



Continuous Integration and Bamboo

Ryan Cutter
CSCI 5828 2012 Spring Semester

Agenda

- What is CI and how can it help me?
- Fundamentals of CI
- Fundamentals of Bamboo
 - Configuration / Price
 - Quick example
 - Features / Discussion
- My Takeaways on CI



Continuous Integration

- Originated in Extreme Programming but many Agile workplaces use it
- Deploying continuously is essential to streamlining the feedback loop, a core Agile tenant
- CI usually means an entire project is rebuilt upon any change to code base
- Swap long, laborious integration efforts with short, automated ones
 - Sound like a fair trade?



CI Benefits

- Bugs more obvious
 - Assumes system tests tightly coupled with CI implementation
- Reduces risk
 - Many non-Agile projects are “on time” until they hit the testing stage!
 - CI makes problems easier to predict and more obvious
 - The earlier a problem is detected, the cheaper it is to resolve



CI – Think of it this way

- The longer developers wait before integrating their code with each other, the higher the chance for diverging or fractured efforts
- Be concerned if you're not working with the latest code iteration every day
 - Even intraday updates beneficial
 - At least update and recompile before committing new code!

And the lord said unto man

“He who checks in files which
break the build for others
shall pay a penance of
doughnuts”

That is the law



CI In Action



1. Check out code from source control
2. Make changes, compile locally, repeat
3. Commit code back into source control
4. CI tools automatically build project on separate machine
5. If compilation is successful and tests pass, you're done
 1. If not, at least the problem was caught quickly!
6. [Optional] Automatically deploy somewhere

Martin Fowler's take on CI



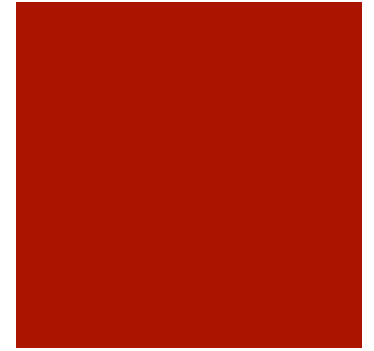
- Published 6 years ago, still quoted extensively
 - <http://martinfowler.com/articles/continuousIntegration.html>
- 1. Maintain single source repo**
- 2. Automate the build**
- 3. Make your build self-testing**
- 4. Everyone commits to the mainline every day
- 5. Every commit should build the mainline on an integration machine

Martin Fowler's take on CI



6. Keep the build fast
- 7. Test in a clone of the production environment**
8. Make it easy for anyone to get the latest executable
9. Everyone can see what's happening
- 10. Automate deployment**

Deeper Dive



- Maintain single source repo
 - Hopefully this goes without saying
 - git has done a tremendous job making source control management ubiquitous
 - Plenty of SCM options available
 - Source code on a developer's local machine grows more stale with every hour that passes without an update
 - Integration risks obviously go up

Deeper Dive

- Automate the build
 - Developers shouldn't have to do anything manually after committing code
 - If they do, you're courting trouble
 - Should be able to build individual portions of project



Deeper Dive

- Make your build self-testing
 - Suite of automated tests check the code base
 - Requires buy-in from developers to seek high coverage
 - JUnit a good example of unit-testing framework
 - Write new functionality then write a JUnit test function
 - Helps spot problems early and without human discovery



Deeper Dive



- Test in a clone of the production environment
 - Mimic production configuration to the greatest extent possible
 - All libraries, drivers, support apps (like databases) should be the same as what you use to deploy
 - Virtualization (such as using software like VMware) is a great way to increase coverage to many different possible environments
 - You don't worry about setting up these environments – it's all magic as far as you're concerned

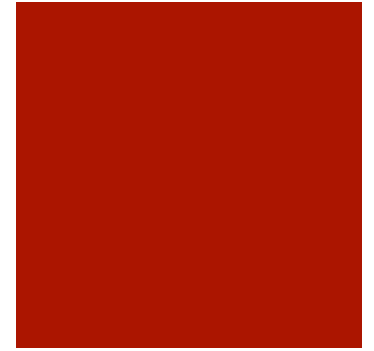
Deeper Dive



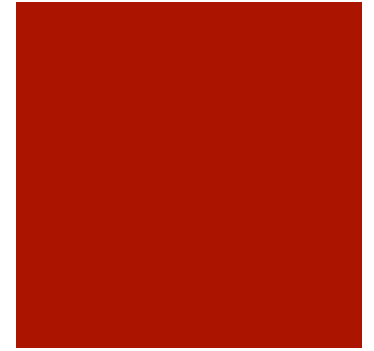
- Automate deployment
 - Continuous Integration is not the same thing as Continuous Deployment but it makes sense to consider the two concurrently
 - Fowler focuses on deploying to production (insert debate here) remember internal deployment just as important
 - < 30 mins after code committed, change should be available
 - If it's a web app, it should be on a test/QA server
 - If it's a "desktop" app, it should be available for download (on Nexus, for example) or running in a virtualized environment

More on Deployment

- Difficult to overstate the value of continuous or “one click” deployment
 - If you’re manually copying RPMs, you’re probably doing it wrong
 - CI systems support custom scripting to integrate this into automated workflows
 - Not only that, but they’ll log these operations (more on this during the Bamboo discussion)



CI Solutions



- Jenkins
 - Open source (MIT License)
 - Forked from Hudson following dispute with Oracle
- AnthillPro
 - Been around for a long time, very mature
 - Full enterprise solution
- Bamboo
 - We'll focus on this product
 - Why? Because I like it

Bamboo: Highlights



- Made by Atlassian
 - They also produce JIRA and Confluence, among others
 - Very popular dev tool company
- Out of the box support for Java, .NET, PHP, Javascript among others
- Completely automated with slick web UI
- Ability to deploy parallel builds to multiple agents
- Build pipelines make it easy to create workflows
 - Analytical tools help highlight what went wrong
- Continuous deployment is snap!

Bamboo: Configs/Pricing



- OnDemand and Download versions
 - OnDemand is AWS-hosted solution
 - Downloads for Unix/Linux, Mac, Windows available
- 30 day free trial
- For a typical startup producing a normal web app (imagine yet another iteration of “social”), for unlimited jobs and 1 – 10 build agents:
 - OnDemand: \$50 - \$250/month
 - Download: \$800 - \$4,000/month
 - Prices as of March 2012
- Higher price points for up to 100 agents available

Bamboo OnDemand

- With “Elastic Bamboo” your build agents live on EC2
- Integrates with local source repo
- The catch is you need to purchase FishEye and Crucible OnDemand as well (though this restriction is being removed shortly)
 - FishEye is a better source history with hooks into JIRA
 - Crucible assists in code reviews



Bamboo: Setup

- Configure the default local agent
 - Sets JDK, builder, etc capabilities
- Create a new plan
 - Build plan is the recipe for a build
 - Defines what gets built (svn repo), how build is triggered, what builder (compiler) to use, artifacts to be created, etc
 - Always associated with a project
 - Define the trigger that will initiate the build
 - Next screen shows screenshot of this step



Bamboo: Create a plan



Screenshot: 'Source Repository — CVS'

1. Plan Details 2. Source Repository 3. Builder 4. Requirements 5. Artifacts 6. Notifications 7. Post Actions 8. Permissions

Source Repository

Repository:

CVS Root: *
The full path to your CVS repository root. Bamboo supports pserver, ext (ssh) and local repository access methods.

Authentication Type:

Password:
(Optional) The password required to access the CVS repository

Quiet Period: *
CVS checkins are not atomic. How many seconds should Bamboo wait between checkins to determine if the checkin is complete?

Module: *
The repository module containing the source code.

Version of module: *
Which version of the module should Bamboo build?

Common repository configuration

Force Clean Build
Removes the source directory and checks it out again prior to each build. This may significantly increase build times.

Include / Exclude Files:
Customise what files Bamboo uses to detect changes.

File Pattern:
A [regular expression](#) to match the file to be included / excluded.

Web Repository:

Web Repository URL:

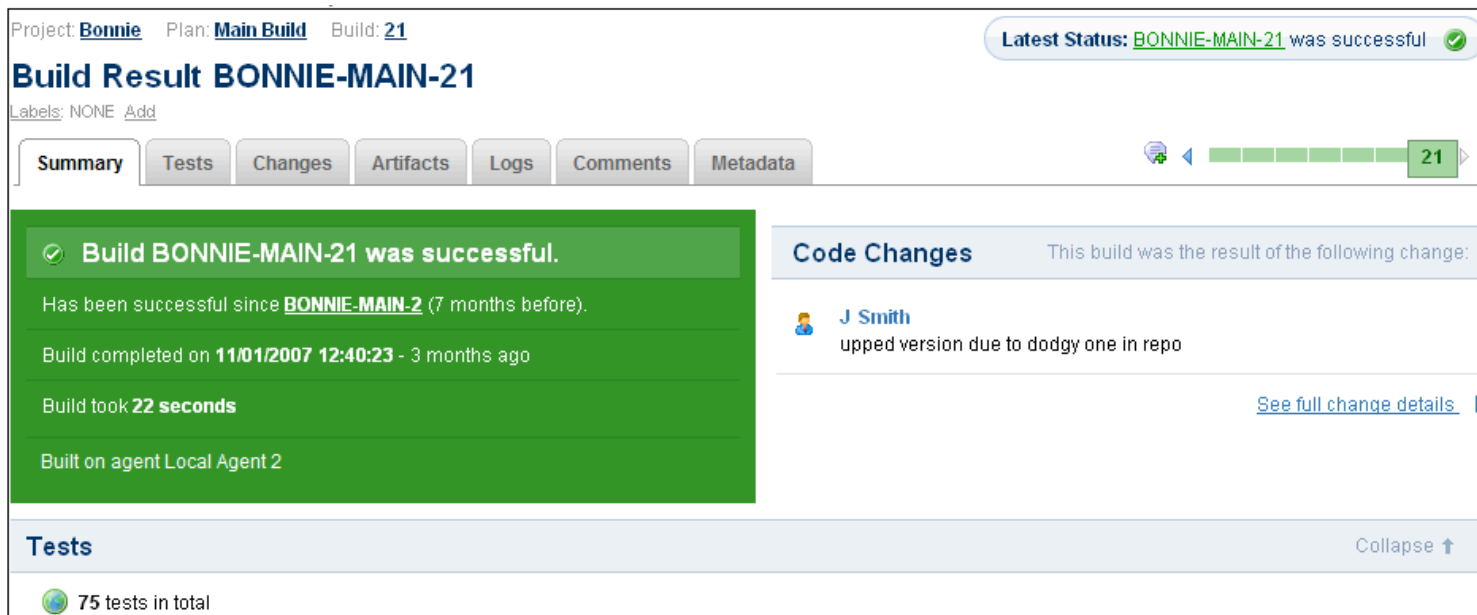
Web Repository Module:

Build Strategy: *
How should Bamboo detect that the source repository has changed?

Polling Frequency:
How often (in seconds) should Bamboo check the repository for changes?

Bamboo: Execution

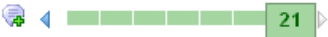
- After plan run, *everything* can be evaluated
 - Test results
 - Build log
 - Build artifacts



Project: [Bonnie](#) Plan: [Main Build](#) Build: [21](#) Latest Status: [BONNIE-MAIN-21](#) was successful ✓

Build Result BONNIE-MAIN-21

Labels: NONE [Add](#)

[Summary](#) [Tests](#) [Changes](#) [Artifacts](#) [Logs](#) [Comments](#) [Metadata](#) 

✓ **Build BONNIE-MAIN-21 was successful.**

Has been successful since [BONNIE-MAIN-2](#) (7 months before).


Build completed on **11/01/2007 12:40:23** - 3 months ago

Build took **22 seconds**

Built on agent Local Agent 2


Code Changes

This build was the result of the following change:

 **J Smith**
upped version due to dodgy one in repo

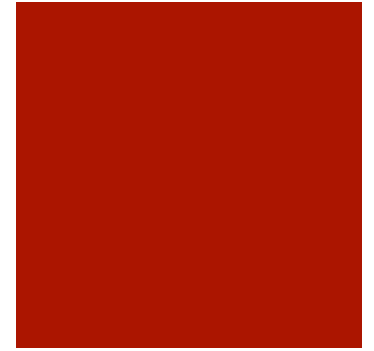
[See full change details](#) ▶

Tests [Collapse](#) ↑

 75 tests in total

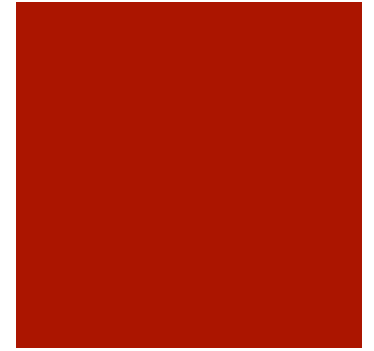
Bamboo: Execution

- Failed builds do not simply result in a failure message
 - Get deep analytics into trouble spots
 - Stack traces
 - Drill into context by viewing changes to build attempt with JIRA and Fisheye



Bamboo: Release

- Build release pipeline in stages to facilitate automated release
- Release using JIRA, build with Bamboo
 - Seeing a trend here? Bamboo and JIRA are tightly coupled



Bamboo: Capabilities

- Integrates with many source control systems
 - Subversion, Mercurial, Git, Perforce, CVS
- Same for the popular build systems
 - Ant, Maven, Make, Command Line, MSBuild
- Ditto for test automation
 - Junit, Selenium, PHPUnit



Bamboo: Integration



- Atlassian not surprisingly provides deep hooks into its other products
 - JIRA – Able to associate tasks with builds, view dashboards/wallboards (see next slide)
 - If you use JIRA or Bamboo, you really need to go for the another
 - Confluence – Track builds in your wiki
 - Clover – Monitor historical code coverage in your build

Bamboo: Integration

- IDE Plugins for Eclipse and IntelliJ
 - Allows you monitor builds through your IDE
- Blitz.io, Tomcat, Xcode, Vmware
- API for further custom development
- REST service too!
 - Extremely simple to build custom tools using REST to view status, kick off new builds, etc



Bamboo: Builds



- Build Triggers
 - Builds can be started with commit-triggers, schedule, and dependency triggers
- Build Dependencies
 - Share artifacts between compilation, testing, and deployment stages
- Queue Management
 - Fine grain control over queue
 - Re-order queue, set timeouts, detect hung builds

Bamboo: Notifications

- Everything you would expect from a CI system
 - SMS
 - Email
 - RSS
 - IM
 - Two-way IM permits adding comments/labels to build results without entering Bamboo



Bamboo: Wallboard



Imagine this on a 55" LCD in your developers' space!



My Musings

- These are only opinions – disregard as desired!
- Almost every team needs someone **dedicated** to CI/build management
 - Not a collateral duty or “you broke the build so now you’re the build manager” – these are paths to failure (or at least mediocrity)
- Gold plating CI is a force multiplier so don’t skimp on it
 - Not just manpower – throw good SW/HW at it
 - Builds should be compiled and deployed quickly
- Developers should be allowed to focus on coding and reap the benefits of good CI



Questions?

- Very thorough Bamboo documentation available:
 - <http://www.atlassian.com/software/bamboo/overview>
- Additional information about CI used in this brief:
 - <http://www.ibm.com/developerworks/java/tutorials/j-cq11207/>

