

O Jenkins είναι ένα -ανοιχτού κώδικα- εργαλείο αυτοματισμού, γραμμένο σε Java, ο οποίος χρησιμοποιείται για την αυτοματοποίηση όλων των ειδών εργασιών που σχετίζονται με την κατασκευή (build), τον έλεγχο (testing), την παράδοση ή την ανάπτυξη λογισμικού.

Ο Jenkins δημιουργήθηκε το 2011, αλλά υπήρχε υπό την επωνυμία Hudson του Kohsuke Kawaguchi από το 2004. Σήμερα ο Jenkins είναι ο κορυφαίος διακομιστής αυτοματισμού ανοιχτού κώδικα με περίπου 1.400 plugins που υποστηρίζουν την αυτοματοποίηση όλων των ειδών εργασιών. Αυτά τα 1.400 plugins καλύπτουν 5 περιοχές: Τις πλατφόρμες, το περιβάλλον χρήστη, το administration, την διαχείρηση πηγαίου κώδικα και συχνότερα, την διαχείριση builds. Κάποια από τα πιο διαδομένα plug-ins είναι τα: Git, Amazon EC2, Maven 2 project, HTLM publisher κ.α.

Η ανάγκη για την δημιουργία ενός τέτοιου server ξεκίνησε επείδη παλαιότερα οι developers δεν είχαν τρόπο να ελέγχουν τον κώδικα αν τρέχει στο commit με αποτέλεσμα στα περισσότερα commits να καταστρέφεται ο κώδικας και να χάνεται πολύτιμος χρόνος περιμένωντας να γίνει το testing και να εντοπιστεί το λάθος. Με το Jenkins αυτό το πρόβλημα λύθηκε επειδή επιτρέπει στους developers να γίνει έκδοση και έλεγχος με κάθε commit κατευθείαν, οπότε σε περίπτωση λάθους θα πρέπει να ασχοληθούν μόνο με το τελευταίο commit. Επίσης αρκούσε μονο το commit στον πηγαίο κώδικα, καθώς όλα τα υπόλοιπα γινόντουσαν αυτόματα απο το Jenkins. Αυτή η τεράστια αλλαγή οδήγησε σε πολύ πιο συχνές κυκλοφορίες λογισμικών κ.α.

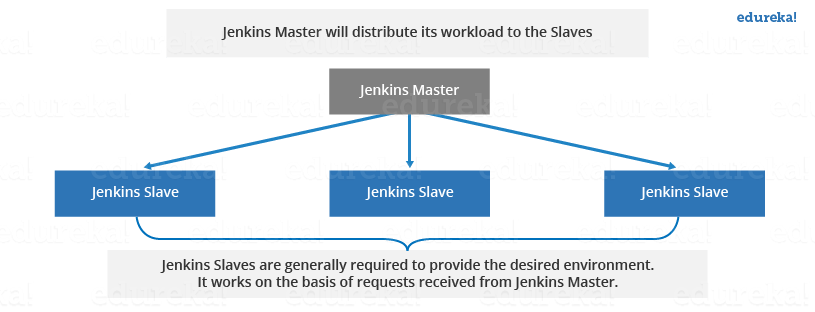
Στην αρχή, η αρχιτεκτονική του Jenkins ήταν η εξής:

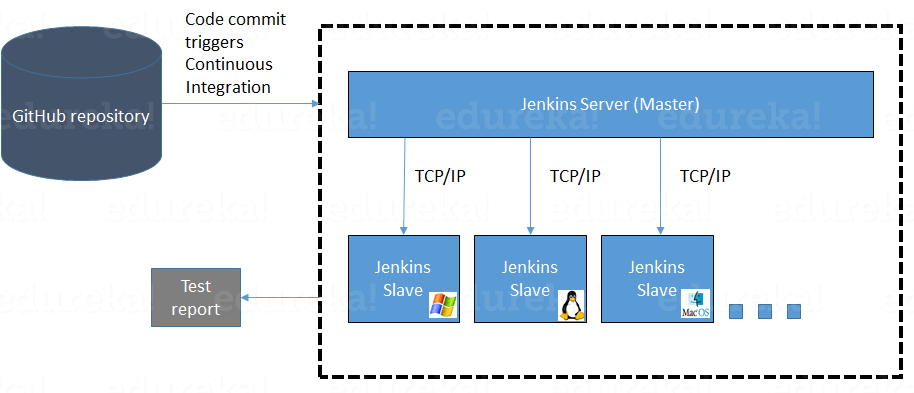
Υπήρχε ένας **μονός Jenkins Server** ο οποίος σε κάθε αλλαγή έπερνε από το Git Repository τα commits των developers, τα έκανε build, τα έστελνε σε έναν test server για έλεγχο και αφού έβγαζε ένα τελικό αποτέλεσμα, το μετέδιδε πίσω στους προγραμματιστές.

Στην συνέχεια όμως, επειδή το μέγεθος των project μεγάλωνε, αλλά και επειδή οι προγραμματιστές μπορεί να θέλαν να δοκιμάσουν τον κώδικα σε διάφορα περιβάλλοντα, ένας μονός server Jenkins δεν αρκούσε. Oπότε άλλαξε την αρχιτεκτονική του σε **“Master/Slave” architecture**.

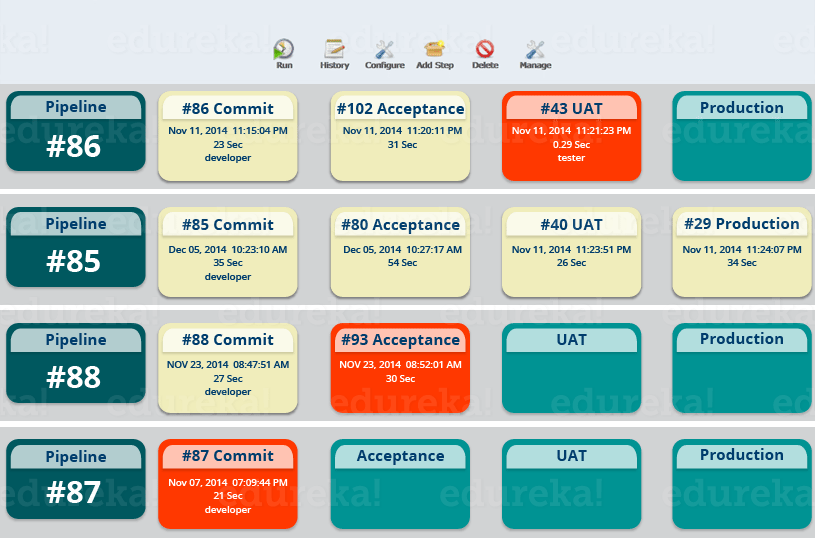
Αυτό σημαίνει ότι υπήρχε:

* Ένας κύριος server, ο Master, ο οποίος η δουλειά του είναι να προγραμματίζει τα builds jobs, να τα στέλνει στα Slaves (βλ.παρακάτω) για να κάνουν την υλοποίηση, να παρακολουθεί τα Slaves, να καταγράφει και να παρουσιάζει τα αποτελέσματα και φυσικά να πραγματοποιεί build jobs από μόνο του.
* Διάφοροι δευτερεύοντες servers, οι Slaves, οι οποίοι «ακούν» εντολές από τον Master και μπορούν να τρέξουν σε διάφορα περιβάλλοντα τα build jobs που έστειλε ο Master.





Επίσης, πολλές φόρες μπορεί πολλοί προγραμματιστές να κάνουν αλλαγές την ίδια στιγμή, με αποτέλεσμα να υπάρχει η ανάγκη για την παρακολούθηση των ενεργειών του Jenkins, δηλαδή ποιό build δέχεται έλεγχο, ποιό build είναι στην ουρά, ποιό build είναι λάθος κ.τ.λ. Για αυτό τον λόγο κατασκεβάστηκε το **Jenkins Pipeline**, το οποίο παρέχει μια γενική εικόνα των εργασιών πού κάνει το Jenkins.



**Σάγκλας Μηνάς, 4218**

**Βιβλιογραφία:**

* <https://jenkins.io/doc/>
* <https://www.infoworld.com/article/3239666/what-is-jenkins-the-ci-server-explained.html>
* <https://www.edureka.co/blog/what-is-jenkins/>
* <https://www.edureka.co/blog/jenkins-tutorial/>
* [https://en.wikipedia.org/wiki/Jenkins\_(software)](https://en.wikipedia.org/wiki/Jenkins_(software)#Security)