WIKIPEDIA

Build automation

Build automation is the process of automating the creation of a software build and the associated processes including: compiling computer source code into binary code, packaging binary code, and running automated tests.

Contents

Overview

Historically, build automation was accomplished through makefiles. Today, there are two general categories of tools:[1]

Build-automation utility
This includes utilities like Make, Rake, CMake, MSBuild, Ant, Maven or Gradle (Java) etc. Their primary purpose is to generate build artifacts through activities like compiling and linking source code.

Build-automation servers
These are general web based tools that execute build-automation utilities on a scheduled or triggered basis; a continuous integration server is a type of build-automation server.

Depending on the level of automation the following classification is possible:

- Makefile - level
  - Make-based tools
  - Non-Make-based tools
- Build script (or Makefile) generation tools
- Continuous-integration tools
- Configuration-management tools
- Meta-build tools or package managers
- Other

A software list for each can be found in list of build automation software.

Build-automation utilities

Build-automation utilities allow the automation of simple, repeatable tasks. When using the tool, it will calculate how to reach the goal by executing tasks in the correct, specific order and running each task. The two ways build tools differ are task-oriented vs. product-oriented. Task-oriented tools describe the dependency of networks in terms of a specific set task and product-oriented tools describe things in terms of the products they generate.[2]

Build-automation servers

Although build servers existed long before continuous-integration servers, they are generally synonymous with continuous-integration servers, however a build server may also be incorporated into an ARA tool or ALM tool.

Server types

- **On-demand automation** such as a user running a script at the command line
- **Scheduled automation** such as a continuous integration server running a nightly build
- **Triggered automation** such as a continuous integration server running a build on every commit to a version-control system.

Distributed build automation

Automation is achieved through the use of a compile farm for either distributed compilation or the execution of the utility step.[3] The distributed build process must have machine intelligence to understand the source-code dependencies to execute the distributed build.

Relationship to continuous delivery and continuous integration

Build automation is considered the first step in moving toward implementing a culture of continuous delivery and DevOps. Build automation combined with continuous integration, deployment, application-release automation, and many other processes help move an organization forward in establishing software-delivery best practices.[4]

Advantages

The advantages of build automation to software development projects include

- A necessary pre-condition for continuous integration and continuous testing
- Improve product quality
- Accelerate the compile and link processing
- Eliminate redundant tasks
- Minimize "bad builds"
- Eliminate dependencies on key personnel
- Have history of builds and releases in order to investigate issues
- Save time and money - because of the reasons listed above.[5]

See also

- Application-release automation (ARA)
- Continuous configuration automation (CCA)
- Continuous integration (CI)
- Continuous delivery (CD)
- Continuous testing
- DevOps
- List of build automation software
- Product family engineering
- Release engineering (RE)
- Software configuration management (SCM)
- Unit testing

References

1. Ceruzzi, Paul E. (2003). *A history of Modern computing* (https://archive.org/details/historyofmodernc00ceru_0). The MIT Press. ISBN 978-0262532037.
2. Clark, Mike (2004). *Pragmatic Project Automation: How to Build, Deploy, and Monitor Java Apps*. The Pragmatic Programmers. ISBN 978-0974514031.
3. Enos, Joe (2013). "Automated Builds: The Key to Consistency" (http://www.infoq.com/articles/Automated-Builds). *InfoQ*. C4Media Inc. Retrieved September 16, 2015.
4. Bashan, Shmuel; Bellagio, David E. (2011). *Work Item Management with IBM Rational ClearQuest and Jazz: A customization Guide*. IBM Press. ISBN 978-0137001798.
5. "Archived copy" (https://web.archive.org/web/20081123044304/http://www.denverjug.org/meetings/files/200410_automation.pdf) (PDF). Archived from the original (http://www.denverjug.org/meetings/files/200410_automation.pdf) (PDF) on 2008-11-23. Retrieved 2008-09-19.