**Software Build** in simpler term is an activity to translate the human-readable source code into the efficient executable program.

Basically, **Build**is the process of creating the application program for a software release, by taking all the relevant source code files and compiling them and then creating a build artifacts, such as binaries or executable program, etc.You can also say that the build process is a combination of several activities which varies for each programming language and for each operating system but please remember the **basic**concepts are universal.

Now, lets consider an organization where the project team got a detail about the work which they have to deliver after sometime.Ex: The Project manager discusses the requirement with their dev team lead(s), the leads then decide whom they need to assign different module so that whole program can be built in sufficient time. Once developers are assigned the task to write a code for the given requirement, all the assigned developers will write the small modules of the program. As soon they finish writing the code and run the tests on their local machine, they can either integrate the code with main version control repository so that all the work done by all the developers can be build in single build or they try to build the code on his/her own machine to test their module functionality.

**What is Build Process?**The **build process** consists of following:

* Fetching the code from the source control repository
* Compile the code and check the dependencies/modules
* Run the automated unit tests
* Link the libraries, code, files, etc accordingly.
* Once successfully passed, Build the artifacts and store them.
* Archive the build logs.
* Send the notification mails.
* Ideally there are two basic types of builds.

### Full Build

* A **full build**which performs a build from scratch. It treats all resources in a project as if they have never been seen by the build server/tool.
* Afull build always starts from empty work area; its taking a full project as a input, figures out the dependencies, compiles all the source files in your project and builds all the parts in order accordingly, and then assembles them into the build artifact as a final output.

### Incremental Build

* An **incremental build** which uses a “last build state,” maintained internally by the build server/tool, to do an optimized build based on the changes in the project since the last build.
* An incremental build checks and compares every source file, as well as anything else that depends on the target, if any dependency has been modified after the target was last built, the target will be rebuilt, otherwise the file from the previous build would be reused.
* Since incremental builds only rebuild what needs to be, they are usually much faster than full builds and use fewer resources.

**Build automation** is the process of automating the creation of a [software build](https://www.revolvy.com/main/index.php?s=Software+build&item_type=topic) and the associated processes including: [compiling](https://www.revolvy.com/main/index.php?s=Compiling&item_type=topic) computer [source code](https://www.revolvy.com/main/index.php?s=Source+code&item_type=topic) into [binary code](https://www.revolvy.com/main/index.php?s=Binary+code&item_type=topic), [packaging](https://www.revolvy.com/main/index.php?s=Software+package+(installation)&item_type=topic) [binary code](https://www.revolvy.com/main/index.php?s=Binary+code&item_type=topic), and running [automated tests](https://www.revolvy.com/main/index.php?s=Test+automation&item_type=topic).

**Overview**

Historically, build automation was accomplished through [makefiles](https://www.revolvy.com/main/index.php?s=Makefile&item_type=topic" \o "Makefile" \t "_blank). Today, there are two general categories of tools:[1]

**Build automation utility (like Make, Rake, Cake, MS build,**[**Ant**](https://www.revolvy.com/main/index.php?s=Apache+Ant&item_type=topic)**, [Gradle](https://www.revolvy.com/main/index.php?s=Gradle&item_type=topic" \o "Gradle" \t "_blank) etc.)**

Whose 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](https://www.revolvy.com/main/index.php?s=Continuous+integration&item_type=topic) 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](https://www.revolvy.com/main/index.php?s=Continuous+integration&item_type=topic) tools
* [Configuration management](https://www.revolvy.com/main/index.php?s=Configuration+management&item_type=topic) tools
* Meta-build tools or [package managers](https://www.revolvy.com/main/index.php?s=Package+manager&item_type=topic)
* Other

A software list for each can be found in [list of build automation software](https://www.revolvy.com/main/index.php?s=List+of+build+automation+software&item_type=topic).

**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 orient 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](https://www.revolvy.com/main/index.php?s=Build+server&item_type=topic) may also be incorporated into an [ARA](https://www.revolvy.com/main/index.php?s=Application+release+automation&item_type=topic) tool or [ALM](https://www.revolvy.com/main/index.php?s=Application+lifecycle+management&item_type=topic) tool.

**Server types**

* **On-demand automation** such as a user running a [script](https://www.revolvy.com/main/index.php?s=Scripting+language&item_type=topic) at the [command line](https://www.revolvy.com/main/index.php?s=Command+line&item_type=topic)
* **Scheduled automation** such as a [continuous integration](https://www.revolvy.com/main/index.php?s=Continuous+integration&item_type=topic) server running a nightly build
* **Triggered automation** such as a continuous integration server running a build on every commit to a [version control](https://www.revolvy.com/main/index.php?s=Version+control&item_type=topic) 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](https://www.revolvy.com/main/index.php?s=Continuous+Delivery&item_type=topic) and [DevOps](https://www.revolvy.com/main/index.php?s=DevOps&item_type=topic" \t "_blank" \o "DevOps). Build automation combined with [Continuous Integration](https://www.revolvy.com/main/index.php?s=Continuous+Integration&item_type=topic), [deployment](https://www.revolvy.com/main/index.php?s=Software+deployment&item_type=topic), [application release automation](https://www.revolvy.com/main/index.php?s=Application+release+automation&item_type=topic), 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](https://www.revolvy.com/main/index.php?s=Continuous+testing&item_type=topic)
* 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]

**Build Tools** allows the automation of repeatable tasks by executing all the tasks in the correct, specific order and running each task accordingly. A build tool is used for building a new version of a program with all the specified requirements in correct manner.There are plenty of tools available in the market, starting with oldest one “make”, then “ant” & “nant”, “msbuild”, followed by “maven”, “ivy”, now a days “gradle” is in boom and so on.Each tool has its own limitations, approaches, process, methods, mechanism, support, pros & cons.

In early days, people were using an Unix uitility “**make**” which was an open source utility. **Make** uses the build file named as “Makefiles” which automatically builds the executable programs and libraries from source code. It also specifies how to derive the target program.

Then many other build utilities emerged in the market which actually were derived from “make”.The revolution in build automation picked up a high speed when Apache introduced their open source build tool for java based application named as “**ant**“.  **Ant** supplies a number of built-in tasks allowing to compile, assemble, test and run Java applications.Ant uses XML as a build file in which user have to specify and write the order in which all the tasks are performed with desired outcome, by default the XML file is named as build.xml.Ant is better for controlling of build process and is a **perfect** fit for straight forward projects. It has very simple structure thus allowing anyone to start using it without any special preparation and knowledge. Apache has also introduced the open source build tool for .Net application named as “**Nant**” (Not ant) an year later.  **Nant** is similar to Apache Ant, but targeted the .NET development instead of Java.

Apache then made and release their first build tool which can resolve the project dependencies named as “**maven**“. **Maven** was introduced to improve some of the problems developers were facing when using Ant. Maven also uses XML as the format to write build specification with different structure all together and uses project object model (POM) file as a build file. Maven comes with pre-defined targets for performing the defined tasks such as compilation of code, linking, packaging, etc.To resolve the project dependencies, Maven downloads the java libraries and its plug-ins from one or more repositories such as the Maven 2 Central Repository, search.maven.org (you can configure in the settings.xml file from where you need to fetch the dependencies) and stores them in a local cache.This local cache of downloaded artifacts can also be updated with artifacts created by local projects and with the public repositories.**Maven** is not just a build software – it can **assist** with testing, produce reports on projects, run web applications and number of other tasks provided by plug-ins. Apache has also introduced “**ivy**“, a transitive dependency manager which is a part of apache ant. Together Ant & **Ivy** become much more powerful where apache ant do its predefined job and ivy helps with the dependency resolution.It works in same fashion as maven to resolve the dependencies but its role is very limited to resolve the dependencies and publishing of artifacts.Microsoft was not behind in the race and introduced their build tool named as “**msbuild**“. The Microsoft Build Engine (**MSBuild**) is the build platform for Microsoft and Visual Studio. msbuild provides an XML schema for a project file that controls how the build platform processes and build the software. **Gradle** is newest among all and an open source build utility, which works on the concepts and uses good parts of both tools Apache Ant & Maven. Gradle provides more flexibility than Maven, but is easier to write and use than Ant. Its not using the traditional XML and instead introduces its own DSL based on Groovy (one of JVM languages).Gradle’s ability to manage dependencies is not just limited to the external libraries.As your project grows in size and complexity, you definitely need to organize the code into modules.Gradle provides powerful support for defining and organizing multiproject builds, as well as modeling dependencies among the projects.The best thing about **Gradle** is that it combines all the best features from other build tools in a single tool.To **conclude**, Its purely a project, scm & developers call which tool they would like use in their project to build it.

**Make-based**

* [GNU make](https://www.revolvy.com/main/index.php?s=Make+(software)&item_type=topic), a widely used make implementation with a large set of extensions
* [make](https://www.revolvy.com/main/index.php?s=Make+(software)&item_type=topic), a classic Unix build tool
* [mk](https://www.revolvy.com/main/index.php?s=Mk+(software)&item_type=topic), developed originally for [Version 10 Unix](https://www.revolvy.com/main/index.php?s=Version+10+Unix&item_type=topic) and [Plan 9](https://www.revolvy.com/main/index.php?s=Plan+9+from+Bell+Labs&item_type=topic), and ported to Unix as part of [plan9port](https://www.revolvy.com/main/index.php?s=Plan+9+from+User+Space&item_type=topic)
* [MPW Make](https://www.revolvy.com/main/index.php?s=Macintosh+Programmer%27s+Workshop&item_type=topic), developed for the [classic Mac OS](https://www.revolvy.com/main/index.php?s=Classic+Mac+OS&item_type=topic) and similar to but not compatible with Unix make; the modern [macOS](https://www.revolvy.com/main/index.php?s=MacOS&item_type=topic" \o "MacOS" \t "_blank) (OS X) comes with both GNU make and BSD make; available as part of Macintosh Programmer's Workshop as a free, unsupported download from Apple
* [nmake](https://www.revolvy.com/main/index.php?s=Nmake&item_type=topic)
* [PVCS](https://www.revolvy.com/main/index.php?s=PVCS&item_type=topic)-make, basically follows the concept of [make](https://www.revolvy.com/main/index.php?s=Make+(software)&item_type=topic) but with a noticeable set of unique syntax features

**Make-incompatible -** [Apache Ant](https://www.revolvy.com/main/index.php?s=Apache+Ant&item_type=topic), popular for [Java](https://www.revolvy.com/main/index.php?s=Java+(software+platform)&item_type=topic) platform development and uses an [XML](https://www.revolvy.com/main/index.php?s=XML&item_type=topic) file format

* [Apache Buildr](https://www.revolvy.com/main/index.php?s=Apache+Buildr&item_type=topic), open-source build system, [Rake](https://www.revolvy.com/main/index.php?s=Rake+(software)&item_type=topic)-based, gives full power of scripting in [Ruby](https://www.revolvy.com/main/index.php?s=Ruby+(programming+language)&item_type=topic) with integral support for most abilities wanted in a build system
* [Apache Maven](https://www.revolvy.com/main/index.php?s=Apache+Maven&item_type=topic), a Java platform tool for dependency management and automated software build
* [ASDF](https://www.revolvy.com/main/index.php?s=Another+System+Definition+Facility&item_type=topic) LISP buildsystem for building LISP projects
* [A-A-P](https://www.revolvy.com/main/index.php?s=A-A-P&item_type=topic), a [Python](https://www.revolvy.com/main/index.php?s=Python+(programming+language)&item_type=topic)-based build tool
* [Bazel](https://www.revolvy.com/main/index.php?s=Bazel+(software)&item_type=topic), Google's own build tool, now publicly available in Beta.
* [BitBake](https://www.revolvy.com/main/index.php?s=BitBake&item_type=topic), a Python-based tool with the special focus of distributions and packages for embedded Linux cross compilation
* [Boot](https://www.revolvy.com/main/index.php?s=Boot+(software)&item_type=topic), a [Java](https://www.revolvy.com/main/index.php?s=Java+(software+platform)&item_type=topic) build and dependency management tool written in [Clojure](https://www.revolvy.com/main/index.php?s=Clojure&item_type=topic" \o "Clojure" \t "_blank)
* [Buck](https://www.revolvy.com/main/index.php?s=Buck+(software)&item_type=topic), a build system developed and used by Facebook
* [Buildout](https://www.revolvy.com/main/index.php?s=Buildout&item_type=topic), a Python-based build system for creating, assembling and deploying applications from multiple parts
* [Cabal](https://www.revolvy.com/main/index.php?s=Cabal+(software)&item_type=topic), common architecture for building applications and libraries in the programming language [Haskell](https://www.revolvy.com/main/index.php?s=Haskell+(programming+language)&item_type=topic)
* [FinalBuilder](https://www.revolvy.com/main/index.php?s=FinalBuilder&item_type=topic), for Windows software developers. FinalBuilder provides a graphical IDE to both create and run build projects in a single application. Final builder also includes the ability the execute unit test, deploy web projects or install and test applications.
* [Flowtracer](https://www.revolvy.com/main/index.php?s=Flowtracer&item_type=topic)
* [Gradle](https://www.revolvy.com/main/index.php?s=Gradle&item_type=topic), an open-source build and automation system with an [Apache Groovy](https://www.revolvy.com/main/index.php?s=Groovy+(programming+language)&item_type=topic)-based [domain specific language](https://www.revolvy.com/main/index.php?s=Domain+specific+language&item_type=topic) (DSL), combining features of [Apache Ant](https://www.revolvy.com/main/index.php?s=Apache+Ant&item_type=topic) and [Apache Maven](https://www.revolvy.com/main/index.php?s=Apache+Maven&item_type=topic) with additional features like a reliable incremental build
* [Grunt](https://www.revolvy.com/main/index.php?s=Grunt+(software)&item_type=topic), a build tool for front-end web development
* [Gulp](https://www.revolvy.com/main/index.php?s=Gulp.js&item_type=topic), another build tool for front-end
* [IncrediBuild](https://www.revolvy.com/main/index.php?s=IncrediBuild&item_type=topic)
* [Leiningen](https://www.revolvy.com/main/index.php?s=Leiningen+(software)&item_type=topic), a tool providing commonly performed tasks in Clojure projects, including build automation
* [MSBuild](https://www.revolvy.com/main/index.php?s=MSBuild&item_type=topic), the [Microsoft](https://www.revolvy.com/main/index.php?s=Microsoft&item_type=topic) build engine
* [NAnt](https://www.revolvy.com/main/index.php?s=NAnt&item_type=topic), a tool similar to Ant for the [.NET Framework](https://www.revolvy.com/main/index.php?s=.NET+Framework&item_type=topic)
* [Ninja](https://www.revolvy.com/main/index.php?s=Ninja+(build+system)&item_type=topic), a small build system focused on speed by using build scripts generated by higher-level build systems
* [Perforce Jam](https://www.revolvy.com/main/index.php?s=Perforce+Jam&item_type=topic), a generally enhanced, ground-up tool which is similar to Make
* [Psake](https://www.revolvy.com/main/index.php?s=Psake&item_type=topic), domain-specific language and build-automation tool written in PowerShell
* [Qt Build System](https://www.revolvy.com/main/index.php?s=Qbs+(build+tool)&item_type=topic)
* [Rake](https://www.revolvy.com/main/index.php?s=Rake+(software)&item_type=topic), a [Ruby](https://www.revolvy.com/main/index.php?s=Ruby+(programming+language)&item_type=topic)-based build tool
* [sbt (Scala Simple Build Tool)](https://www.revolvy.com/main/index.php?s=SBT+(software)&item_type=topic), a build tool built on a [Scala](https://www.revolvy.com/main/index.php?s=Scala+(programming+language)&item_type=topic" \o "Scala (programming language)" \t "_blank)-based DSL
* [SCons](https://www.revolvy.com/main/index.php?s=SCons&item_type=topic), Python-based, with integrated functionality similar to [autoconf](https://www.revolvy.com/main/index.php?s=Autoconf&item_type=topic" \o "Autoconf" \t "_blank)/[automake](https://www.revolvy.com/main/index.php?s=Automake&item_type=topic" \o "Automake" \t "_blank)
* [Stack](https://www.revolvy.com/main/index.php?s=Stack+(Haskell)&item_type=topic), a tool to build Haskell projects, manage their dependencies (compilers and libraries), and for testing and benchmarking.
* [Tweaker](https://www.revolvy.com/main/index.php?s=Tweaker+(build+tool)&item_type=topic), allowing task definitions to be written in any languages (or intermixed languages) while providing a consistent interface for them all
* [Visual Build](https://www.revolvy.com/main/index.php?s=Visual+Build&item_type=topic), a [graphical user interface](https://www.revolvy.com/main/index.php?s=Graphical+user+interface&item_type=topic) software for software builds
* [Waf](https://www.revolvy.com/main/index.php?s=Waf&item_type=topic) is a Python-based tool for configuring, compiling and installing applications. It is a replacement for other tools such as [Autotools](https://www.revolvy.com/main/index.php?s=Autotools&item_type=topic" \o "Autotools" \t "_blank), [Scons](https://www.revolvy.com/main/index.php?s=Scons&item_type=topic" \o "Scons" \t "_blank), [CMake](https://www.revolvy.com/main/index.php?s=CMake&item_type=topic" \o "CMake" \t "_blank) or Ant

**Build script generation -**These generator tools do not build directly, but rather generate files to be used by a native build tool.

* [BuildAMation](https://www.revolvy.com/main/index.php?s=BuildAMation&item_type=topic), a multi-platform tool, using a declarative syntax in C# scripts, that builds C/C++ code in a terminal using multiple threads, or generates project files for Microsoft Visual Studio, Xcode or MakeFiles.
* [configure](https://www.revolvy.com/main/index.php?s=Configure+script&item_type=topic)
* [CMake](https://www.revolvy.com/main/index.php?s=CMake&item_type=topic) generates files for various build tools, such as [make](https://www.revolvy.com/main/index.php?s=Make+(software)&item_type=topic), [ninja](https://www.revolvy.com/main/index.php?s=Ninja+(build+system)&item_type=topic), Apple's [Xcode](https://www.revolvy.com/main/index.php?s=Xcode&item_type=topic" \o "Xcode" \t "_blank), and [Microsoft Visual Studio](https://www.revolvy.com/main/index.php?s=Microsoft+Visual+Studio&item_type=topic).[2]CMake is also direcly used by some [IDE](https://www.revolvy.com/main/index.php?s=Integrated+development+environment&item_type=topic) as [Qt Creator](https://www.revolvy.com/main/index.php?s=Qt+Creator&item_type=topic)[3]and [GNOME Builder](https://www.revolvy.com/main/index.php?s=GNOME+Builder&item_type=topic)[4].
* [GNU Build System](https://www.revolvy.com/main/index.php?s=GNU+Build+System&item_type=topic) (aka autotools), a collection of tools for portable builds. These in particular include [Autoconf](https://www.revolvy.com/main/index.php?s=Autoconf&item_type=topic" \o "Autoconf" \t "_blank) and [Automake](https://www.revolvy.com/main/index.php?s=Automake&item_type=topic" \o "Automake" \t "_blank), cross-platform tools that together generate appropriate localized makefiles.
* [GYP](https://www.revolvy.com/main/index.php?s=GYP+(software)&item_type=topic) (Generate Your Projects) - Created for [Chromium](https://www.revolvy.com/main/index.php?s=Chromium+(web+browser)&item_type=topic); it is another tool that generates files for the native build environment
* [imake](https://www.revolvy.com/main/index.php?s=Imake&item_type=topic)
* [Meson](https://www.revolvy.com/main/index.php?s=Meson+(software)&item_type=topic), a build system optimized for performance and usability is based on [ninja](https://www.revolvy.com/main/index.php?s=Ninja+(build+system)&item_type=topic) on Linux, [Visual Studio](https://www.revolvy.com/main/index.php?s=Visual+Studio&item_type=topic) on Windows and [Xcode](https://www.revolvy.com/main/index.php?s=Xcode&item_type=topic" \o "Xcode" \t "_blank) on macOS. Meson is also directly used by [GNOME Builder](https://www.revolvy.com/main/index.php?s=GNOME+Builder&item_type=topic).[5]
* [OpenMake Software](https://www.revolvy.com/main/index.php?s=OpenMake+Software&item_type=topic) Meister
* [Premake](https://www.revolvy.com/main/index.php?s=Premake&item_type=topic), a Lua-based tool for making makefiles, Visual Studio files, Xcode projects, and more
* [qmake](https://www.revolvy.com/main/index.php?s=Qmake&item_type=topic)

**Continuous integration -**[AnthillPro](https://www.revolvy.com/main/index.php?s=AnthillPro&item_type=topic" \t "_blank" \o "AnthillPro), build automation with pipeline support for deployment automation and testing. Cross-platform, cross-language

* [Apache Continuum](https://www.revolvy.com/main/index.php?s=Apache+Continuum&item_type=topic) - discontinued
* [Bamboo](https://www.revolvy.com/main/index.php?s=Bamboo+(software)&item_type=topic), continuous integration software
* [Buildbot](https://www.revolvy.com/main/index.php?s=Buildbot&item_type=topic), a Python-based software development continuous integration tool which automates the compile/test cycle
* [BuildIT](https://www.revolvy.com/main/index.php?s=BuildIT&item_type=topic), a free graphical build or task tool for Windows with an emphasis on simplicity and ease of use
* [CABIE](https://www.revolvy.com/main/index.php?s=CABIE&item_type=topic) Continuous Automated Build and Integration Environment, open source, written in Perl
* [CruiseControl](https://www.revolvy.com/main/index.php?s=CruiseControl&item_type=topic), for Java and .NET
* [Go continuous delivery](https://www.revolvy.com/main/index.php?s=Go+continuous+delivery&item_type=topic), open source, cross-platform
* [Hudson](https://www.revolvy.com/main/index.php?s=Hudson+(software)&item_type=topic), an extensible continuous integration engine
* [Jenkins](https://www.revolvy.com/main/index.php?s=Jenkins+(software)&item_type=topic), an extensible continuous integration engine, [forked](https://www.revolvy.com/main/index.php?s=Fork+(software+development)&item_type=topic) from Hudson
* [TeamCity](https://www.revolvy.com/main/index.php?s=TeamCity&item_type=topic)
* [Team Foundation Server](https://www.revolvy.com/main/index.php?s=Team+Foundation+Server&item_type=topic), an extensible continuous integration engine from Microsoft
* [Travis CI](https://www.revolvy.com/main/index.php?s=Travis+CI&item_type=topic), a hosted continuous integration service
* [Visual Studio Team Services](https://www.revolvy.com/main/index.php?s=Microsoft+Visual+Studio&item_type=topic), provides [Azure](https://www.revolvy.com/main/index.php?s=Microsoft+Azure&item_type=topic) hosted build capabilities

**Configuration management**

* [Ansible](https://www.revolvy.com/main/index.php?s=Ansible+(software)&item_type=topic) (Python-based)
* [CFEngine](https://www.revolvy.com/main/index.php?s=CFEngine&item_type=topic)
* [Chef](https://www.revolvy.com/main/index.php?s=Chef+(software)&item_type=topic) (Ruby-based)
* [Distelli](https://www.revolvy.com/main/index.php?s=Distelli&item_type=topic)
* [OpenMake Software](https://www.revolvy.com/main/index.php?s=OpenMake+Software&item_type=topic) Release Engineer
* [Otter](https://www.revolvy.com/main/index.php?s=Otter+(software)&item_type=topic)
* [Puppet](https://www.revolvy.com/main/index.php?s=Puppet+(software)&item_type=topic) (Ruby-based)
* [Salt](https://www.revolvy.com/main/index.php?s=Salt+(software)&item_type=topic) (Python-based)
* [Rex](https://www.revolvy.com/main/index.php?s=Rex+(software)&item_type=topic) (Perl-based)

**Meta-build**

A meta-build tool is capable of building many different projects using a subset of existing buildtools. Since these usually provide a list of packages to build, they are also often called [package managers](https://www.revolvy.com/main/index.php?s=Package+manager&item_type=topic).

* [Pkgsrc](https://www.revolvy.com/main/index.php?s=Pkgsrc&item_type=topic), package manager of NetBSD and other operating systems.
* [Portage](https://www.revolvy.com/main/index.php?s=Portage+(software)&item_type=topic), package manager of the Gentoo Linux distribution
* [Nix](https://www.revolvy.com/main/index.php?s=Nix+package+manager&item_type=topic), functional package manager for the [NixOS](https://www.revolvy.com/main/index.php?s=NixOS&item_type=topic" \o "NixOS" \t "_blank) Linux distribution focusing on [reproducible builds](https://www.revolvy.com/main/index.php?s=Reproducible+build&item_type=topic)
* [Guix](https://www.revolvy.com/main/index.php?s=GNU+Guix&item_type=topic), functional package manager for the GuixSD Linux distribution, based on Nix.
* [Collective Knowledge](https://www.revolvy.com/main/index.php?s=Collective+Knowledge+(software)&item_type=topic), cross-platform package manager to rebuild software environment for research workflows

**Others**

* [checkinstall](https://www.revolvy.com/main/index.php?s=Checkinstall&item_type=topic), checkinstall is a program that monitors an installation procedure, and creates a standard package for your distribution.
* [Open Build Service](https://www.revolvy.com/main/index.php?s=Open+Build+Service&item_type=topic), a hosted service to help build packages for various Linux distributions

**Dependency Network -***A list of tasks linked by dependency relationships. To run a task, you invoke its dependencies, running those tasks as prerequisites.* Building a software system is a common predicament for software developers. At various points, there are various things that you may want to do: just compile the program, or run tests. If you want to run tests, you need to make sure your compilation is up-to-date first. In order to compile, you need to ensure you've carried out some code generation.A Dependency Network organizes functionality into a directed acyclic graph (DAG) of tasks and their dependencies on other tasks. In the case above, we would say that the test task is dependent upon the compilation task, and the compilation task is dependent upon the code generation task. When you request a task, we first find any tasks it depends on and ensure they are executed first, if needed. We can navigate through a dependency network to ensure that all the prerequisite tasks necessary for the requested task are executed. We can also make sure that even if a task crops up more than once through different dependency paths, it's still executed only once.

<http://scmquest.com/software-build-best-practices-scm/>