[[1]](#footnote-1)

Continuous Integration: Processes and APIs (December 2016)

Kyle S. Healy

*Abstract*—This paper is to discuss continuous integration, it’s processes, tools, and APIs. Continuous integration is the process of continually developing, testing, running, and interweaving new features into software. The benefits and limitations of continuous integration will be analyzed in great detail. Continuous Integration in industry and the up and comings in academia will be described. After, we’ll look at the actual processes of continuous integration, how they were created, and the best practices currently. Continuous integration is in itself a process; Every process has tools and APIs. Some of industries current tools include Jenkins, Buildbot, Travis CI, Strider, Go, and Integrity. Each of these tools will be compared and contrasted to find the best and worst pieces about them and what they can bring to the table to help a software team develop faster, efficiently, and ultimately usable maintainable code.

*Index Terms*—Continuous Integration, Processes, APIs,

Sort these alphabetically

# INTRODUCTION

C

ontinuous integration is not a new concept to the software industry, but it is quickly being adopted by a lot of programs. Continuous integration is a concept to get software developers to consistently commit code to the central repository for building and testing. Amazon Web Service describes continuous integration as “a DevOps software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run” [1]. ThoughtWorks® states “Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early” [2]. As the multiple companies above described, continuous integration is becoming a key piece to the software development toolbox.

Continuous integration is a process with multiple different steps and APIs. Continuous Integration is used to help find bugs early and notify developers if the baseline has been broken. The process of how to set and use continuous integration in development will be discussed in detail. Continuous integration is already used in industry. There are many software companies that have already fully integrated continuous integration into their development cycle. Besides industry, academia has picked up continuous integration, and we will take a look into any new developments trending in academia as well as in industry. Next, a comparison to integration tools and APIs will be conducted. Each tool and API brings something different to the table. Having a better understanding of how these tools work and what they can and cannot do will be hugely beneficial.

# Continuous Integration: What it is?

Continuous integration is a key piece to software development cycles. Continuous integration is used much more in agile development cycles, but it is making its way into the waterfall method as well. Continuous integration is used to help software developers build and test their code before committing each and every time they commit. The build/test/commit cycle greatly increases the software developers’ effectiveness, because they can immediately know if the new code builds and passes all the current tests. This is helpful because each new commit has the possibility of breaking the current build. Most developers focus on adding a new piece of code or “feature” to the baseline. When they focus on just that feature, they may not understand what else they are changing, and having a continuous integration process that builds and tests the code on each commit will immediately let the software developer know the build is broken. This is ideal because than new code is not being built on top of code that has a bug in it.

## What is Continuous Integration

Continuous integration can be broken down into multiple steps: 1. Building new code changes, 2. Testing new changes, 3. Report results to the software developer, and 4. Committing new changes to baseline repository. These are the four basic steps of continuous integration, but this is just the high level steps. Figure 1 shows a diagram of the continuous integration cycle occurring. The diagram shows continuous integration as a cycle. A cycle is a good way to describe continuous integration because the cycle is continually iterating. Each step is extremely necessary and a building block to get to the next step. The diagram equates to our steps as follows. Step 1, building new code changes is ‘Initiate CI Process’ in the diagram. Step 2, Testing new changes is ‘Test’ in the diagram. Step 3, report results to the software developer, is ‘Report’ in the diagram. Lastly, Step 4, Committing new changes to the repository is ‘commit’ in the diagram. Now, we will go into each step in more detail.

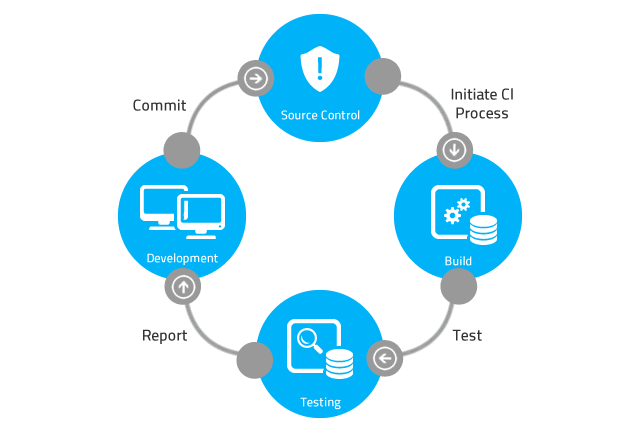


Fig. 1. Image of a basic continuous integration process [3].

### *Building new code changes*

### New code changes are the very beginning process of initiating the continuous integration process. Each developer is generally developing code separately for a different feature. This new code that the developer creates is continually locally built and tested by the developer on his or her own local machine. Once the developer is happy with the code changes, they will begin the continuous integration process with the build phase. This phase includes some changes to the developer side of the house as well. Developers must now get used to building code more consistently and committing it up to the server multiple times a day. The purpose of multiple commits a day is to keep the baseline the most up to date and make small changes. As the new smaller changes get created, they will be tested and errors can be caught earlier. Maintaining a working baseline is extremely important to continuous integration. Teams will continually make sure they This testing will be discussed next.

### *Testing new changes*

### The testing phase of continuous integration is a baseline of tests that continually run against the code to confirm the integrity of the baseline with each new commit. The baselined tests may continually change by either growing or decreasing. The baseline tests should cover all paths that the code may handle. This is not always realistic, but it is the goal of this phase. The purpose of the testing phase is be able to test each new build over a larger variety of tests to confirm the changes are not subsequently breaking a different feature within the baseline of code. New tests need to be added to handle the new features as well. If new tests are not added consistently with new features, the tests will not cover the whole baseline. If the whole baseline isn’t covered, the ability to continually confirm that new changes don’t break the baseline is compromised. This testing phase is crucial to the continuous integration process.

### *Report results to the software developer*

### After the continuous integration software builds and tests the baseline, the software will report the results back to the development team. These results are extremely beneficial to development teams as they rely on a baseline being working when they commit. The most common way this phase is handled is to only report when the baseline breaks. This means that when something builds incorrectly or a test case fails, the software developers find out, otherwise no action is taken. The reporting procedure can occur in multiple ways, most commonly, an email is sent to the entire development team. When a development team receives this email, it is a show stopper for the team. It doesn’t make sense to commit new changes to a broken baseline, so the baseline must be fixed before new efforts can be added. The entire team receives an email because the broken code or tests are a priority, so everyone must know to not commit new code on top of it and that it needs to be fixed immediately. Most tools do have a GUI or web interface to keep track of the builds and tests that occur over time. These tools help to see trends in the baseline to see if it is stable or continually breaking. These trends are extremely helpful to teams to be able to verify the robustness of the baseline.

4) Committing new changes to baseline repository

The last step in the continuous integration process is to commit the changes to the repository. The continuous integration process tries to be a blocker between the software developer and the configuration manager when it comes to bad code commits. The continuous integration software uses the building and testing phases head off problems of poor code being committed to the configuration management repository. Even if bad code gets committed to the repository, most continuous integration tools are very good at being able to point out directly where the issue occurred and what changes were made to cause the problem. All changes to the repository happen through the continuous integration process.

## What is the Purpose of Continuous Integration?

The purpose of continuous integration is to be able to have multiple developers successfully and efficiently be able to commit code multiple times a day, have it fully tested, and report any failures if they occur. The continuous integration process is extremely useful in finding new bugs and helping mitigate the introduction of errors from new code. Each new piece of code has the possibility of introducing bugs directly in the code written, or in subsequent code that relies on the code changes.

Continuous integration tries to automate a lot of the tedious processes that are “best practices” but in theory are rarely done in a development environment. An example of these “best practices” is regression testing on the whole baseline after each new change is made. Often local testing on that specific new feature is done by the developer, and perhaps they baseline that test. That is a decent situation than, but did that developer run a full regression test on the old features of the baseline before the commit? Most likely they did not. Continuous integration works to automate that process and consistently run those regression tests on a server before the new changes are committed. Taking that testing process and automating it on a server gives the developer the chance to move forward to the next tasking without having to spend time to regression test the baseline with each new feature. The developer gets an email if something breaks, than they can jump back to the previous code and find the issue that occurred. This compartmentalizing and automation help speed up the process of development and helps build and maintain a strong, robust baseline to be deployed rapidly.

In summary, the purpose of continuous integration is to maintain the integrity of a baseline through robust, automated building and testing of a baseline before committing the changes to the configuration manager. The major change in thought process comes from the developers. Developers must work on continually committing smaller changes more often. Being able to commit changes more often minimizes the differences between the developers’ stream and the configuration management stream. Smaller changes are easier to debug and find the problems that exist.

## Current Use Cases of Continuous Integration

Continuous Integration has many use cases in the software realm now. Continuous Integration can be seen in the software industry in the agile development cycle as well as the waterfall development cycle. It is always being taught now in an academic environment and not just learned on the job anymore. The benefits of continuous integration are very strong and making the field applicable to software developers of all kinds. Below are a few break downs of where continuous integration is currently being used. It is not a complete all-inclusive list, but it does touch on some of the key areas that are currently developing within the continuous integration framework.

### *Continuous Integration Uses in Industry*

### The continuous integration framework is currently being used throughout the software industry. Continuous integration has been the push for multiple companies to help keep a stable baseline that developers can continually build off of without fear of corrupting a whole system. The examples come from the defense industry. The defense industry has many companies in it. A few will be discussed below.

### Raytheon is a defense contractor that is currently using continuous integration on multiple programs. Raytheon engineers created a next generation Air and Missile Defense Radar (AMDR) [4]. Raytheon engineers have implemented continuous integration into their agile software environment. Mike Meservey, chief product owner for AMDR, said “In one case, we found a minor issue that we recognized as a very low probability event. It was a latent defect that in a traditional development environment might have gone undetected for months or years. But with our automated test suite and continuous integration, we identified and corrected it in a matter of days.” Raytheon is saying they are using continuous integration on a daily basis to help find bugs and fix problems in there baseline. They also have programs that have moved to an agile environment. An agile environment and continuous integration work well together and push productivity and efficiency to heighten levels.

### Northrop Grumman is another company that uses continuous integration in their development environment [5] [6]. Northrop Grumman has created a specific tool called Virtual System Integration Lab or VSIL [6] that couples nicely with continuous integration. The VSIL is an integration tool to let developers test code and find faults before moving to expensive hardware. The VSIL is a tool that provides capabilities to automate tests and the VSIL adds to the continuous integration process. Northrop Grumman is working with the US government to create a “continuous delivery and continuous integration pipeline” [5] to deploy their GEOINT services. The US government has a demand to implement continuous integration, and Northrop Grumman is helping to lead that process.

### *Continuous Integration Uses in Academia*

### Continuous integration is being used in industry a lot more today. It is being used so much that the colleges and universities are starting to actually teach continuous integration. Many free open courseware classes are also teaching continuous integration to help spread the knowledge to all software developers about its benefits. CloudAcademy [7] has an online course that teaches continuous integration. It describes continuous integration as “the first step towards a completely automated development, deployment and operations pipeline” [7]. Other online courses and on campus courses are beginning to teach continuous integration. The concepts behind continuous integration are necessary for all software developers. Learning them early can greatly increase the productivity and efficiency of a new software developer in the field.

### *System set ups*

### Continuous integration can be set up multiple different ways. One way continuous integration can be set up is through a server set up. This set up includes using a server to host the repository and run all the builds and tests. This set up maximizes the efficiency of the software developer because it takes the building and testing away from the programmer and puts it on a remote server. This is how most continuous integration systems are set up. If the server is starting to back up because too many jobs are coming in, the system can be upgraded to have slave nodes as well. These slave nodes can be tasked by the master server to run builds or tests concurrently, reducing the load on the master server. These parallel builds help to decrease time in between builds and bugs and errors can be discovered faster. Finding bugs and errors faster helps to produce better code that can be delivered whenever it is needed to be.

* Another way systems can be set up is to have the work be distributed between users PCs. This case is seen less often, but if a small team doesn’t have a server, it’s a good way to start up the continuous integration process. This process has been successfully seen in industry and works when necessary. It is less reliable because computers must be left on and connected to be able to run jobs when they are queued up, but it is better than not running any type of automated build and test suite.

## Benefits of Continuous Integration

The benefits of continuous integration include automated building, consistent testing, quick reporting and easy traceability when it comes to the baseline. There are more benefits to continuous integration as well, but discussing all of the benefits would take up massive amounts of space and time, so we will stick to the few stated above.

The automated building is a major benefit to continuous integration. The ability to build a baseline while continuing to develop new code is a great time saver. Software developers can spend more time thinking creatively to find solutions to problems that exist instead of spending time in idle while they wait for the baseline to build the changes over and over again. Each build is automated and run on servers or individual machines to minimize the impact to each individual developer at their work stations.

Consistent testing is a large benefit because it is how a software developer knows if they’re code works or not. It also tells the developer that the code they wrote or changes they made didn’t break another piece of code. Having some standard regression tests helps keep the baseline in a stable state. The tests are automatically run on the server after each new build. These auto-run tests help the developers keep their code up to date and working correctly. All software should have accompanying unit tests that are performed to verify the integrity of the baseline.

Quickly reporting build or test failures back to the development team helps to fix issues significantly earlier. A stable baseline is a happy baseline. Having a reporting system that informs the whole team if the baseline breaks is essential to maintaining the integrity of the baseline. Software developers learn to not commit to a broken baseline. Adding changes to a baseline that is not working only masks the problem if the new commit is anything but a fix. Reporting failures gives the software developer the confidence to continue onto the next feature without having to wait for every single build to complete to determine of the code’s going to work.

Tracing the problem, when it exists, can become a huge hassle. Having an integration tool that can tell you exactly where the build broke or test failed is useful. Tracing the problem helps the software developer spend less time debugging and spend more time creating new additional features. Also, continuous integration helps to enable greater communication by increasing visibility of problems as they occur and displaying those problems to the end user.

Continuous integration minimizes the painful process of the long intense integrations of new features when they are added to the baseline in any other framework. If each developer is continually committing to the baseline, the changes are small and minor. Changes can be easily traced and debugging takes significantly less time. Also, having a reliable baseline adds confidence to each software developer that they are building on a strong foundation. The same is true for a robust suite of unit, integration, and regression tests.

Continuous integration is extremely cheap. There is open source free software out there for almost every language when it comes to continuous integration. It is cost effective and integration often saves time. The longer between integrations, the better chance of not catching all the bugs and fixing the problems. Larger integrations also take longer, and time directly translates into money in most cases.

## Limitations of Continuous Integration

Continuous integration isn’t a catch all solution. The build needs to be automated before continuous integration can occur. Unit tests must exist and reliably pass before being integrated into the framework. New tests need to be created and added by the user to keep the test cases up to date and completely covering the baseline. If developers never update the test cases, the new code is not consistently tests and bugs may begin to fester inside that untested code.

A major limitation of continuous integration is the culture of the software team to adopt it. Continuous integration is not nearly as effective as it could be if all it does is build and test the code. The developer needs to make smaller commits more often on a regular basis. This is a major culture change for some software developers who are used to making large commits and integrating very infrequently. Continuous integration tools also build, test and report metrics back. If software teams are not doing anything with these metrics, (ie trying to improve them) the use of a continuous integration tool and process becomes minimal. The whole software team has to embrace the continuous integration process to reap the most benefits from the process.

# Continuous Integration Processes

The continuous integration process most often occurs as seen in figure 2, although this is not the only set up for it. Usually, there are multiple developers that are all developing locally on their own PCs. Multiple times a day, preferably, they would commit those changes to the repository. The source control server than gets polled by the continuous integration server. The continuous integration server pulls in the new baseline and builds it. If there is a failure, the continuous integration server reports that failure back to the developers. If the build is successful, it begins unit tests on the new baseline. Again, if any tests fail, the continuous integration server reports a failure, which test failed, where it failed etc. back to the developer. After the unit tests all pass, the continuous integration server will attempt to run any integration tests and regression tests against the new baseline. After this point, the developers are notified regardless of a success or failure to complete. This process can be seen in figure 2.

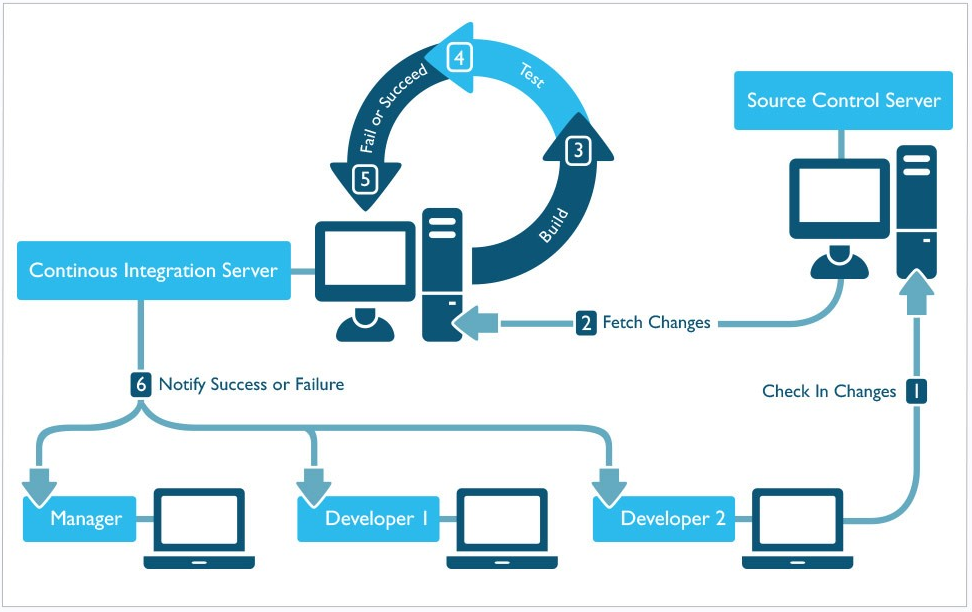


Fig 2: Continuous Integration Process [8].

# Continuous Integration Tools and APIs

There are many tools and APIs out there for continuous integration. Each tool provides a different interface and hook into the code. Each separate tool may do things slightly differently, but they all provide the continuous integration process as described above. Some tools may seem superior in certain situations, but all the tools have served a purpose and may be better in different scenarios.

Tools, such as Jenkins, Buildbot, Travis CI, Strider, Go, and Integrity, will be described and analyzed here. These tools all provide different benefits to their customers, but they also all have different limitations as well. Next we’ll take some time to explore these benefits and limitations of each tool in greater detail.

## Jenkins

Jenkins is an open source automation server that provides continuous integration solutions to its customers. Jenkins is written in Java. Jenkin’s was originally developed under the Hudson [10]. Jenkins helps developers by automating as much of the non-human part as possible. Jenkins has expanded on the continuous integration framework and now enables the development teams to implement continuous delivery. Continuous delivery is the next step from continuous integration. Continuous delivery allows software teams to build, test, and stage their software for release much faster and more often. Continuous delivery is the next step after continuous integration, but it is outside the scope of this paper.

Jenkins is cross-platform and free to download. It runs as a server-based system in a servlet container (Ex. Apache Tomcat). Jenkins supports many software configuration management tools. These software configuration management tools that Jenkins supports include AccuRev, CVS, Subversion, Git, Mercurial, Perforce, Clearcase, and RTC. Jenkins can execute shell scripts, Windows batch commands, Apache Ant and Apache Maven based projects.

Jenkins uses software configuration management tools to help it automate the build process. Jenkins can be triggered by various means. One way to trigger a Jenkins build is to commit a version to the control system. Jenkins will use a scheduling tool to poll the source control server. This polling of the source control server kicks off a Jenkins integration build, and the whole process begins.

Jenkins is set up as a continuous integration server. It polls the source control server for new changes at a set rate. Whenever a new change is found it fetches the changes and begins a full build on the new baseline. If the build passes, it begins to unit test this new build. After the unit tests pass, Jenkins will notify the developers that it successfully passed the test for the new baseline. If anything goes wrong, either in the build or the unit tests, Jenkins will immediately notify the developers as well. That is the basic life cycle of a regular commit inside Jenkins.

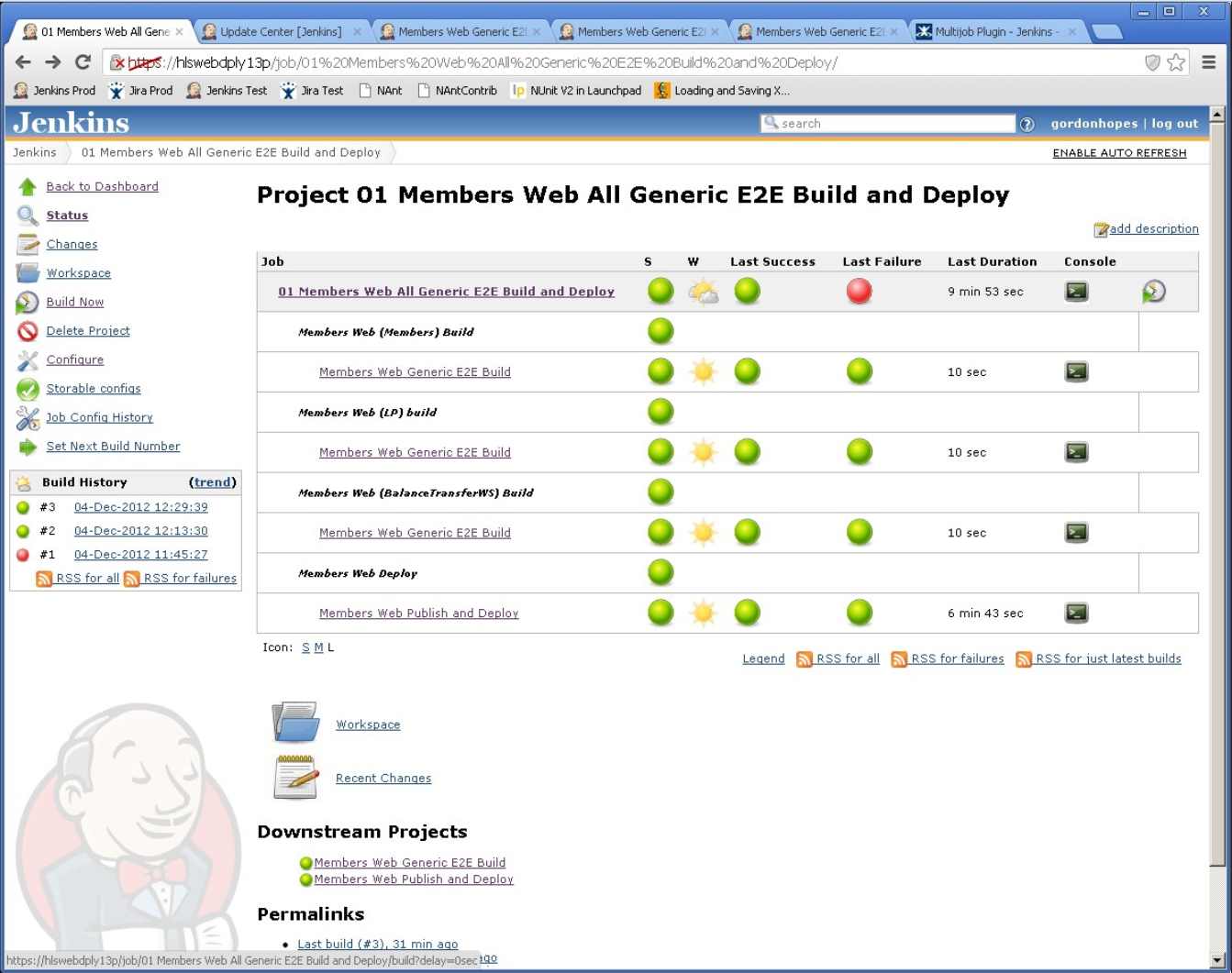


Fig. 3: Screen shot of a Jenkins home page [11]

Jenkins has the ability to run nightly builds with more in depth testing. Jenkins can run at a set time each day a full build and full test suite that doesn’t get run on every commit. These nightly builds run longer, more in depth tests on the code baseline. These tests may incorporate the whole system and stand up more pieces than the unit tests do. The purpose of the nightly builds is to take away the long testing process at each build so the build and test after individual commits is sped up. At the same time, it provides a daily full suite of tests that are run after hours to test the integrity of the baseline with the new changes that day.

On top of the underlying functionality that Jenkins provides, it also provides a GUI that is easy to use and intuitive. The GUI is updated in real time against the building projects and gives a lot of feedback to the development team. The GUI shows if the baseline is broken or intact. It also shows historical data of the baseline. It keeps track of which builds passed and which builds failed. This is extremely helpful to verify that the baseline is robust enough and the build has been stable for a while. The longer a build is stable the more reliable it is, and developers can trust it. Jenkins’ GUI also tracks which tests are run, passed, failed, and skipped. If an error occurs, Jenkins makes it easy to drill down into the broken project and find exactly where the error occurred. This speeds up debugging and helps developers find problems quicker. Jenkins runs tests consistently, automating the process so developers no longer have to run each test.

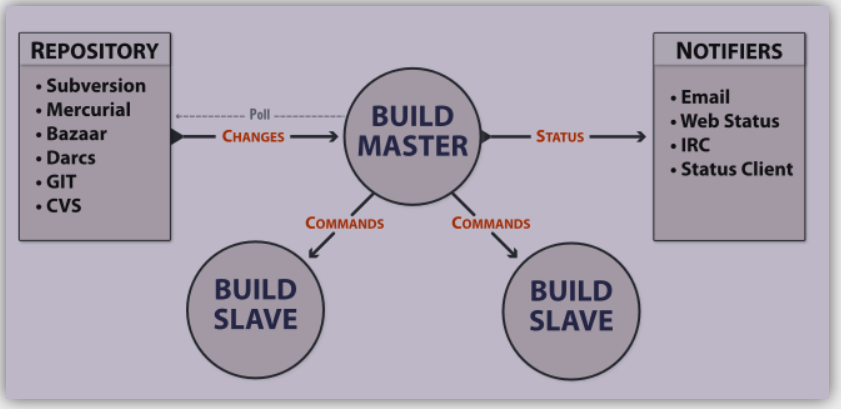


Fig. 5: Buildbot basic set up [15]

Jenkins has a lot of plugins that can be utilized to help improve a software development teams’ experience. Plugins are created by the Jenkins community and open to all. There are over 1000 plugins currently [9]. Some plugins include “Mailer” which allows the development team to configure email notifications regarding the build results. This is an extremely helpful tool when the team needs updates on who the build is doing. Other plug ins include Junit, Javadoc, Ant, SSH Slaves. Each plugin brings a different capability to the Jenkins framework and helps each user be able to customize Jenkins to their specific needs.

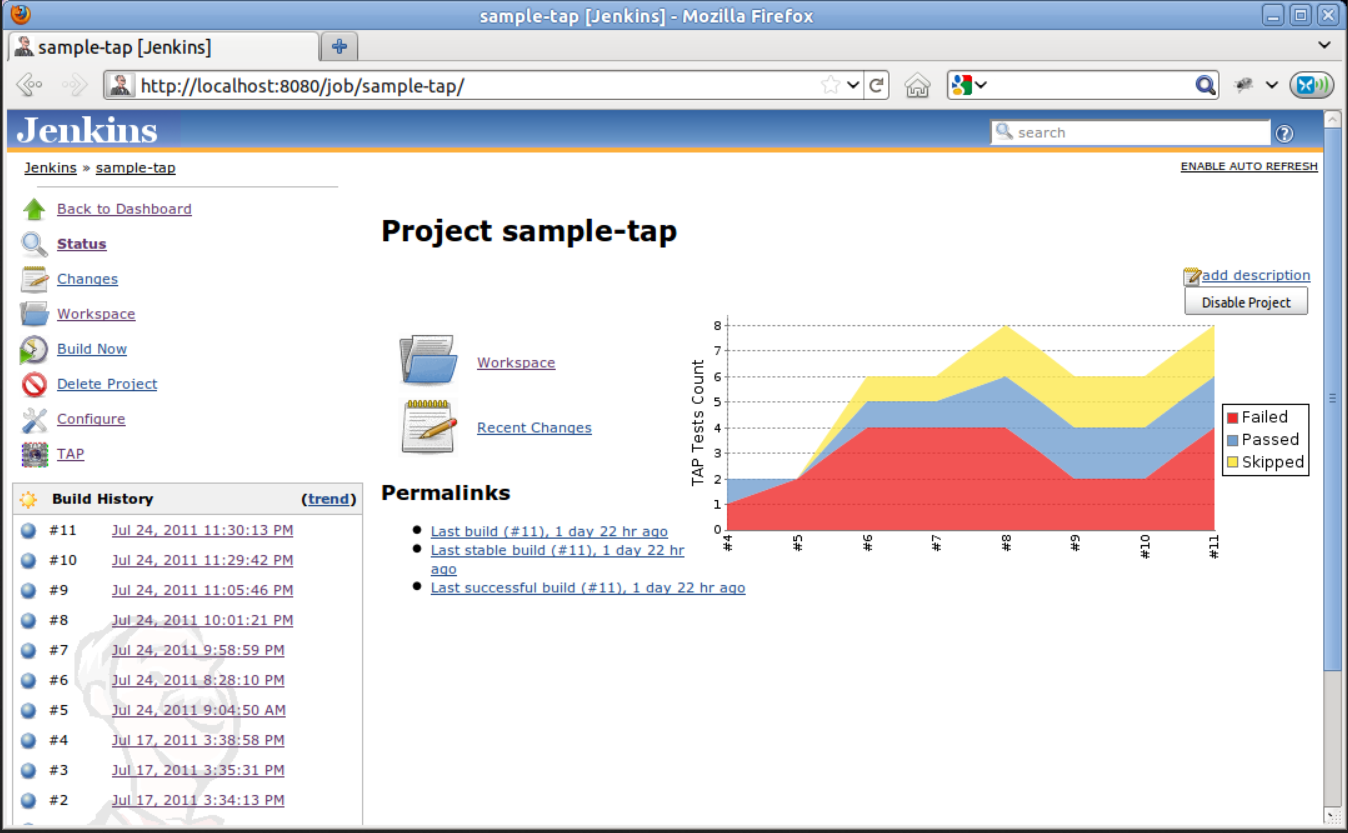


Fig. 4: Screen shot of a Jenkins project page [12]

Jenkins makes it possible to be able to develop in most main stream languages. Current supported languages by Jenkins include Java, C/C++, Python, and Ruby. Support for more languages exists. Jenkins supports other tools, such as Android Development, GitHub, Docker and much more. Each of these tools adds to the appeal of Jenkins and helps software development teams become more open to adopting Jenkins as a continuous integration tool.

## Buildbot

Buildbot is a software continuous integration tool that helps to automate the compile/build and test cycle required in each individual build. Each individual build and test cycle is used to validate the baseline of the project. Buildbot is written in python and was initially released April 29, 2003 [13].

Buildbot support multiple source control management tools. These tools include CVS, Bazaar, Darcs, SVN, Perforce, Mercurial, Git, Montotone, Repo, and BitKeeper [14].

Buildbot describes their basic functionality as follows, “ Buildbot supports distributed, parallel execution of jobs across multiple platforms, flexible integration with version-control systems, extensive status reporting, and more.” Buildbot is first and foremost a job scheduling system. It’s primary purpose is to manage resources and schedule jobs for execution when resources become available. It also reports the results of these jobs after they run.

Buildbot can have one or more masters and a collection of slave nodes. The build masters do most of the managing and coordinating. The build master polls the repository for changes and manages the resources of the build slave nodes. The build master also sends statuses back to the development team as necessary. These statuses are sent by email, web status, IRC or Status Client. The build master commands the build slaves. Each build slave can be run on a variety of operating systems to help fully test code compatibility with every operating system.

Buildbot is highly configurable. The Buildbot build master uses a python configuration script to configure the whole system. Because this script is in python, it can either be a very simple configuration file or it can take full advantage of all the libraries and built-in components python has available. The build slaves are also highly configurable. The build slaves are controlled by the build master. The build slaves are the work horses of the Buildbot system. They build the baseline, run it, and test it. They can do this in multiple operation systems and in multiple languages, depending on how they are configured. Buildbot is a continuous integration tool that is highly configurable and manages its resources extremely well.

## Travis CI

Microcontroller Assembly Code is so tightly coupled with the microcontroller

detail.

## Strider

Microcontroller Assembly Code is so tightly coupled with the microcontroller

## Go

Microcontroller Assembly Code is so tightly coupled with the microcontroller

## Integrity

Microcontroller Assembly Code is so tightly coupled with the microcontroller

Model checking for microcontroller assembly code are a large effort up front with some big pay offs. All errors inside the development stage can be detected. Instructions are easy to handle, clean and well documented. It is hardware dependent and operates as close to the actual execution as possible.

# Continuous Integration Processes

Information systems are large complex systems that require validation of specifications. The paper *Comparison of Model Checking Tools for Information Systems,* [Chane-Yack-Fa

Next, a deeper analysis of the major model checkers and the languages behind them.

## Concepts of New Knowledge Presented - Major Model Checkers

Model checkers fall into four major classes. First, the explicit state model checkers use “an explicit representation of the transition system associated with the model specification of the system to extend out to maximal lengths. Lastly, constraint model checkers verify formula using logic programming. Some of these logic languages include support for property specification.

## Model Checking Tools

Multiple model checking tools are presented to handle the four different model checkers presented above. A few of the model checkers that will be discussed are Spin, NuSMV, and organizes it’s B specifications into abstract machines. ProB’s operations are specified in the Generalized Substitution Language, a generalization of Dijkstra’s guarded command notation.

## Classes of Properties for Information System

There are four main classes of properties identified with information systems. These four properties include, sufficient. CTL makes SCEF easy to specify in NuSMV, and ProB. SCEF is not possible in Spin because Spin does not support LTL. Lastly, INV is handled in all model checkers without any significant problem.

# Proposed Application of the Concepts Presented

Databases, and affects the next sentence’s meaning. Lastly, Pragmatic Analysis is the process of re-interpretting what was said vs what was meant. This is similar to a requirement of the system having real world knowledge.

# Conclusion

## Embedded software verification and model checking for growth and increase in technology, the servers, algorithms, and computing power have caused major increases in complexity of the system. Again with information systems, a bad outcome could be a life or death situation.

References

1. Amazon Web Service (2016). What is Continuous Integration?. [Online]. Available by web page. <https://aws.amazon.com/develops/continuous-integration/>
2. ThoughtWorks (2016). “Continuous Integration” [Online]. Available by web page. <https://www.thoughtworks.com/continuous-integration>
3. Khan, Aman (August 14, 2015). The Existing challenges of Continuous Integration (CI). [Online]. Available by web page. <http://www.360logica.com/blog/2015/08/the-existing-challenges-of-continuous-integration-ci.html>
4. Raytheon Corporation (2014). Engineering on the Fast Track. Raytheon’s Agile Engineering Creates Next-Gen Radar [Online]. Available by web page. <http://www.raytheon.com/news/feature/amdr_agile.html>
5. Northrop Grumman (2016). GEOINT Symposium. [Online]. Available by web page. <http://www.northropgrumman.com/MediaResources/MediaKits/GEOINT/Pages/default.aspx>
6. Northrop Grumman (2016). Virtual System Integration Lab (VSIL). [Online]. Available by web page. <http://www.northropgrumman.com/Capabilities/VSIL/Pages/default.aspx>
7. CloudAcademy (2016). Introduction to Continuous Integration. [Online]. Available by web page. <https://cloudacademy.com/cloud-computing/introduction-to-continuous-integration-course/>
8. Cois, C. Aaron (2015, January). Continuous Integration in DevOps *Software Engineering Institute Carnegie Mellon University* [Online]. Available by web page. <https://insights.sei.cmu.edu/devops/2015/01/continuous-integration-in-devops-1.html>
9. Jenkins (2016). Jenkins “Build great things at any scale.” [Online]. Available by web page. <https://jenkins.io/>
10. Kawaguchi, Kohsuke. “Hudson” (PDF). <https://www.java.net//blog/kohsuke/archive/20070514/Hudson$20J1.pdf>
11. Wsource (2016). Jenkins Screen Shot. [Online]. Available by web page. <https://issues.jenkins-ci.org/secure/attachment/22920/screenshot-1.jpg>
12. Kinoshita, Bruno (2016, Oct). TAP Plugin. [Online]. Available by web page. <https://wiki.jenkins-ci.org/display/JENKINS/TAP+Plugin>
13. Sourceforge (2005, September). Buildbot Release 0.3.1 (29 Apr 2003). [Online]. Available by web page. <http://sourceforge.net/project/shownotes.php?release_id=155929>
14. Mitchell, Dustin J. (2014 June). Builtbot / MAINTAINERS.txt [Online[. Available by web page or git pull. <https://github.com/buildbot/buildbot/blob/v0.8.7p1/MAINTAINERS.txt>
15. Buldbot (2016). Buildbot The Continuous Integration Framework. [Online]. Available by web page. <http://buildbot.net/index.html>

1. This work was supported solely by the author. It was used for a Research in Computer Science Course at Lewis University.

   Kyle S. Healy is working this paper for Lewis University, though he works full time for Northrop Grumman Corporation, Rolling Meadows, IL 60008, USA (email: healykys@gmail.com). [↑](#footnote-ref-1)