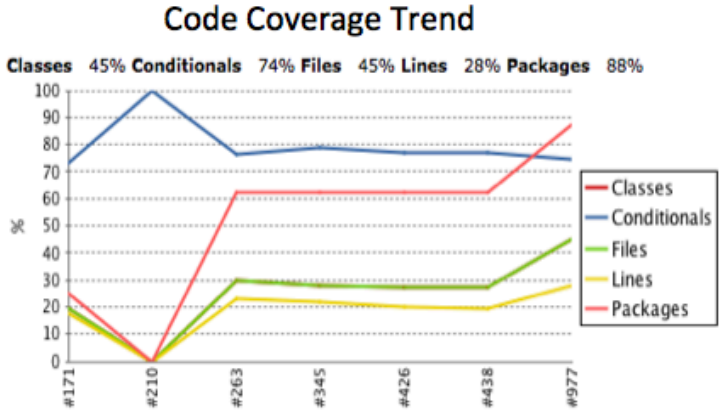
Monitor Test Trends

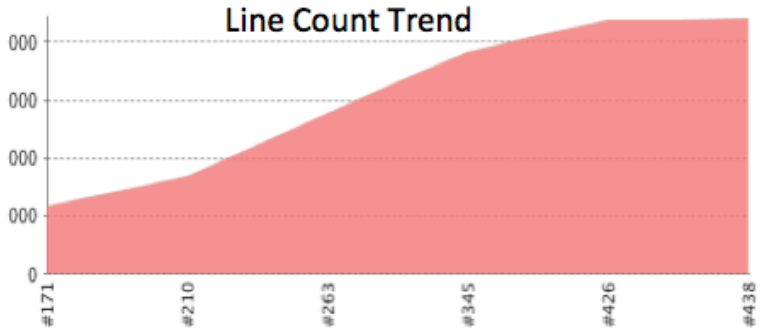


In a CI build, the unit tests should never fail.
During the initial stage of the project, the integration test may be in flux.

Monitor Quality Metrics Trend





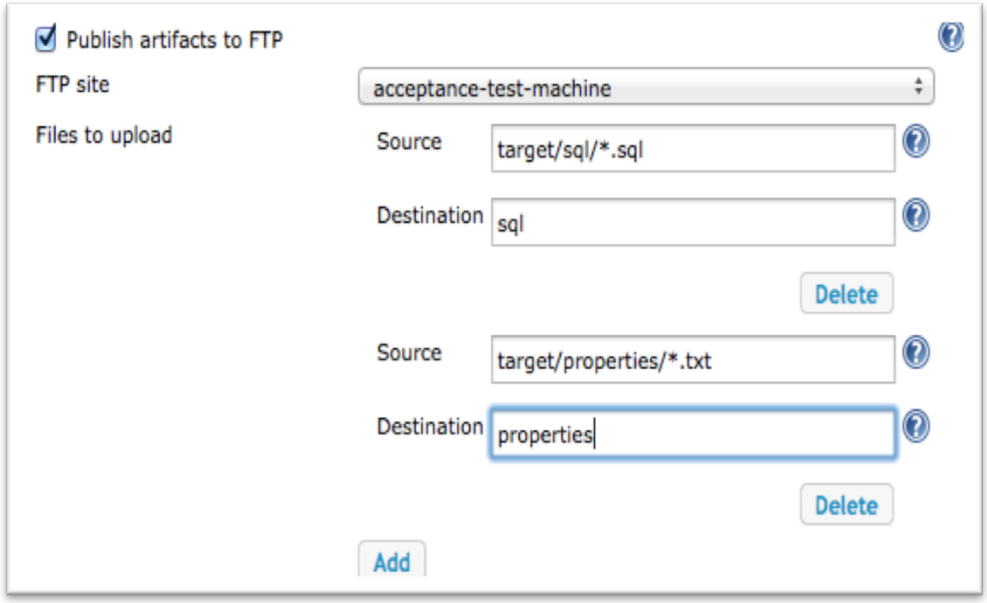


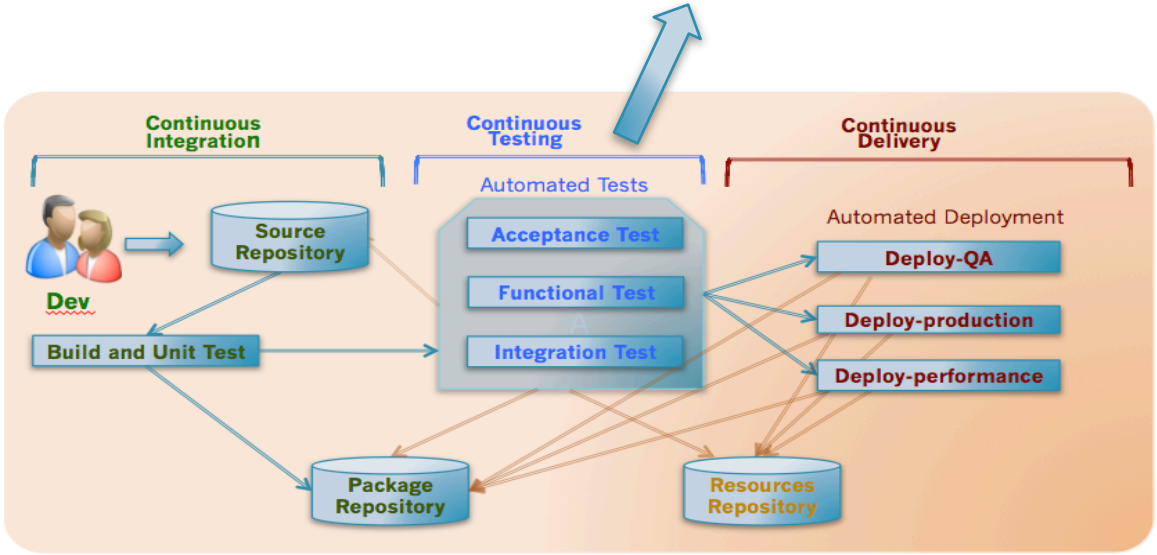
Code quality measurement is important in Continuous delivery.
Improves the confidence of the product being deliverable state.

Automated Upload

(FTP)

As part of Build pipeline, often there may be requirements to copy configuration files or database schemas, test scripts, properties files, install scripts etc., which are part of a build to another machine to facilitate additional test run

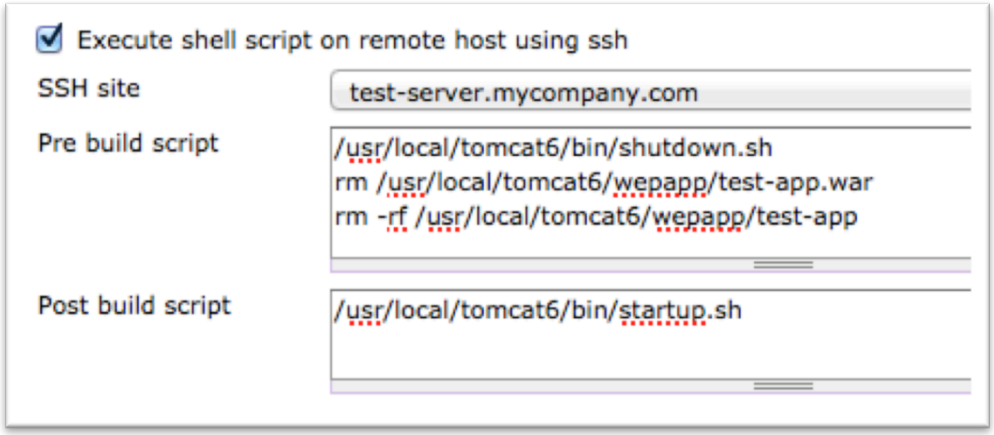


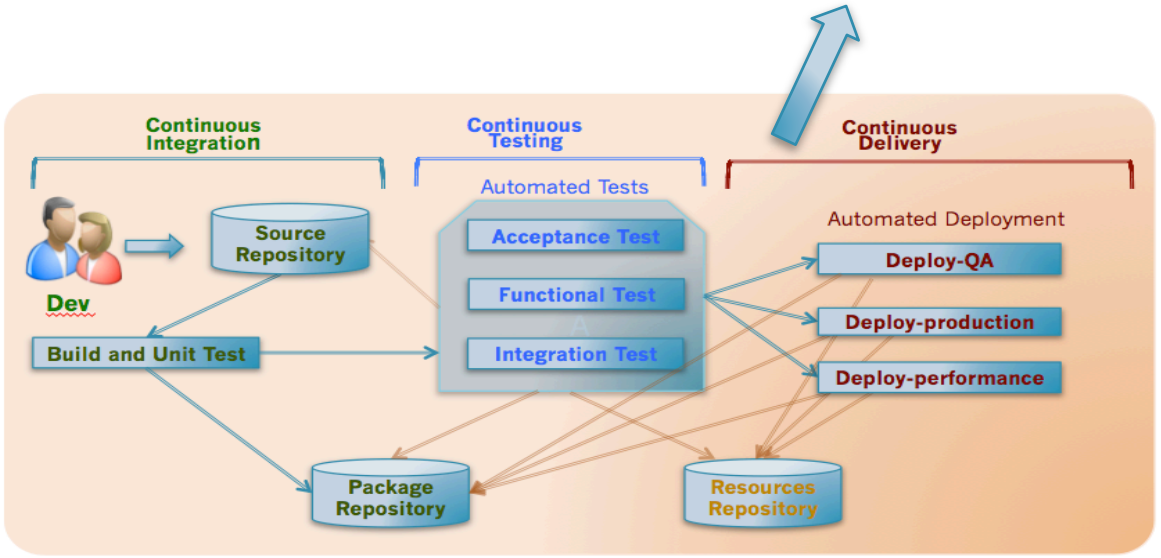


Automated Execution

(SSH)

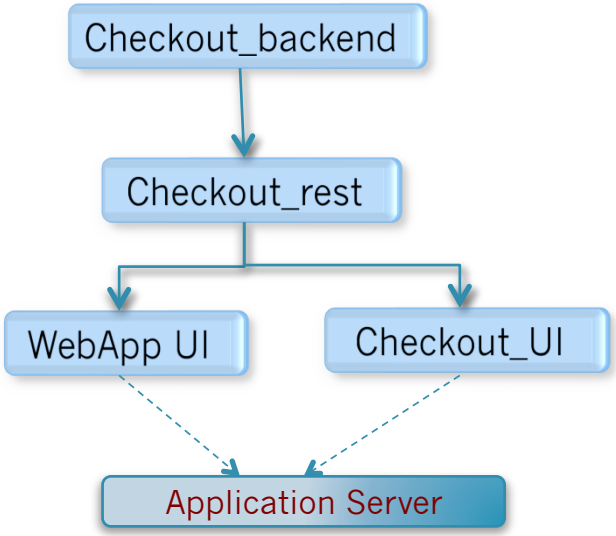
Execute commands on that remote machine to ready the machine for automatic deployment.

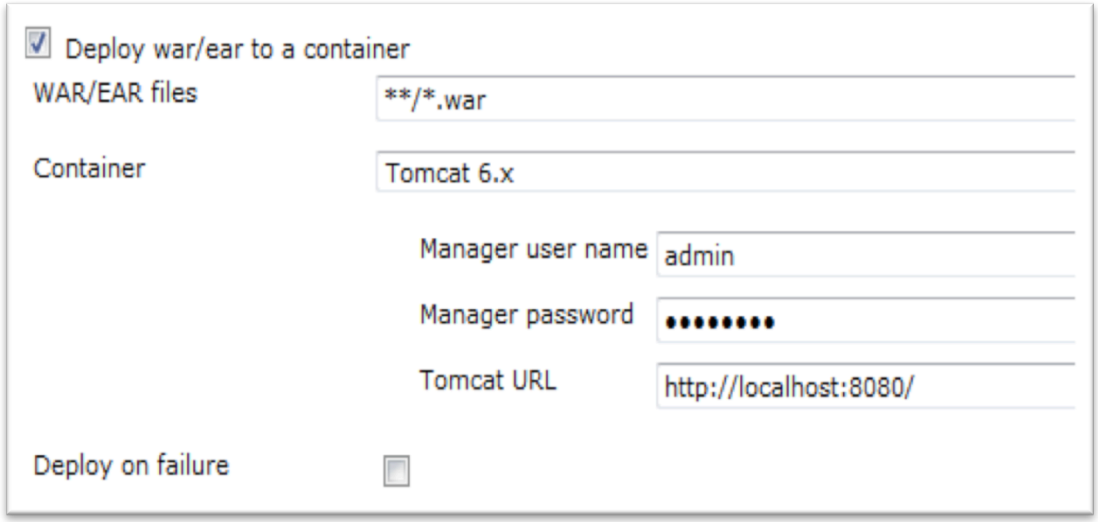


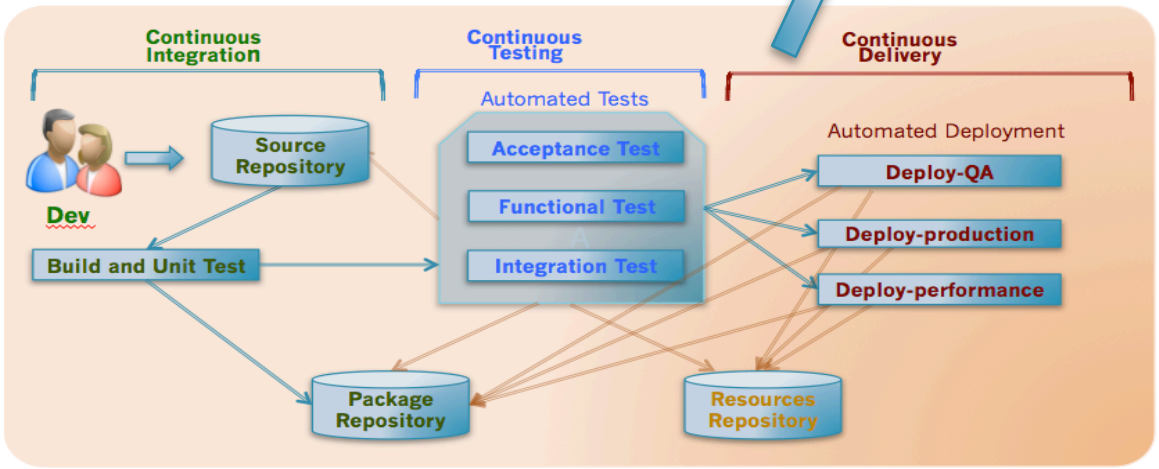


Automated Deployment

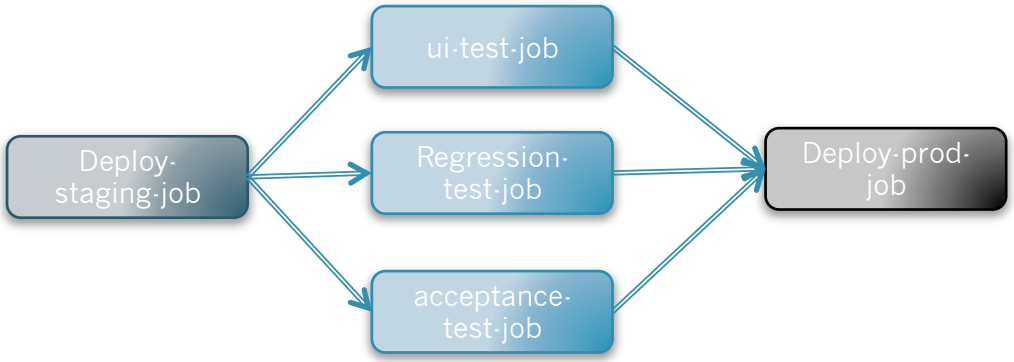
(Application Servers)

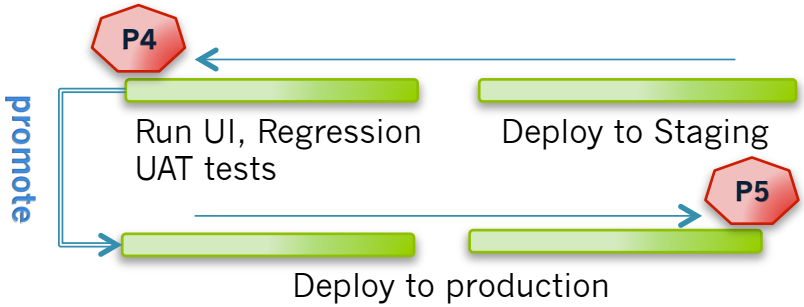




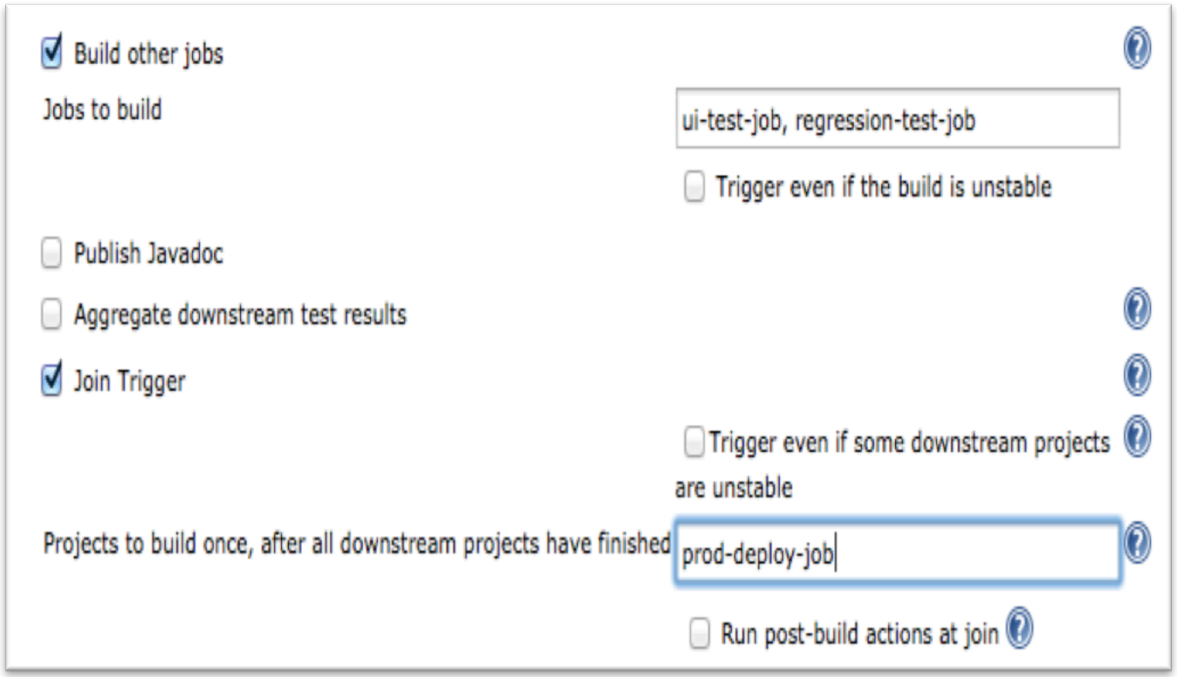


Diamond build pattern



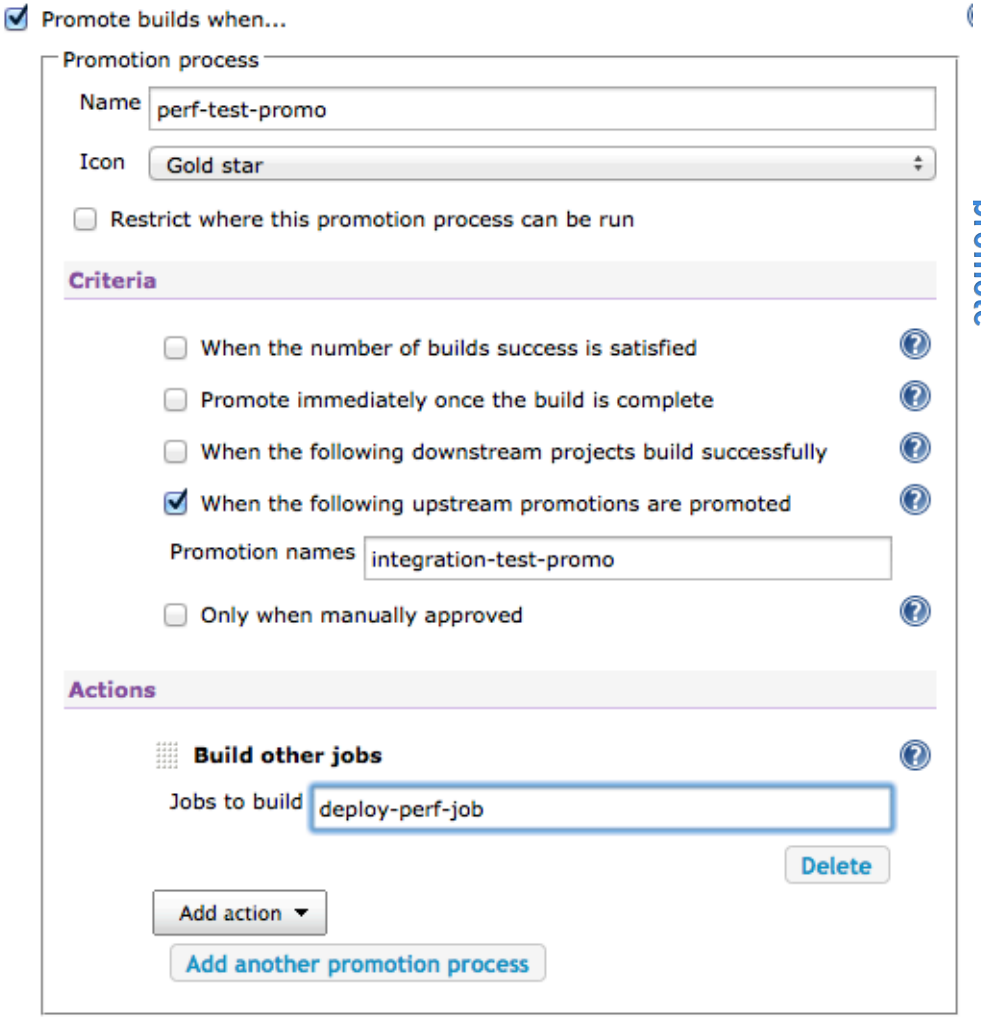


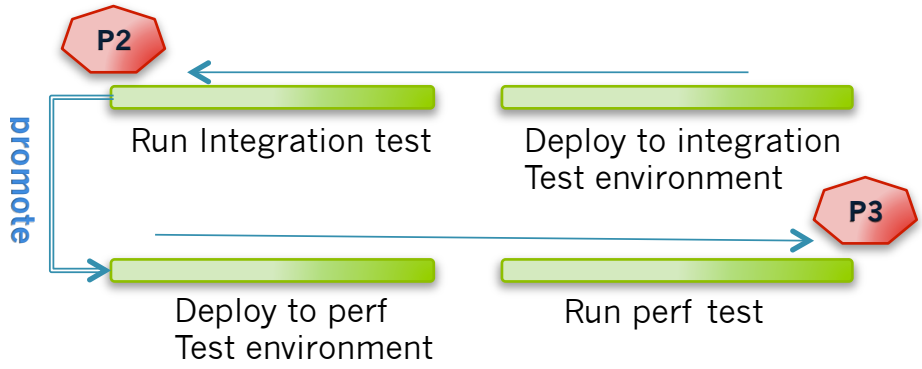
Join plugin triggers a job after all the downstream jobs are completed in parallel. This allows a pipeline to branch out to perform many steps in parallel, and then run another job after all the parallel jobs are finished.



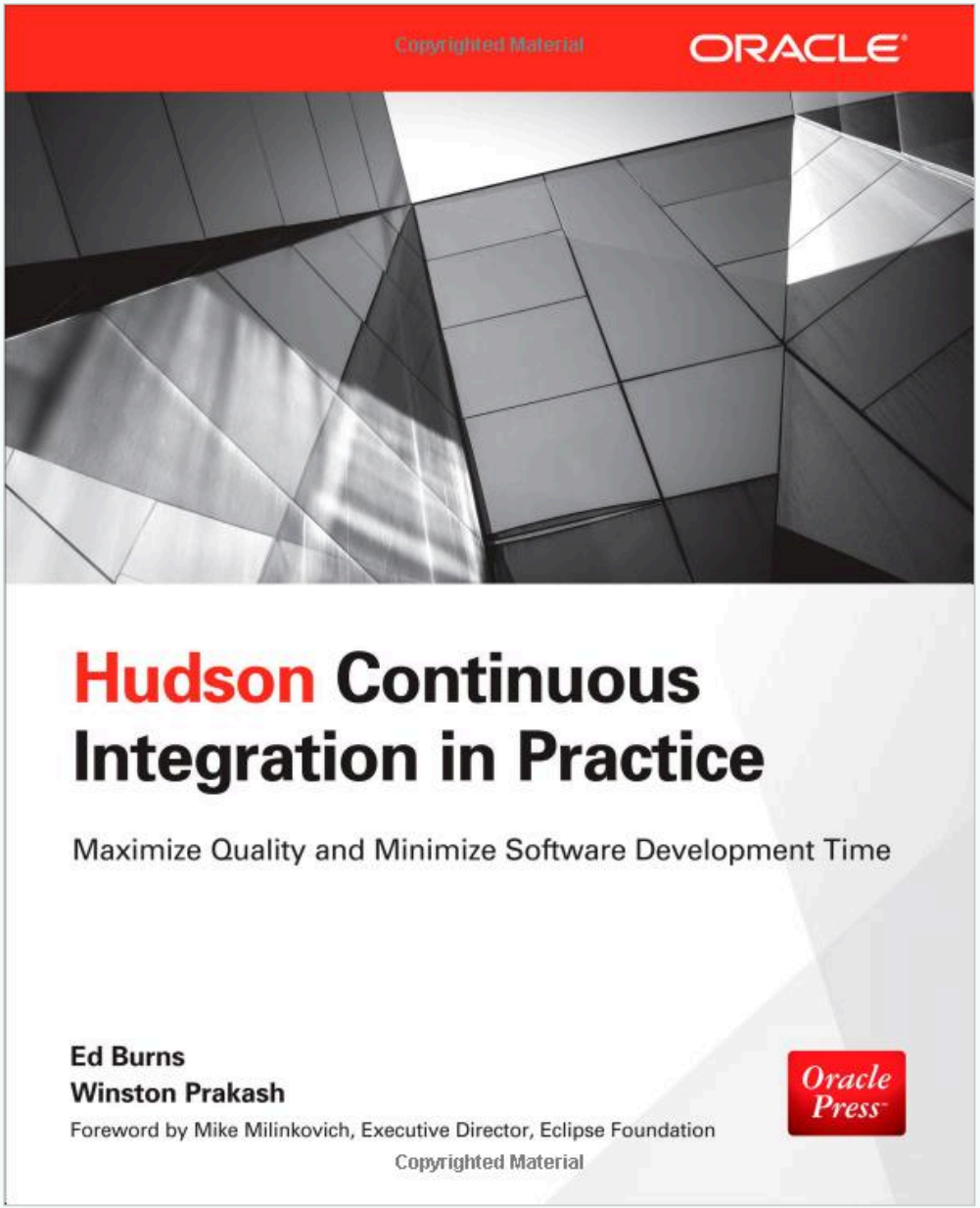
Automated Promotion

Though promoted build plugin provides opportunity to promote every build, typically the promotion process is done for a pipeline





Hudson Book



Part1:

Practicing effective CI using Hudson

Part2:

Hudson plugin development

http://www.amazon.com/Hudson-Continuous-Integration-Practice-Burns/dp/0071804285