www.spkaa.com
Ph: 888-310-4540

SPK and Associates
900 E Hamilton Ave, Ste.100
Campbell, CA 95008

Continuous Builds - A Customer Experience

Introduction

In our last article we discussed speeding up your build by leveraging tools such as Electric Accelerator. In this article we will discuss the process of Continuous Integration along with a customer experience.

What is Continuous Integration?

Continuous Integration (CI) is the practice of compiling & testing your software in an incremental fashion, utilizing on-going automation.

Via on-demand or scheduled, these compile->test cycles typically happen multiple times throughout the day.

CI offers early feedback to developers uncover issues early (before code is promoted to the official CM stream) which aids in quality improvement.

"An important part of any software development process is getting reliable builds of the software. Despite its importance, we are often surprised when this isn't done. We stress a fully automated and reproducible build, including testing, that runs many times a day. This allows each developer to integrate daily thus reducing integration problems."

Martin Fowler and Matthew Foemmel, Continuous Integration

"The macro process of object-oriented development is one of "continuous integration." ... At regular intervals, the process of "continuous integration" yields executable releases that grow in functionality at every release. ... It is through these milestones that management can measure progress and quality, and hence anticipate, identify, and then actively attach risks on an ongoing basis."

Grady Booch, Object-Oriented Analysis and Design with Applications

Continuous Integration Benefits Include

- Less time spent on builds/deployment
- Lower risk of error as operations are scripted
- Early notification of errors
- Easy to revert to bug-free state
- Constant availability of latest executables
- Shorter feedback cycles
- Let's you focus on real problems, not mechanics
- Makes your boss happy – saves time and $$$

Scott Bateman, Quorum Business Solutions, Inc

Continuous Integration Components

Martin Fowler has a comprehensive website detailing aspects of CI. He outlines components (Practices of CI) as the following:

- **Maintaining a Single Source Repository**

It is important to anchor your process with a robust software configuration management system. There are dozens of open source and commercial options to choose from (Subversion, CVS, Perforce, AccuRev, ClearCase, GIT…). All your artifacts required for the build must be

www.spkaa.com
Ph: 888-310-4540

SPK and Associates
900 E Hamilton Ave, Ste.100
Campbell, CA 95008

captured and managed through the SCM system. They include items such as:

- Source Code
- Build Scripts
- Test Scripts
- Test Data
- DB files and DB scripts
- Property and Configuration Files
- Project and 3rd Party libraries
- Installation Scripts

In essence, you should be able to go to a clean disk and checkout everything you need to do a build.

- **The Compile/Link/Tests Process Needs To Be Automated**

Whether you're using ANT or some UNIX tools, your build/test scripts need to be complete and automated.

- **You Must Run Tests For Validation After Each Build**

After compiling & Linking, the CI process should run a battery of unit/system tests to catch issues. A best practice is utilizing Test Driven Development (TDD) where engineers to create automated unit tests that define code requirements (immediately) before writing the code itself. And those tests are run upfront in the CI validation cycle.

- **Everyone Commits to the Mainline Every Day**

Developers who hold on to their changes too long are inviting non-trivial large merge events. It is difficult to manage the communication between engineers to resolve conflicts so it is best to not let them accumulate.

- **The Builds & Tests Should Be Done Exactly the Same way as the formal CM process (production environment)**

"It works on my machine" syndrome is common when there are inconsistencies in build/test methodologies. Making sure everyone is doing it the same way will help remove this issue.

- **The Builds (and tests) have To Be Fast**

Engineers will not utilize a CI build process if feedback is not immediate. They will tend to hold his/her code until ready to build all changes at once. The risk incurred may include a lack of incremental change sets making debugging (or backing out) code more difficult.

- **The Executables Should Be Easily Accessible**

Agile development depends on quick feedback. This may come from other developers, QA engineers, SE's, beta partners or numerous other sources. Making it easy for people to find and run your build output will provide the most amount of feedback to improve your quality and/or adjust your feature content.

- **You Should Include a Robust Report Dashboard**

It is important to have visibility into the status of your builds/tests. Results communication for each CI integration should be published and updated regularly. Doing so will help provide metrics for consumers of your build as well as Program Management data.

- **You Should Automate the Deployment**

Design a model which allows the movement of artifacts from development to QA to Production. This should include a method to be able to roll-back if issues are discovered.

Continuous Integration – Electric Commander Selected For Our Customer

There are several Continuous Integration software tools available on the market. For a comprehensive list including feature breakdown review the ThoughtWorks website:

http://confluence.public.thoughtworks.org/display/CC/CI+Feature+Matrix

www.spkaa.com
Ph: 888-310-4540

SPK and Associates
900 E Hamilton Ave, Ste.100
Campbell, CA 95008

In our last article we described how we brought the build turn-around down to 30 minutes (from 6 hours) utilizing Electric Accelerator. While this is not the most optimal for CI, it definitely opened the door for multiple builds/test runs throughout the day. Continue effort exists to make builds even faster.

Our CI tool of choice was Electric Commander. The reasons for our selection included:

- A tight integration with our make and DRM solution (Leveraging Electric Accelerator).

- The ability to grow with the organization. We had completed one product line and wish to extend the same tool to other areas. Electric Command offer good scalability allowing us to add teams and projects.

- Reusability. Electric Commander allowed us to create generic modular procedures which could then be leveraged by other products. i.e. checkout, build, test, deploy. Within each procedure there are steps with robust support for Perl scripts and XML Property Sheets.

- One Key Click Extensibility. We were able integrate not only the build and test processes but also extend the automation to 3rd party tools such as static analysis, code coverage and bug tracking systems. A user could execute the complete flow with one click.

- Virtualization – Electric Commander allowed us to provision on-demand resources via its integration with VMware.

Electric Commander is the software that runs your factor floor. It moves the software through your defined methodology and gathers key metrics to measure performance of both the software production processes and the actual results of your builds.

We liked that it was web based which is great for globally distributed development teams. We had folks in Moscow, Bangalore, Beijing, and the U.S. all having web-based access to the software production infrastructure and processes. That meant our developers (where ever they were) could use the same standardize methodology.

We liked that it is designed to work with existing tools and processes in our environment. Electric Commander moves the control of these tools out of the scripts that they were currently implemented in and into an interface so that they can be more easily identified and managed.

We liked that Electric Commander abstracted the actual machines that runs builds and tests. Folks no longer needed to be aware of specific build/test hardware. And this meant that different teams can use those resources for production without actually knowing the specifics of the infrastructure (the machine names, the IP addresses, the login credentials).

And lastly, we liked the fine grain security ensuring that the users only have access to relevant information.

SPK and Associates is a partner with Electric Cloud. Call us today to help define your Continuous Integration environment to improve your operations ROI.

Carlos Almeida
SPK and Associates
Architect, Software Engineering