Introduction to CI

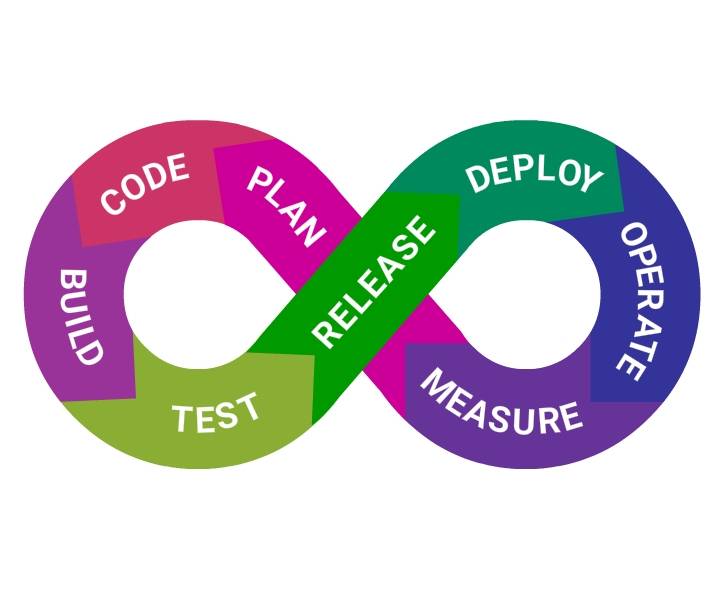
Will you be pleased to discover an error, a show stopper, at a crucial stage in your software?

Obviously, the answer is ***No***!

What steps can be taken to detect and fix coding issues at an early stage of the development cycle?

**Continous Integration** (CI) can help in providing a solution.

What is CI?



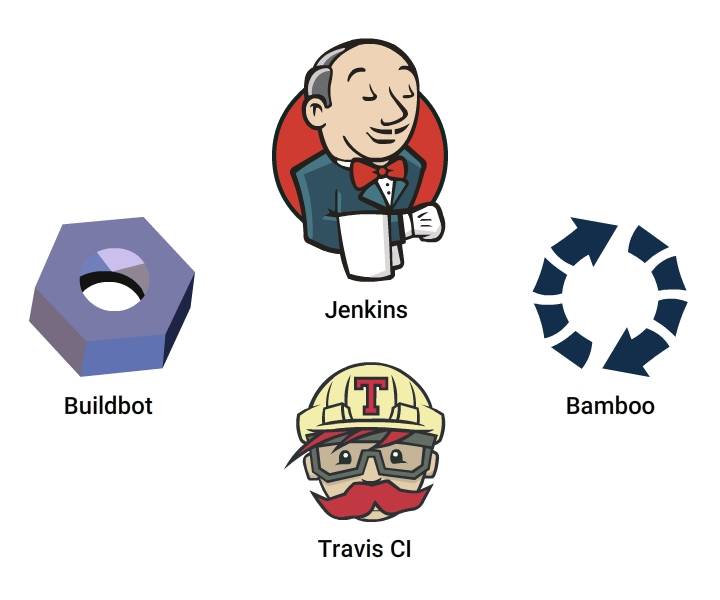
According to Martin Fowler,

**Continuous Integration** is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible.

##### Quick Fact

The term **Continuous Integration** was first coined by Grady Booch.

How to Enable CI?



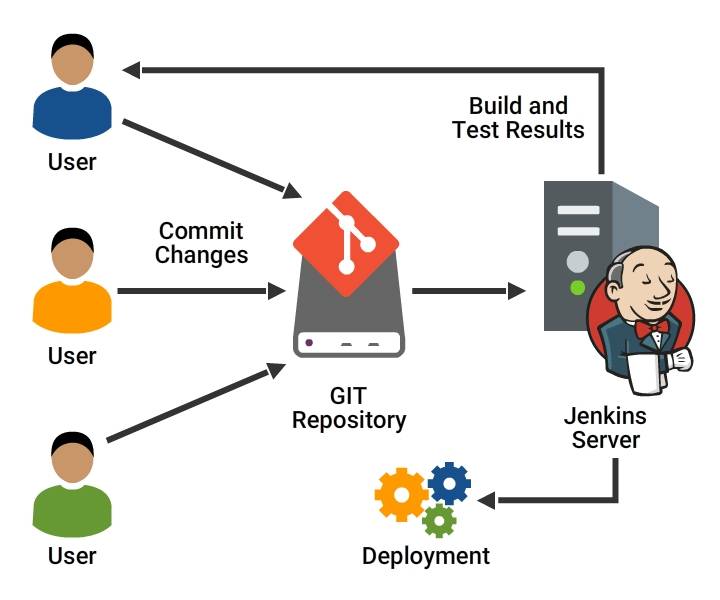
You need a tool to create a CI enabled environment. Jenkins, Travis CI, Bamboo,Buildbot are different tools available to enable CI.

In this course, you will be learning about **Jenkins**, a widely used open source CI tool, written in Java.

##### Quick Fact

Jenkins was earlier referred to as **Hudson**

##### What does Jenkins do?



* **Compiles** and **builds** the code
* **Runs** an internal shell or command line script
* **Starts execution** of the integration tests
* **Monitor** execution of tasks
* **Stops build** in case of failure
* **Notify user** on the build status
* **Deploy** in test or production environments

##### Features of Jenkins

* **Easy to install**
* **Easy to configure** various tasks
* **Rich plugin ecosystem** - Integrates with a variety of build, test,deploy,reportingtools
* **Permanent links** - Jenkins provides direct links to the latest or failed build, which can be used for easy communication

##### Features of Jenkins Contd.

* **Extensibility** - Customize Jenkins to suit your needs
* **Distributed builds** - Jenkins can distribute build, test jobs to multiple computers with different operating systems
* **File fingerprinting** - Manages dependencies
* **Email integration**- Emails the build status

##### Difference made by Jenkins

***Pre Jenkins:***

* Source code was completely built and then tested
* Bugs identified during testing in the source code, should be fixed and then re-tested
* Slows the software delivery, as the entire process is manual

***Post Jenkins :***

* Once code change is committed,Jenkins automatically takes care of the build , test and reporting of results.

##### Quick Fact

Jenkins was primarily developed by Kuhsoke Kawaguchi.

##### Pre-Requisites to Install Jenkins



Recommended minimum configuration for installing Jenkins:

* **locally** - Java 8, 256MB RAM and > 1 GB free disk space
* **small team** - Java 8, >1 GB RAM and > 50 GB free disk space

Since all the builds take place on the Jenkins machine , system should have enough disk space for build storage. Jenkins can be installed on **Windows**, **Ubuntu/Debian**, **Red Hat**, **Fedora/CentOS**, **Mac OS**, **X openSUSE**

##### Installing Jenkins

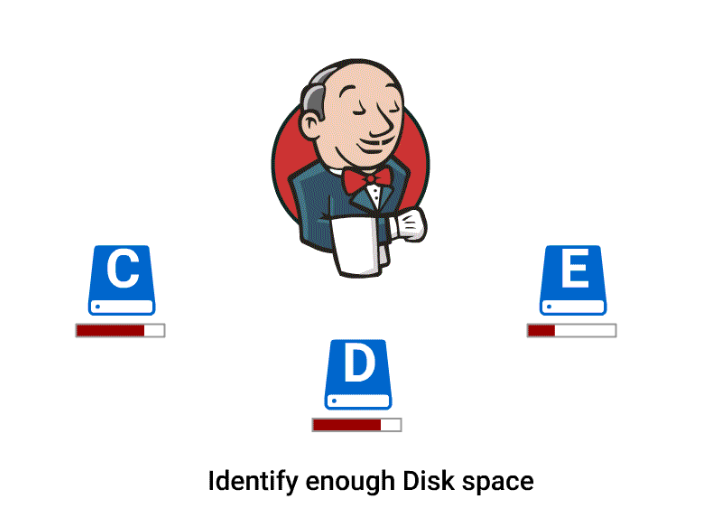
Jenkins can be started from command line or can run on a web application server.

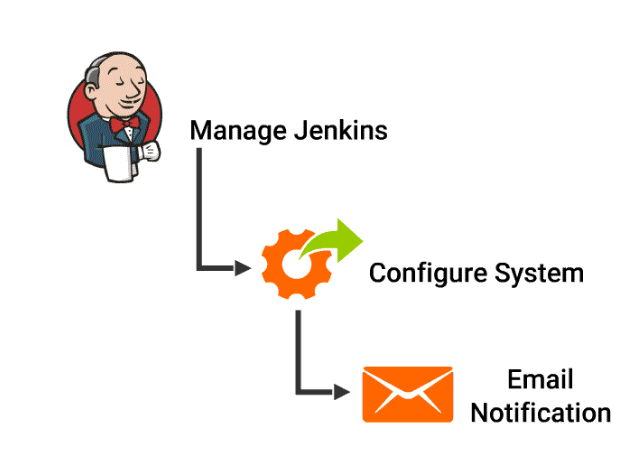
* Download the jenkins.war file from [Jenkins](https://jenkins.io/)
* Start Jenkins directly from Command line with java-jar jenkins.war
* On Successful completion ,Jenkins can be accessed locally from http://localhost:8080/

**To run it from *Tomcat server***

* **Put** the .war file into the webapps directory
* **Start** Tomcat, Jenkins installation will be available on (<http://localhost:8080/jenkins>)

##### Configuring Jenkins





Jenkins installation is now complete. What are the next steps?

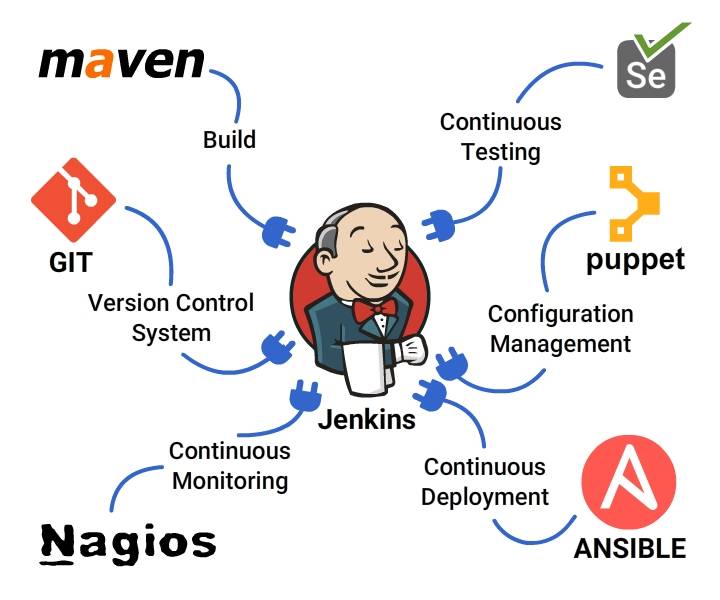
* Select the Jenkins home directory (prefer location with enough free space)
* Decide number of concurrent job executions to be allowed on ***Jenkins*** machine

Configuring Jenkins Contd.

* Add custom environment variables
* Mention SMTP server,user email suffix in the email notification section
* Configure the location of JDK installation
* To build Maven applications configure the location of Maven Home

You can perform these tasks by selecting ***Configure System*** under ***Manage Jenkins***.

Extending Jenkins Functionality



* Once Jenkins is installed, it is time to configure it, to fit your needs
* Jenkins has relatively few abilities, but it aids the s/w developers by providing a variety of plugins
* Plugins are add-ons that allow Jenkins to interact with many other softwares
* The exact plugins you install depends on the nature of your project

##### Plugins in Jenkins

[Jenkins Plugins Index](https://plugins.jenkins.io/) provides you the various plugin options such as

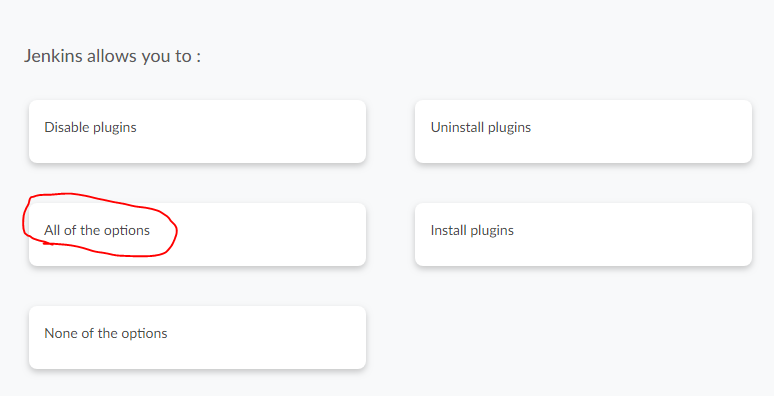
* **Source Control**: Git, SVN, Mercurial
* **Testing** : Selenium, Windmill
* **Triggers**: Jabber, Directory watchers
* **Artifact**: To copy components between projects like Amazon S3, SCP

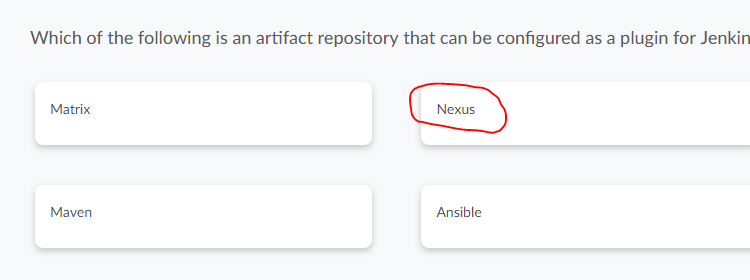
##### Plugins in Jenkins Contd.

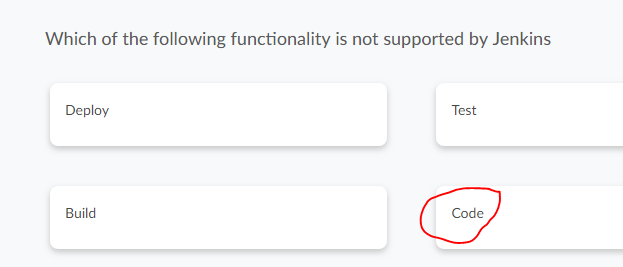
* **Code Analysis**: To parse the code with tools like CheckStyle,Findbugs,PMD
* **Build Tools**: In large projects use a build manager such as **Maven** or **Ant**.
* **Reporting**: Jenkins provides its own reports. It can be extended using tools like Static Analysis Collector that collects the different analysis results and shows it in a combined trend graph.

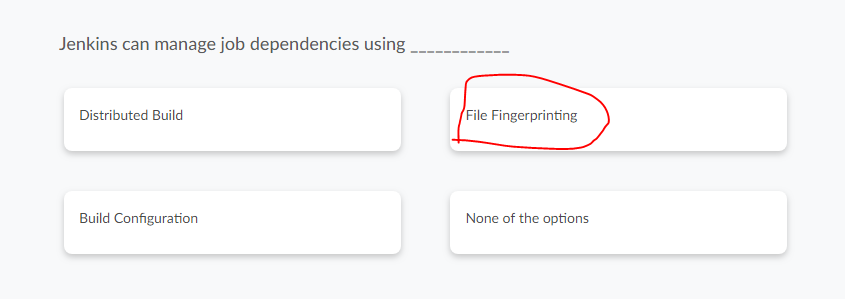
Plugins can be configured via the **Manage Plugins** under **Manage Jenkins**.

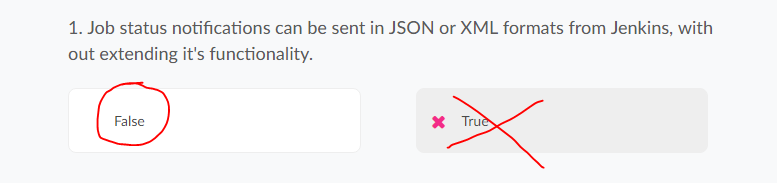
To learn how to add a plugin, try out steps 1,2 and 3 in [katacoda playground](https://katacoda.com/courses/jenkins/build-docker-images) .

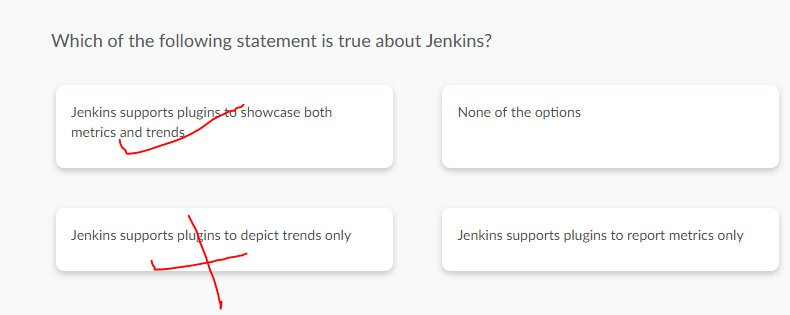




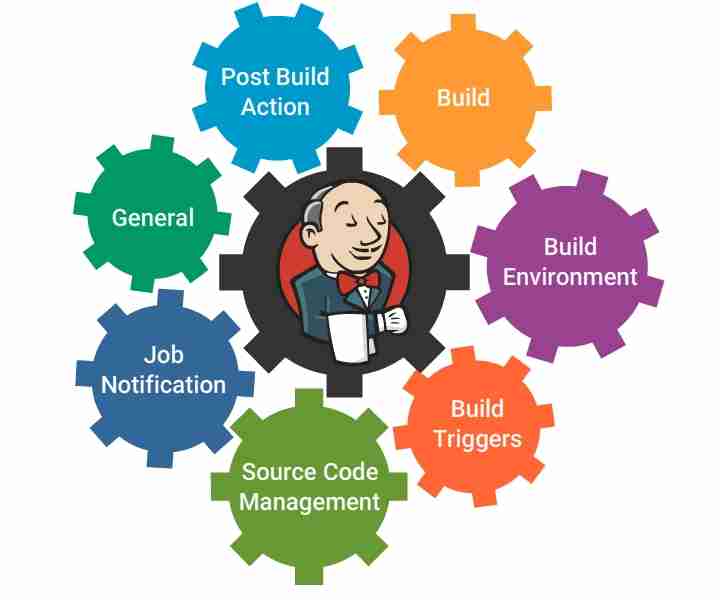








Set up a simple project



For setting up a new project in ***Jenkins*** , following sections are to be planned and configured as required:

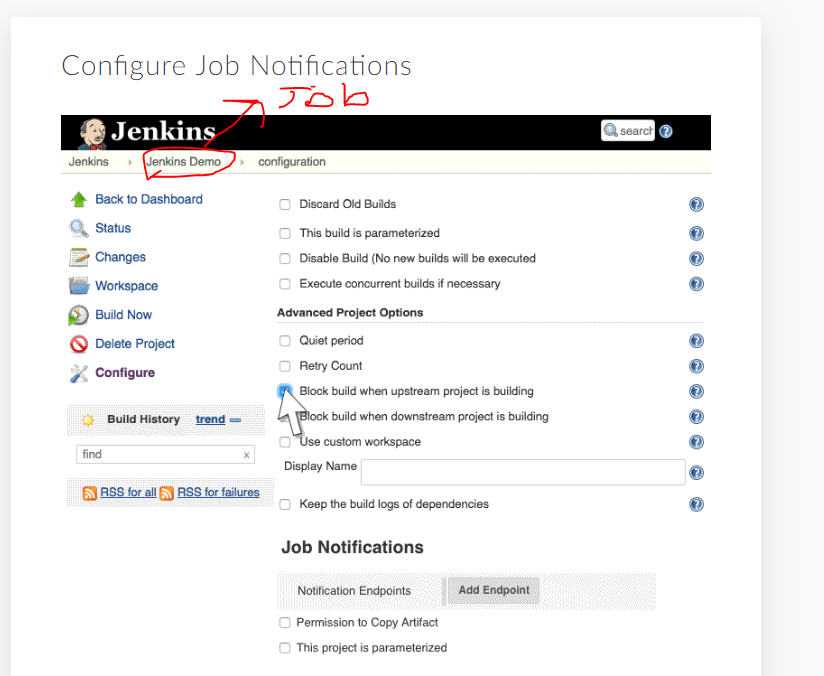
* ***SCM*** - Associate with a version control server
* ***Triggering Build*** - Control when Jenkins will perform builds by Polling, Periodic or Build based on other projects
* ***Execution of scripts, Ant and Maven targets***
* ***Archiving the artifacts***
* ***Recording and publishing*** build and test results
* ***Email notifications***

Next few cards, will help you in defining these sections.

Create a new project

Now you will learn, how to create a simple ***build*** project.

* Select New Item from Jenkins dashboard
* Type the project name
* Select Freestyle project (freestyle is the most configurable and flexible option, easy to setup!)
* Click ok





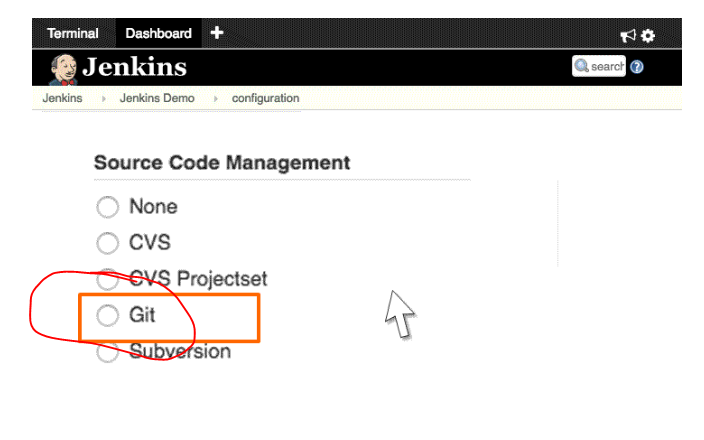
In the job notifications section, you can choose to -

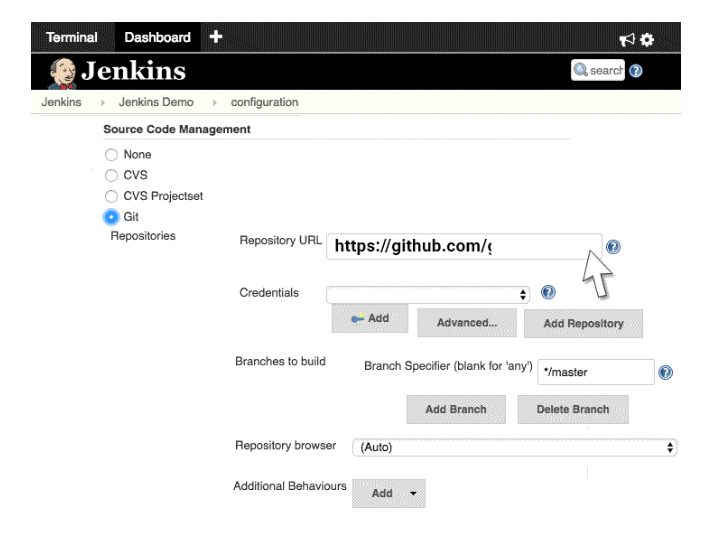
* Prevent build of your project, when the dependent upstream or downstream job is in the queue or is building

Notifications plugin like Tikal will expand the available notification options.

* On installing Tikal , you can send job status in JSON and XML formats
* Set up the notification endpoint section with the required details

Configure SCM



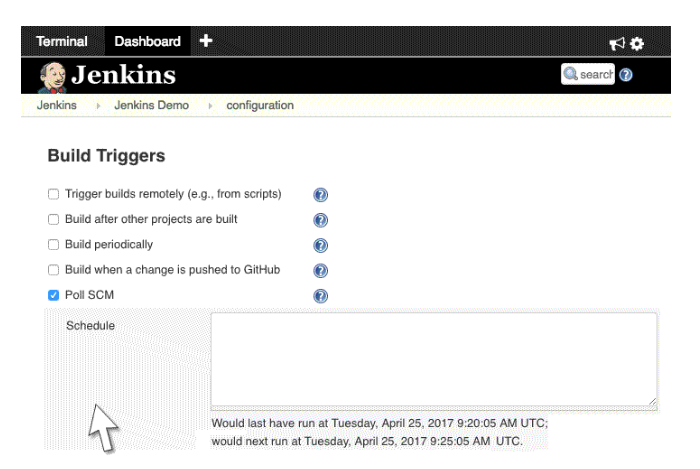


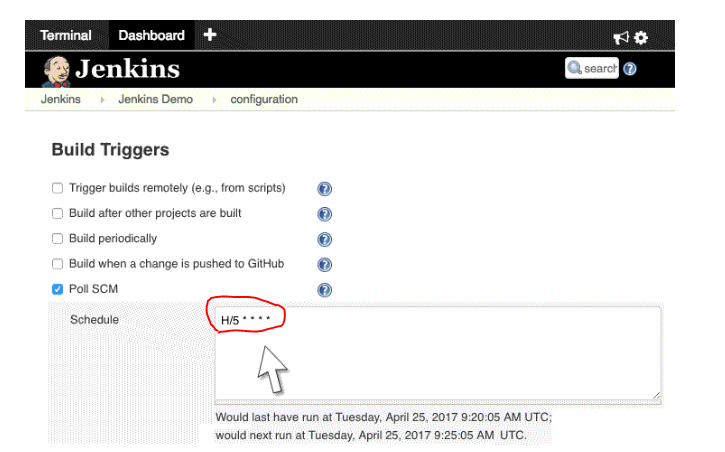
Jenkins works by **checking** the changes to the source code of your project, **building** it in its own workspace.

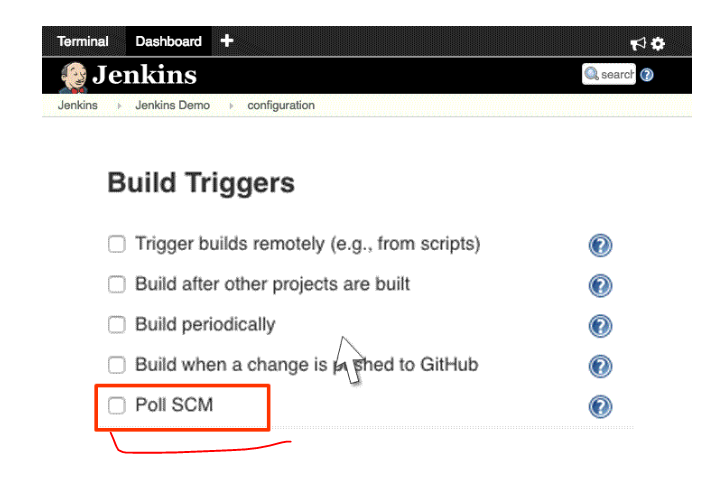
In the ***source code management section*** -

* Select location of files to be built
* If you select git, then under repository URL mention the git location
* If location is github, then under ADD button, mention the user id and ***password***for the github repository

Configure Build Triggers







After source code location is defined , you need to ***configure*** Jenkins to check for code changes, so that, build is triggered automatically.

Various options to trigger the builds are:

* Build whenever a SNAPSHOT dependency is built
* Trigger builds remotely (e.g., from scripts)
* Build after other projects are built
* Build periodically (Runs on CRON job)
* Poll SCM (Runs on CRON job)

How to define CRON expression?

Jenkins schedules are configured using the CRON syntax.

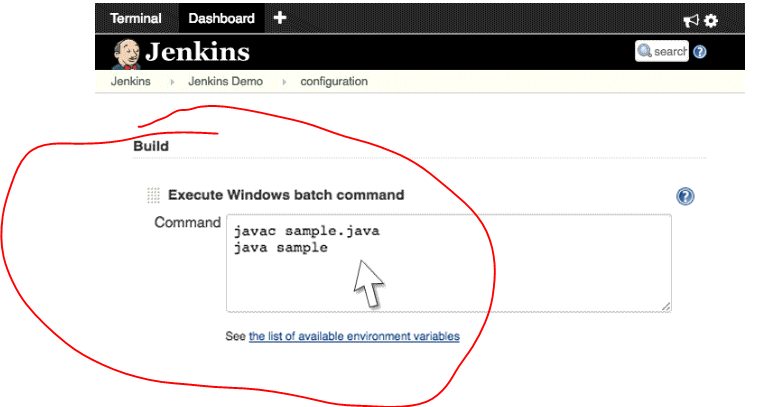
It consists of ***five fields*** separated by white space, indicating respectively the minute (0–59), hour (0–23), day of the month (1–31), month (1–12) and the day of the week (0–7, with 0 and 7 being Sunday). star is a wildcard character which accepts any valid value for that field.

* “\* \* \* \* \*” means every minute of every hour of every day.
* “\* 9-17 \* \* \*” means “every minute of every day, between 9am and 5pm.”

There are other convenient short-hands, such as “@daily” and “@hourly”.

Adding a Build Step





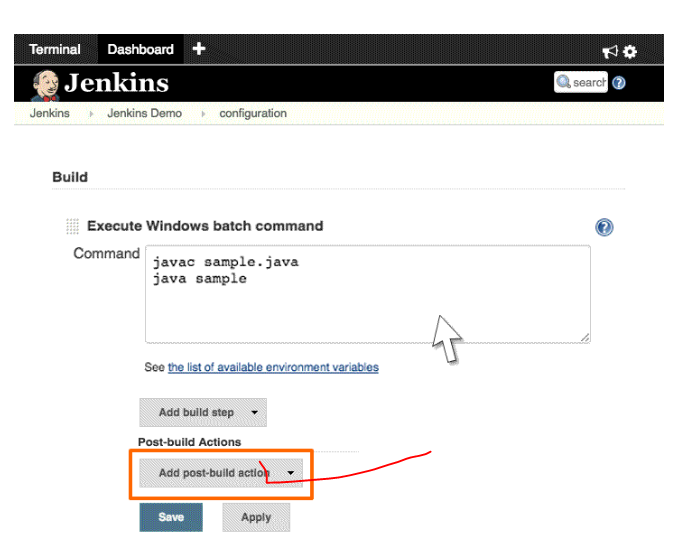
Now you have learnt how to setup the code location and build frequency. Next step is to configure the build action.

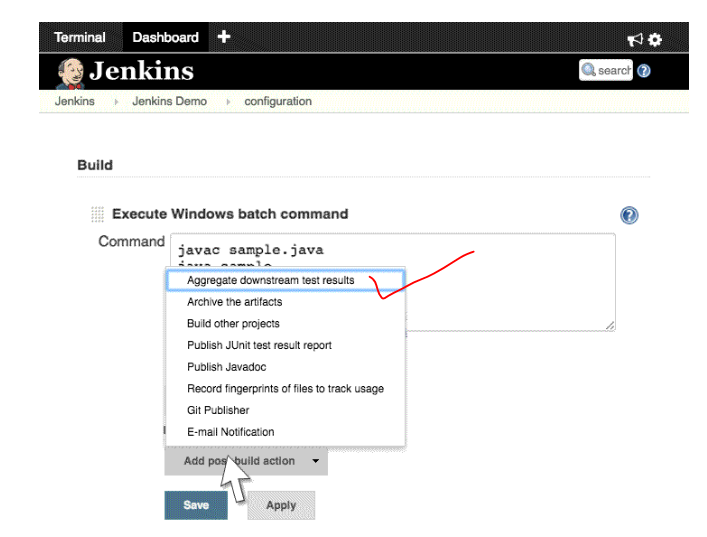
In the build section, select ***Add Build step*** and opt for the required build option.

* ***For simple java build*** - Select Execute Windows Batch command and enter the script in the command window
* ***For Maven build*** - Select Invoke top-level Maven targets. Enter clean package,clean install or clean test as appropriate in the Goals field .

Hope you know - clean package will delete any previous build artifacts, compile code, run unit tests and generates a JAR file.

Configure Post Build Activities



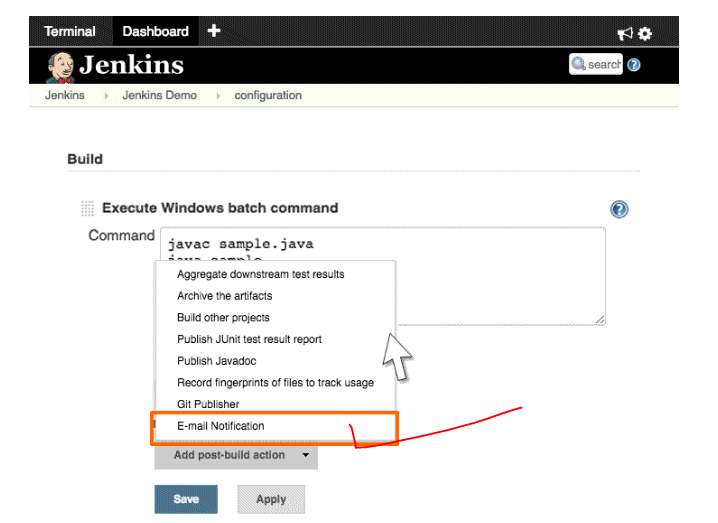


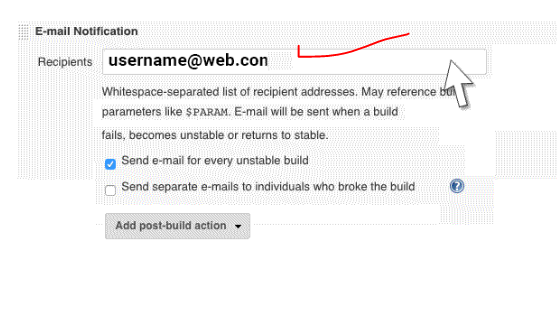
Once your code is built, the results should be displayed for you to check and act. Jenkinsdoes a great job of displaying test results and trends. Some of those are :

* Aggregate downstream test results
* Archive the artifacts
* Build other projects
* Deploy artifacts to maven repository
* Record fingerprints of files to track usage
* Email Notification

Install post-build script plugin to help you execute scripts after build completion.

##### Notifications





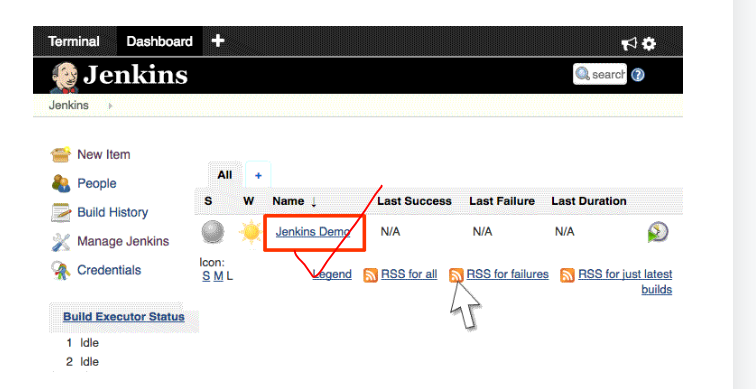
Now you have setup everything that is required for a build job : Detect code change, Trigger build , Record results.

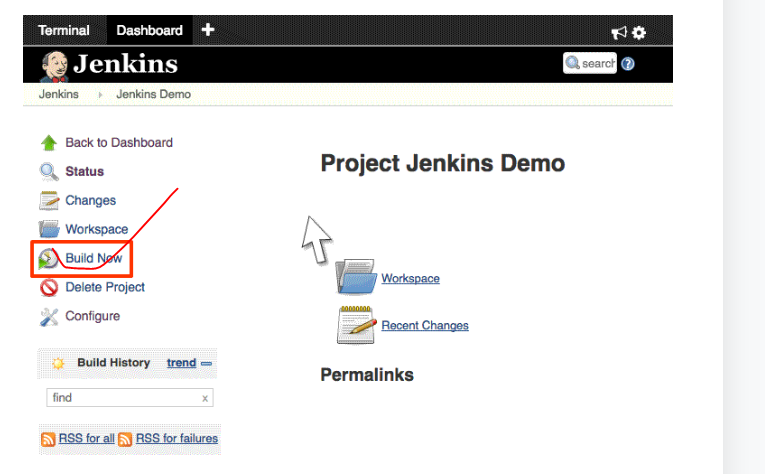
But, will you not require an automatic notification of the build status?

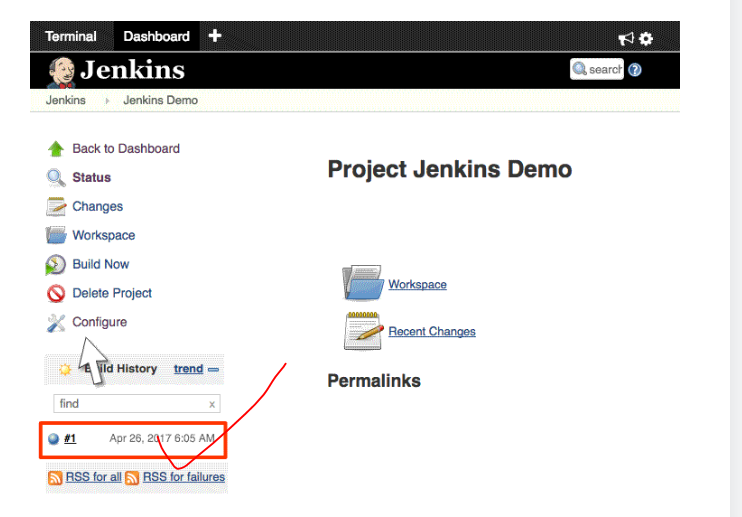
How about an email notification?

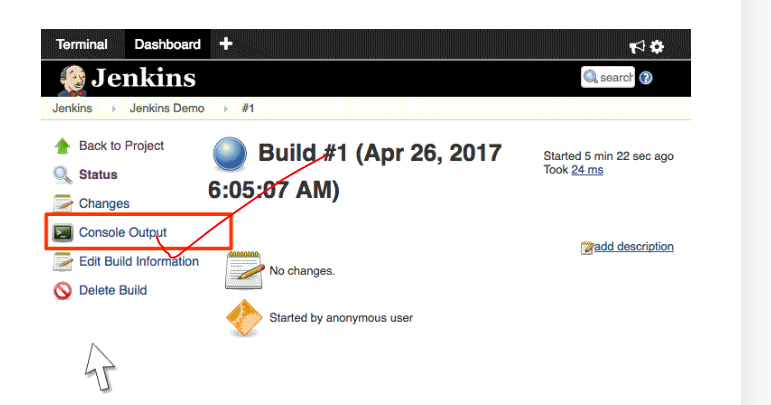
***Jenkins* provides an *Email Notification section***. Mention the recipients under your project configuration.

Executing the build job









Once a project is created successfully in Jenkins:

* Build job is displayed on the Jenkins dashboard
* Build starts automatically based on the build triggersettings
* To run the job manually select Build Now
* Progress of the build is displayed in the Build History section
* Once build completes,click on the build number
* Select the console output to see the details of the build