Jenkins

**Continuous Integration** is a software development practice where members of a team integrate their work frequently.

The term Continuous Integration was first coined by Grady Booch.

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

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, reporting tools
* Permanent links - Jenkins provides direct links to the latest or failed build, which can be used for easy communication

**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

**Configuring Jenkins**

* 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.

**Jenkins Plugins**

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

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

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!)

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.

**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”.

**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.

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

**Post-build**

* 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.

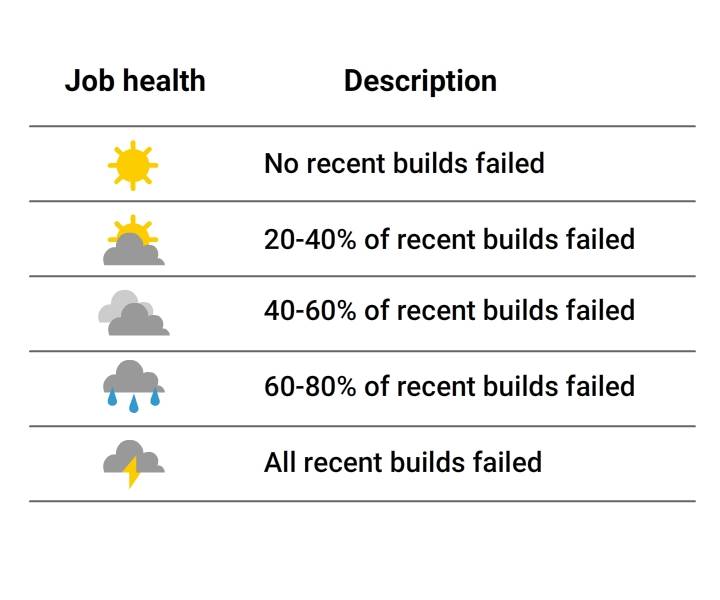
**Executing the build job**

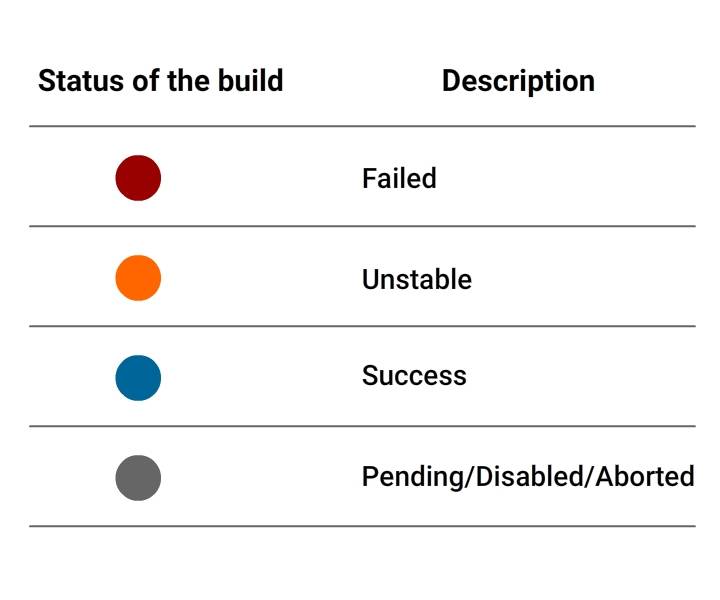
Once a project is created successfully in Jenkins:

* Build job is displayed on the Jenkins dashboard
* Build starts automatically based on the build trigger settings
* 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.

Build Status is indicated in two ways :

* A weather icon (on the home page dashboard) - shows you a record of multiple builds
* A colored ball (on the individual project page) - shows you status of a single build
* Status has corresponding tooltips with explanations , when you hover over it.





##### Setting up Jenkins CI Pipeline

* In the Katacoda exercise, you would have learnt how to use Jenkins and Docker in the CI pipeline.

Below are steps to setup Jenkins CI pipeline from Git to Tomcat.

**Pre-Requisites**

* Jenkins, Maven, Tomcat should be up and running in your machine
* Souce code should be made available in Git

**Plugins to install**

* Deploy to Container , Git

**Jenkins Jobs to be created-**

* Create a Pull job that detects code changes in Git
* Create a Build job that is triggered after the Pull job
* Create a Deploy job to push results to tomcat.

##### Setting up Deploy Job

You should be able to create the first two jobs easily. Below steps will help you to create the deploy job.

**In the Post-build Actions section:**

* Select deploy war/ear to a container
* Mention the \*.war file
* Mention Context path- where your application will be published in Tomcat
* Select Tomcat version in the Container dropdown
* Enter credentials of user who can access Tomcat
* Mention Tomcat URL to reach your tomcat instance

On changing your code in Git, you will see these jobs will trigger automatically on Jenkins.

Jenkins provides a host of plugins for unit testing such as Junit and Mstest for .Net unit tests.

**If you choose to use the Junit test**

* Select project and choose the configure option
* In Add Build step, choose action to invoke Ant.
* Select Advanced and enter the location of build.xml file
* In Add post-build, select Publish Junit test results.
* In Test Report XMLs field, enter location of result xml files produced by executing Junit test cases
* Save and then build.

**Use Automated tests**

* For this, install Hudson Selenium Plugin from ***Manage Plugins*** section.
* Once plugin is installed successfully, select your project, click on the configure option.
* In the Add Build Step section, select the Selenium SuiteRun option.
* Add the necessary details for tests like browser, suitefile, resultfile and Save.

***Now you are ready to build and get it going...***

##### How to distribute work load?

* What if, as part of a large project, you need to build codes and report results regularly?
* Running all the builds on a centralized machine would not be the best option.
* What if, you need to automatically test the code in various environments?
* Different environments are to be set up in separate machines.

**Jenkins uses Master** - Slave architecture for managing distributed builds. In master node, you need to install Jenkins. While it can also execute build jobs, it handles all related tasks for build system -

* Schedule build job
* Dispatch build job to slaves
* Monitor the slave
* Report the build results

**Master-Slave Communication**

Slave node offloads work from the master.

* It can run on various operating systems.
* Executes build jobs dispatched by master.
* Jenkins need not be fully installed on slave.
* To operate, master and slave will establish bi-directional communication link like TCP/IP

##### How to Setup Slave node ?

To setup slave node:

* Select Manage Node under Mange Jenkinsand Click New Node
* Select the dumb slave or permanent agent option and Click Ok
* In the next window, enter details like the IP address and the user credentials
* Select the launch method and then Save.

On completion of the steps mentioned in the previous card, the new node machine will be online and ready .

* To execute the project in the slave node:
* Select 'Restrict where this job can be run' option under your 'project configuration`
* Mention the slave name in Label expression field.

##### How to Backup Jenkins ?

Taking project backups are crucial in software delivery process.

In Jenkins, all build logs, archives, configuration settings are stored under the Jenkins home directory.

To backup the home directory :

* You can manually copy it to another location OR Install backup plugin
* If you choose to use the backup plugin, you need to trigger backup manually
* If full backup of home directory is not needed, use ThinBackup plugin to backup specific details, like jobs, configuration, build history.