Advance Features of Smart Build Tools

Evaluation Document

2/18/2010
Rajesh Kumar
eMail: rajesh@scmgalaxy.com

SMART BUILD - ADVANCED FEATURES

Table of Contents

Agile Development Challenges

Two of the core principles of agile development are: "deliver working code frequently" and "working software is the primary measure of progress." More and more teams, from ISVs to enterprise IT groups, are recognizing the quality and productivity benefits of integrating early and often. Whether you want to just build and test more frequently or implement a comprehensive agile process, you'll need to ensure that your build and test process is:

- fast
- automated (not requiring manual intervention)
- reliable (broken builds are the enemy of an agile approach)

Addressing the build-test-deploy process is one of the best first steps toward an agile development model.

Agile Development Challenge	Impact
Builds require manual intervention	Integrating often will overwhelm a manual build process
Slow build cycle (whether long individual builds or a large number of build targets)	Long builds limit the number of iterations possible in a day Builds longer than ~30 minutes rule out Continuous Integration
Developers introducing errors during integration or production builds	If developers can't do preflight builds and tests on all targets/platforms prior to check-in, Continuous Integration can turn into Continuously Broken Builds

Deployment Challenges

Deployment of a real world application is a complex, and all-too-often, a manual process. The complexity comes from the fact that the application may need to be deployed to any number of environments: Development (DEV), Quality Assurance (QA), User Acceptance Testing (UAT), Pre-production (STAGE), and Production (PROD). Each of these environments is typically slightly different requiring different accounts, password, and setting and thus it may seem like manual intervention is required.

Complicating matters is the fact that today's business applications comprise multiple moving parts that all need to be synchronized and deployed together. For example, a typical J2EE application can be made up of an EAR file, database changes, some static content, and environment changes (such as JDBC connection pools, Message Queues, etc.). Making sure that all these pieces get deployed on the correct machines and in the correct order can feel like a daunting task, especially if you need to repeat it for each environment. Keep in mind that each environment may have a different number of database servers, application server, HTTP server, etc.

Build Acceleration Challenges

Slow software build cycles are more than just an annoyance. Whether you have long monolithic builds or a large number of smaller builds that take a long time, they are having a real impact on your development organization.

- **Wasted time, frustration waiting for builds**
- **Fixing broken builds delays testing**
- **Fewer bugs fixed, fewer features added**
- **Developers avoid builds**
- **QA may skip tests to meet deadlines**

A recent survey of more than 350 development professionals found that 68% reported broken builds at least a few times per month (and often weekly), and fixing those typically wasted 2 weeks per year. The majority said their builds were each more than 1 hour long, with 24% reporting builds of 2-4 hours, which translates to time not spent creating great software.

Integration with Elastic and Cloud Computing

In a large infrastructure with many projects, the CI server often experiences variable task loads with unpredictable load peaks (such as during releases). In an environment with fixed resources, this leads to build queue growths, increased build times, and poor build scheduling control.

Computing clouds such as Amazon EC2 provide a nice and efficient way to scale resources to fulfill variable demands with random load spikes. Such rapid demand growths is exactly what happens in Build Server during CI "rush hours," when everyone starts personal builds before going home, and also during pre-release days, when every new fix needs another round of unit tests, and build feedback time becomes mission-critical.

Build Server utilizes EC2 via virtual build agents. Virtual build agents are similar to standard ones, except that they run on virtual instances on the Amazon EC2. This means that Build Server is able to dynamically start as many instances of such agents as needed, in order to keep the build queue under control under high loads. Additionally, Build Server can shut down virtual build agents when they are not needed anymore, to minimize EC2 instances uptime.

While being a great solution for mitigating peak loads and maintaining constant build feedback time, Amazon EC2 integration also provides great cost optimization opportunities, as each resource accumulates costs only while in use.

Workflow Management

In recent years, build management solutions have provided even more relief when it comes to automating the build process. Both commercial and open source solutions are available to perform more automated build and workflow processing. Some solutions focus on automating the pre and post steps around the calling of the build scripts, while others go beyond the pre and post build script processing and drive down into streamlining the actual compile and linker calls without much manual scripting.

Tools Comparison

	OpenMake	**Bamboo**	**AnthillPro**	**TeamCity**	**Electric Cloud**
Agile Support	YES	YES	YES	YES	YES
Deployment	YES	**NO**	YES	**No**	YES
Remote Agent	YES	YES	YES	YES	YES
Cloud - Elastic	**NO**	YES	**No**	YES	YES
WorkFlow Mgmt	YES	**YES - Limited**	YES	**YES - Limited**	**YES, Limited**

Reference:

http://www.electric-cloud.com

www.**anthillpro**.com

http://www.atlassian.com

www.jetbrains.com/**teamcity**

http://www.openmakesoftware.com