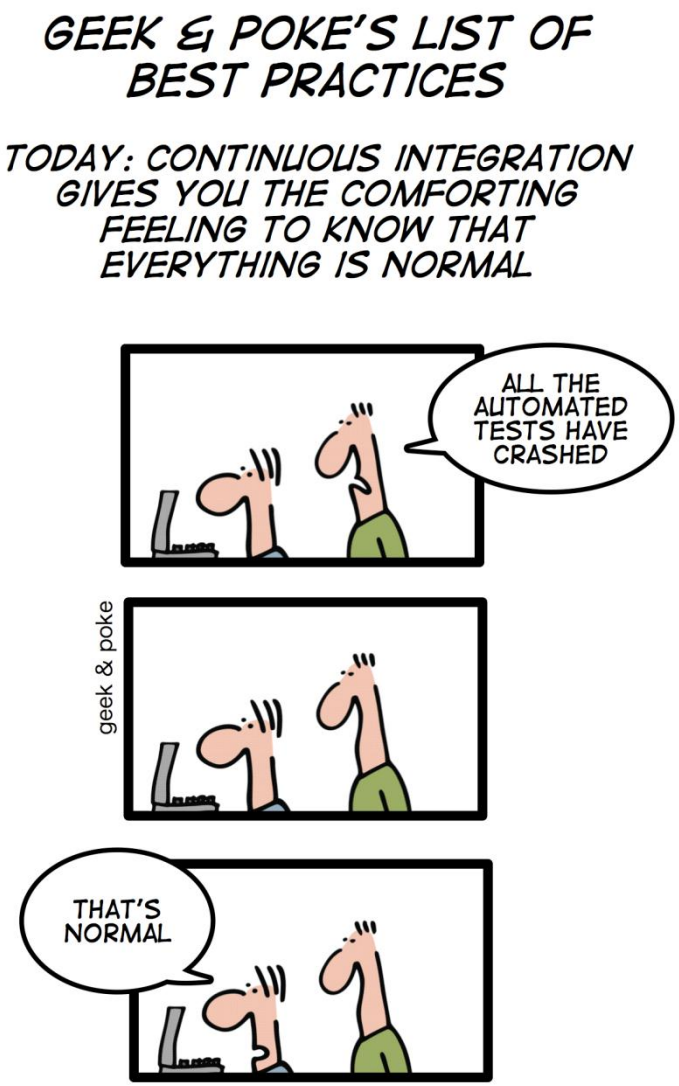

Continuous Integration & Deployment

Hudson/Jenkins

geek & poke/Oliver Widder

“Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily – leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly.”

Martin Fowler

XP's twelve practices (1/4)

- Twelve practices classified in:
  - Coding practices
  - Developer practices
  - Business practices

XP's twelve practices (2/4)

- Coding practices
  - **Code** and design **simply**
    - To produce software that's easy to change
  - **Refactor** mercilessly
    - To find the code's optimal design
  - Develop **coding standards**
    - To communicate ideas clearly through code
  - Develop a **common vocabulary**
    - To communicate ideas about code clearly

XP's twelve practices (3/4)

- Developer practices
  - Develop **test-driven development**
    - To prove that the code works as it should
  - Practice **pair-programming**
    - To spread knowledge, experience, and ideas
  - Adopt **collective code ownership**
    - To spread responsibility for the code to the whole team
  - **Integrate continuously**
    - To reduce the impact of adding new features

XP's twelve practices (4/4)

- Business practices
  - Add a **customer to the team**
    - To address business concerns accurately and directly
  - Play the **planning game**
    - To schedule the most important work
  - **Release regularly**
    - To return the customer's investment often
  - Have a **sustainable pace**
    - To go home tired, but not exhausted

Developer practice: Integrate continuously

- CI tightens the feedback loop when creating a software
- Small and frequent adjustments…
  - … have fewer collisions than large and dramatic changes
  - … are easier to correct if collisions occur
- CI supports:
  - Refactoring (coding practice)
  - Releasing regularly (business practice)

But, wait, what is integration?

- Long and unpredictable process
- Consists in assembling individually tested software components to form a larger software
- Usually done by performing integration tests
- Integration tests are aimed at ensuring that all individually developed and tested components work well together

CI benefits

- Reduced risks
- Eases bugs detection and fixing
- Reduces/avoids the broken windows syndrome
- Eases frequent deployments

CI principles (1/2)

- Check out the latest code frequently
- Merge new tasks and tests into the main source code (that is, integrate) frequently (usually daily) and as soon as they're completed
- Run the tests to ensure that the new code fits well into the system
- Quickly fix any errors/bad builds

CI principles (2/2)

- Keep tasks small
- Keep the integration itself simple
  - Run the tests, merge and commit
  - No more
- Keep the integration quick and painless
- Automate as much as you can

A typical CI workflow (1/3)

- Complete a task by
  - Adding/changing tests
  - Adding/changing the production code
- Run the automated build on the local machine

A typical CI workflow (2/3)

- If OK
  - Update working copy
  - Merge the changes and run the local automated build again, if necessary
  - Commit into the repository
  - Do the integration build
  - Fix the bad build, if necessary

A typical CI workflow (3/3)

- If KO
  - **Fix it** and **do the whole workflow** again!

CI requirements

- A RCS repository
- Everything to be in this repository (developers' machines should be stateless)
- Most of the time, a CI server
- Comprehensive testing (that is, a high degree of automated tests)
- Quick test suite (integration should be kept fast)

CI servers (1/6)

- CruiseControl
  - BSD-style license
  - http://cruisecontrol.sourceforge.net/
  - Grandfather (!) of CI servers
  - Lots of extensibility
  - XML-based configuration

CI servers (2/6)

- Apache Continuum
  - Apache Public License
  - http://continuum.apache.org/
  - Restricted set of handled tools
    - Maven/Maven2
    - Ant
    - Shell scripts

CI servers (3/6)

- Atlassian Bamboo
  - Proprietary software
  - http://www.atlassian.com/software/bamboo/
  - Can be extended through the use of plugins

CI servers (4/6)

- Hudson
  - MIT License
  - Backed by Oracle and Sonatype
  - http://hudson-ci.org/
  - Great support of external technologies through the use of plugins
  - Doesn’t reduce itself to Maven builds

CI servers (5/6)

- Jenkins
  - MIT License
  - Fork from Hudson (January, 2010)
  - http://jenkins-ci.org/
  - Great support of external technologies through the use of plugins
  - Doesn't reduce itself to Maven builds

CI servers (6/6)

- Hudson/Jenkins
  - Used by Jenkins
    - http://ci.jenkins-ci.org/
  - Used by the Apache Software Foundation!
    - http://hudson.zones.apache.org/hudson/
  - Used by JBoss
    - http://hudson.jboss.org/hudson/
  - Used by eXo
    - http://builder.exoplatform.org/

Hudson/Jenkins: Demo

CI practice – Maintain a RCS repository

- Everything should be in the repository
  - Source code (including tests)
  - Build files
  - Additional libraries/software packages
  - Etc.
- The software should be built rapidly from a fresh machine by simply getting what's into the repository

CI practice – Automate the build

- Software should be built and tested using a single command
- If the software has to be deployed onto a runtime environment to be tested, the command must do it
- If a database has to be created and populated with data, the command must do it
- To keep the build fast:
  - Make the build as incremental as possible
  - Make steps of the build optional

CI practice – Make the build self-testing

- Again, include automated tests in the build
- Automated tests do include unit tests, but are not restricted to that
- May be achieved by
  - Applying XP
  - More simply, running test-driven development
- Test failure = build failure

CI practice – Commit the changes at least once a day

- Fixing issues quickly = finding them quickly
- Before committing, developers should
  - Update their working copy
  - Merge the changes
  - Run the automated build successfully

CI practice – Every commit should be built on the integration machine

- To be sure developers did have updated their working copy before committing
- To be sure developers have their development environment correctly set up
- The use of a CI server is then required
  - Builds can be started each time a commit is done (SCM polling)
  - Builds can be scheduled (daily, etc.)
  - Builds can be manually triggered

CI practice – Keep the build fast

- Rapid feedback is important
- Can be achieved by splitting the build into smaller builds
  - One build with compilation and unit-tests/mocked tests
  - Another build with full regression tests, database connectivity, etc.
  - If the second build fails, it may mean some tests are missing in the first build

CI practice – Test in a cloned production environment

- The goal of testing is to finally ensure that the software will run on a prod environment
- A difference between the test environment and the prod environment may be a source of issues

CI practice – Make it easy to get the latest deliverables

- In order to demo it, test it, deploy it, etc.

CI practice – Everyone can see the results of the builds

- Feedback is a key part of continuous integration
- Everyone must be able to see the current state of the software
- Think about:
  - Visual notifications
  - Mails
  - RSS feeds
  - Etc.

CI practice – Everyone can see the results of the builds

- Feedback is a key part of continuous integration
- Everyone must be able to see the current state of the software
- Think about:
  - Visual notifications
  - Mails
  - RSS feeds
  - Etc.

CI practice – Automate deployment

- Full testing requires the software to be deployed onto a runtime environment
- This should be automated
- Don't forget about automated rollback, or a least about backuping the as-is environment
- Hint: If you are able to deploy on an integration environment, you are surely able to deploy on a production environment...