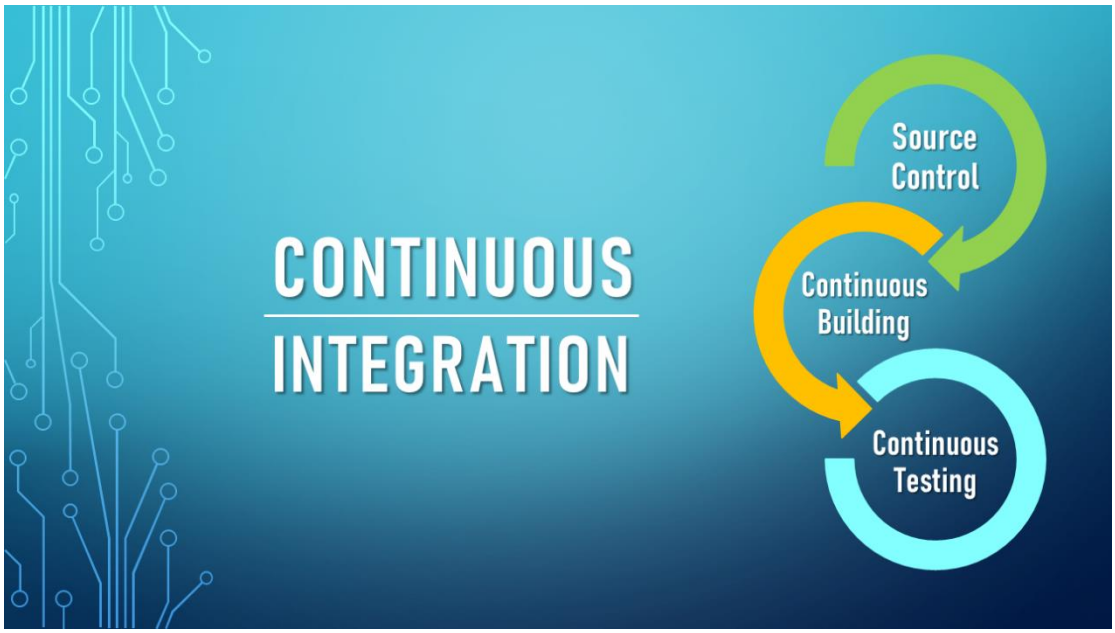
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ & ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

Διπλωματική Εργασία

Ανάπτυξη διαδικτυακής εφαρμογής με αυτόματη προσαρμογή πολυμεσικού περιεχομένου, σε περιβάλλον συνεχούς ενσωμάτωσης.



Φοιτητής: Μωραΐτης Αριστομένης

ΑΜ: 47172

Επιβλέπων Καθηγητής: Δρ. Μιχαήλ Φειδάκης

Συνεπιβλέπων Καθηγητής: Δρ. Χαράλαμπος Πατρικάκης

ΑΘΗΝΑ-ΑΙΓΑΛΕΩ, ΣΕΠΤΕΜΒΡΙΟΣ 2022

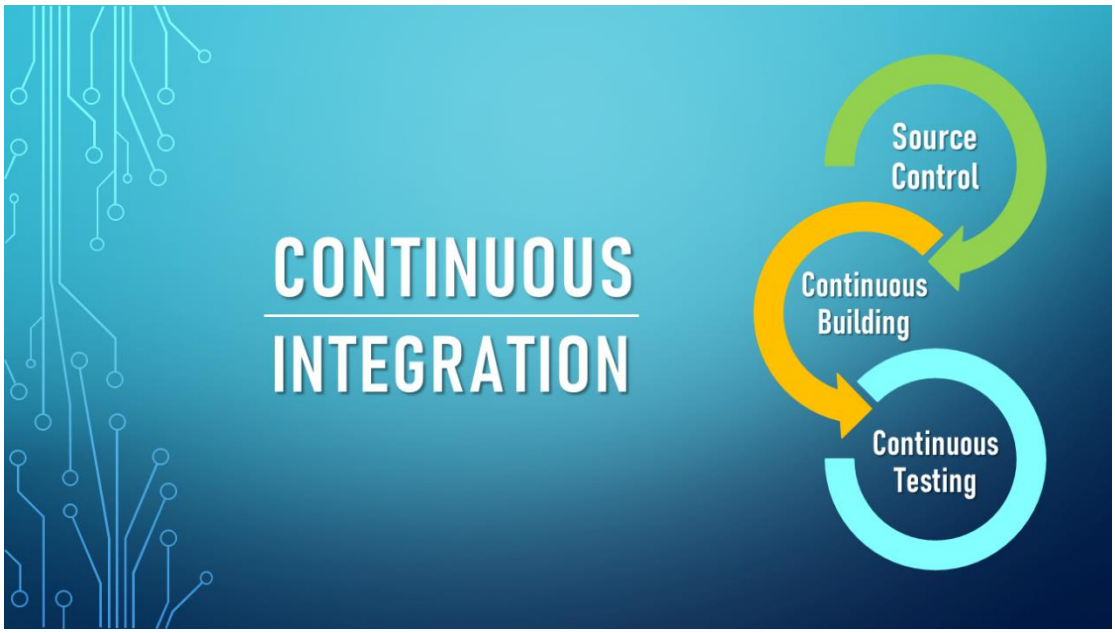
UNIVERSITY OF WEST ATTICA

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

Diploma Thesis

Development of a web application for automated multimedia adaptation, deploying continuous integration.



Student: Moraitis Aristomenis

Registration Number: 47172

Supervisors

Dr. Michail Feidakis

Prof. Charalampos Patrikakis

ATHENS-EGALEO, SEPTEMBER 2022

Η Διπλωματική Εργασία έγινε αποδεκτή και βαθμολογήθηκε από την εξής τριμελή επιτροπή:

(Ονοματεπώνυμο), (βαθμίδα)	(Ονοματεπώνυμο), (βαθμίδα)	(Ονοματεπώνυμο), (βαθμίδα)
Michail Feidakis 2022.10.24 13:51:54 +03'00'	Digitally signed by Charalampos Patrikakis Date: 2022.10.24 14:27:26 +03'00'	Panagiotis Tsiakas Digitally signed by Panagiotis Tsiakas Date: 2022.10.30 20:26:37 +02'00'
(Υπογραφή)	(Υπογραφή)	(Υπογραφή)

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος **Μωραΐτης Αριστομένης** του **Κωνσταντίνου**, με αριθμό μητρώου **47172** φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής ΜΗΧΑΝΙΚΩΝ του Τμήματος ΗΛΕΚΤΡΟΛΟΓΩΝ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ,

δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του διπλώματός μου.

Επιθυμώ την απαγόρευση πρόσβασης στο πλήρες κείμενο της εργασίας μου μέχρι **14/10/2022** και έπειτα από αίτησή μου στη Βιβλιοθήκη και έγκριση του επιβλέποντος/ουσας καθηγητή/ήτριας.»

Ο Δηλών

Μωραΐτης Αριστομένης

Ευχαριστίες

Με την παρούσα διπλωματική εργασία ολοκληρώνεται ο κύκλος των προπτυχιακών μου σπουδών στο Τμήμα Ηλεκτρολόγων και Ηλεκτρονικών Μηχανικών του Πανεπιστήμιου Δυτικής Αττικής.

Αρχικά, θα ήθελα να ευχαριστήσω την οικογένεια μου, που στάθηκαν δίπλα μου όλα και τα χρόνια που διένυσα ως προπτυχιακός φοιτητής και καθ'όλη την διάρκεια εκπόνησης της διπλωματικής μου εργασίας, προσφέροντας την απαραίτητη υποστήριξη που χρειαζόμουν ώστε να ολοκληρώσω τις σπουδές μου με επιτυχία.

Το μεγαλύτερο ευχαριστώ, το οφείλω στον επιβλέπων καθηγητή Μιχαήλ Φειδάκη και στον συνεπιβλέπων καθηγητή Χαράλαμπο Πατρικάκη για την καθοδήγησή τους και όλη την υπομονή που έδειξαν όλο αυτό διάστημα της έρευνας, μελέτης και υλοποίησης της διπλωματικής εργασίας, παρέχοντάς μου πολύτιμες συμβουλές και γνώσεις που βοήθησαν στην βελτιστοποίηση της παρούσας εργασίας.

Περίληψη

Σε μία εποχή όπου οι άνθρωποι έχουν συνηθίσει να έχουν άμεση πρόσβαση σε οποιαδήποτε πληροφορία και οι καταναλωτικές τους επιλογές καθορίζονται πλέον σε μεγάλο βαθμό από τον παράγοντα αυτό, οι εταιρείες αναζητούν τρόπους και λύσεις που θα τους βοηθήσουν να αυξήσουν τον ρυθμό ανάπτυξης και παράδοσης των λογισμικών τους. Για να το πετύχουν όμως αυτό, απαιτούνται εργαλεία τα οποία θα έχουν τη δυνατότητα να εντοπίζουν άμεσα προβλήματα και σφάλματα πριν δημοσιευθούν στους πελάτες, θα διευκολύνουν την εργασία των υπαλλήλων και θα ενισχύουν την παραγωγικότητα και την ομαδικότητά τους, ενώ θα μπορούν να εκτελούν παράλληλα αρκετά από τα στάδια και τις ενέργειες που χρειάζονται για την ταχύτερη ανάπτυξη και παράδοση του λογισμικού. Όλες αυτές οι απαιτήσεις πληρούνται σήμερα από την πρακτική της Συνεχούς Ενσωμάτωσης (Continuous Integration – CI) και τα εργαλεία που την εφαρμόζουν, έχουν καταφέρει πλέον να εισχωρήσουν και να εδραιωθούν, ακόμη και στις εφαρμογές των μεγαλύτερων και πιο δημοφιλών εταιρειών στο χώρο της τεχνολογίας και του διαδικτύου.

Λέξεις – κλειδιά

συνεχής ενσωμάτωση, συνεχής παράδοση, δοκιμές, κατασκευή, συνεχής ανάπτυξη, αποθετήριο, αγωγός ci, εργαλεία ci, λογισμικό, εφαρμογή.

Abstract

At a time when people are used to having instant access to any information and their consumer choices are now largely determined by this factor, companies are looking for ways and solutions that will help them increase the pace of development and delivery of their software. In order to achieve this, they need tools that have the ability to identify problems and bugs before they are going to be published to customers, facilitate the work of employees and enhance their productivity and teamwork, while being able to perform, simultaneously, several of the steps and actions needed to speed up software development and delivery. All of these requirements currently met by the practice of Continuous Integration (CI) and the tools that implement it have now managed to penetrate and establish themselves, even in the applications of the largest and most popular companies in the technology and web space.

Keywords

continuous integration, continuous delivery, tests, build, continuous deployment, repository, ci pipeline, ci tools, software, application.

Περιεχόμενα

Κατάλογος Πινάκων

Κατάλογος Εικόνων

Αλφαβητικό Ευρετήριο

APA: American Psychological Association

IEEE: The Institute for Electrical and Electronics Engineers

API (Application Programming Interface): διεπαφή προγραμματισμού εφαρμογών που χρησιμοποιείται για τη μεταβίβαση δεδομένων μεταξύ εφαρμογών λογισμικού με έναν τυποποιημένο τρόπο.

AppSec (Application Security): είναι η διαδικασία εύρεσης, επιδιόρθωσης και πρόληψης ευπαθειών ασφαλείας σε επίπεδο εφαρμογής λειτουργικού, λογισμικού και διαδικασιών ανάπτυξης.

Cloud: τεχνολογία για την αποθήκευση οποιασδήποτε μορφής πληροφοριών διαδικτυακά.

Pull Request: αίτημα για αποδοχή και συγχώνευση μεταβολής (αρχείου) σε ένα σύστημα ελέγχου εκδόσεων.

QA – Quality Assurance: Τμήμα Διασφάλισης Ποιότητας (Quality Assurance – QA)

Rollback: επιστροφή σε προηγούμενη κατάσταση.

RPM Package Manager: ισχυρό σύστημα διαχείρισης πακέτων

RSS (Rich Site Summary): προτυποποιημένη μέθοδος ανταλλαγής ψηφιακού πληροφοριακού περιεχομένου διαμέσου του Διαδικτύου, στηριγμένη στην πρότυπη, καθιερωμένη και ευρέως υποστηριζόμενη γλώσσα σήμανσης XML.

Server: Διακομιστής

Shell Scripting: σενάρια κελύφους, αρχείο κειμένου που περιέχει μια σειρά εντολών.

SLAs (Service Level Agreements): σύμβαση μεταξύ ενός παρόχου και ενός τελικού χρήστη, η οποία ορίζει και δεσμεύει τον πρώτο στο απαιτούμενο επίπεδο υπηρεσιών.

Staging: είναι ένα σχεδόν ακριβές αντίγραφο ενός περιβάλλοντος παραγωγής για δοκιμές λογισμικού/

Tooltips: εργαλειοθήκες

UI (User Interface): στοιχεία διεπαφής χρήστη.

Webhook: μέθοδος επαύξησης ή αλλαγής της συμπεριφοράς μιας ιστοσελίδας ή μιας διαδικτυακής εφαρμογής με προσαρμοσμένες ανακλήσεις.

1 ΚΕΦΑΛΑΙΟ 1ο : (ΕΙΣΑΓΩΓΗ)

Σε αυτό το κεφάλαιο θα επισημανθεί το αντικείμενο, ο σκοπός και οι στόχοι της παρούσας διπλωματικής εργασίας ενώ στη συνέχεια θα παρουσιαστεί η μεθοδολογία που ακολουθήθηκε μαζί με τη δομή της χωρισμένη σε κεφάλαια.

1.1 Αντικείμενο

Την τελευταία δεκαετία, η ανάγκη των εταιρειών να αναπτύσουν τις νέες εκδόσεις των εφαρμογών τους πιο εύκολα και γρήγορα, οδήγησαν την αγορά στην δημιουργία και ενσωμάτωση νέων πρακτικών, όπως το CI.

Το CI ορίζεται ως μία πρακτική ανάπτυξης λογισμικού, στην οποία ο προγραμματιστής μίας ομάδας, συγχωνεύει κάθε αλλαγή που κάνει στον κώδικά του σε ένα κεντρικό αποθετήριο όπου η κάθε συγχώνευση, ενεργοποιεί μία αυτοματοποιημένη ακολουθία κατασκεύης κώδικα και δοκιμών, μέσα από την οποία ο προγραμματιστής είναι σε θέση να εντοπίζει άμεσα σφάλματα και συγκρούσεις μεταξύ προηγούμενων συγχωνεύσεων ή εκδόσεων άλλων προγραμματιστών της ομάδας.

Έτσι λοιπόν αρκετές εταιρείες, εντάσσοντας νέα τμήματα στις ομάδες τους, ικανά να χρησιμοποιοήσουν εξειδικευμένα εργαλεία CI, κατάφεραν να αυξήσουν σε μεγάλο βαθμό, την ταχύτητα ανάπτυξης και παράδοσης των νέων εκδόσεων σε περιβάλλον παραγωγής, αλλά και να περιορίσουν τυχών προβλήματα και σφάλματα που θα εμφανίζονταν στους τελικούς χρήστες και πελάτες τους πετυχαίνοντας έτσι μείωση του χρόνου που απαιτείται για την επικύρωση και την κυκλοφορία των νέων ενημερώσεων των εφαρμογών τους.

Αντικείμενο της παρούσας διπλωματικής εργασίας αποτελεί η ανάπτυξη διαδικτυακής δοκιμαστικής εφαρμογής μέσω της πρακτικής του CI, χρησιμοποιώντας εργαλείο ειδικά σχεδιασμένο ώστε να εφαρμόζει τις αρχές και τον ορισμό αυτής της πρακτικής.

1.2 Σκοπός και Στόχοι

Καθώς η ανάπτυξη εφαρμογών με χρήση CI από ελληνικές εταιρείες δεν είναι ακόμη τόσο διαδεδομένη και στο διαδίκτυο οι πληροφορίες προς αξιοποίηση στην ελληνική γλώσσα είναι ελάχιστες, σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη μίας δοκιμαστικής εφαρμογής με χρήση ενός εργαλείου CI, ώστε να εμπνεύσει και να βοηθήσει περισσότερες εταιρείες και ιδιώτες στην Ελλάδα να γνωρίσουν τα οφέλη του CI και να το εφαρμόσουν στην πράξη.

Συγκεκριμένα η εργασία έχει ως στόχο να απαντήσει στα παρακάτω ερωτήματα:

- Ποια είναι τα διαθέσιμα διαδικτυακά εργαλεία για την δημιουργία ενός ολοκληρωμένου αγωγού CI για την ανάπτυξη εφαρμογής σε περιβάλλον παραγωγής;
- Ποιο είναι το καλύτερο εργαλείο CI στην αγορά;

- Τα εργαλεία αυτά αρκούν για να διαικπεραιώσουν όλα τα στάδια υλοποίησης ή πρέπει να χρησιμοποιηθούν επιπλέον εργαλεία/πλατφόρμες;
- Ποια είναι πρακτικά η μεθοδολογία και τα βήματα δημιουργίας ενός αγωγού CI για την ανάπτυξη μίας εφαρμογής;
- Αρκούν τα στάδια του Continuous Integration για να αναπτυχθεί μία εφαρμογή σε περιβάλλον παραγωγής ή είναι απαραίτητη η χρήση Continuous Delivery /Continuous Deployment;
- Πως μπορεί να επιτευχθεί ανάπτυξη μίας νέας έκδοσης κώδικα σε μία ιστοσελίδα πλατφόρμας Wordpress με χρήση CI;

Για να δοθούν απαντήσεις στα παραπάνω ερωτήματα, κρίθηκε απαραίτητη, η ποιοτική ανάλυση και σύγκριση 8 σχετικών εργαλείων CI, ενώ στη συνέχεια επιλέχθηκε το ιδανικό για την επιτεύξη του σκοπού της εργασίας.

1.3 Μεθοδολογία - Υλοποίηση

Η παρούσα Διπλωματική εργασία χωρίστηκε σε 2 μέρη. Στο 1ο μέρος πραγματοποιήθηκε έρευνα σχετικά με το θεωρητικό υπόβαθρο του CI, όπου ακολουθώντας τη μεθοδολογία της κριτικής βιβλιογραφικής επισκόπησης, διατυπώθηκαν οι απαραίτητοι ορισμοί τόσο του CI, όσο και των σταδιών και πρακτικών που το αποτελούν και συνεργάζονται μαζί του αντίστοιχα , ενώ παράλληλα μελετήθηκαν οι τρόποι και μέθοδοι με τους οποίους εφαρμόζουν CI, τέσσερις από τις μεγαλύτερες και πιο κερδοφόρες εταιρείες στο διαδίκτυο, αποκτώντας έτσι μία καλύτερη αντίληψη και κατανόηση της λειτουργίας και χρήσης του CI στην πράξη. Επόμενο βήμα ήταν η αναζήτηση και έρευνα οκτώ διαφορετικών εργαλείων CI, ώστε να εντοπιστούν τα χαρακτηριστικά του καθενός, μέσα από τα οποία θα προέκυπτε τελικά το εργαλείο που θα ικανοποιούσε περισσότερο τον σκοπό της εργασίας. Πιο συγκεκριμένα έγινε η ανάλυση και η σύγκριση των εργαλείων θέτοντας ως κριτήρια επιλογής τις παραμέτρους που αναγράφονται στον Πίνακα 1.

Στο 2ο μέρος πραγματοποιήθηκε η υλοποίηση της διαδικτυακής δοκιμαστικής εφαρμογής επιλέγοντας ως κατάλληλο εργαλείο CI το Buddy και ακολουθήθηκαν τα πρακτικά όλα τα βήματα με βάση το θεωρητικό υπόβαθρο του 1^{ου} μέρους. Συγκεκριμένα, για την δημιουργία αποθετηρίου χρησιμοποιήθηκε η πλατφόρμα του Github, διαμορφώνοντας έτσι το στάδιο του Source Control που είναι και το πρώτο στάδιο για την επίτευξη της πρακτικής CI. Στη συνέχεια, για τα στάδια της κατασκευής (build) και των δοκιμών (tests) ήταν απαραίτητη η δημιουργία λογαριασμού στην διαδικτυακή πλατφόρμα του Buddy και έπειτα η δημιουργία Νέου Project και η διασύνδεσή με το αποθετήριο στο Github. Με την ολοκλήρωση της διασύνδεσης αυτής, επόμενο βήμα ήταν η δημιουργία αγωγού μέσω της πλατφόρμας του Buddy όπου προσθέτοντας τις κατάλληλες ενέργειες διαμορφώθηκαν τα στάδια Build και Continuous Testing. Τέλος, για την δημοσίευση των αλλαγών στο περιβάλλον παραγωγής, κρίθηκε αναγκαία η προέκταση του αγωγού CI προσθέτοντας σε αυτόν το στάδιο του Release μέσω της πρακτικής του Continuous Delivery όπου με την

ολοκλήρωση και των τελευταίων δοκιμών (Visual Tests), το σύστημα ενημέρωνε τον υπεύθυνο έργου μέσω email, ότι όλα τα στάδια του CI έχουν ολοκληρωθεί επιτυχώς και εκκρεμεί, ο οπτικός έλεγχος από τον ίδιο και η επιβεβαιώσή του, για να δημοσιευτεί η νέα έκδοση στον server παραγωγής.

1.4 Καινοτομία

Αναλύοντας τόσο την μεθοδολογία όσο και τη διαδικασία υλοποίησής με όσο το δυνατόν πιο απλό και αναλυτικό τρόπο, η παρούσα εργασία, θα μπορούσε να χρησιμοποιηθεί, ως ένα ελληνικό εγχειρίδιο εφαρμογής CI, ώστε να εμπνεύσει και να βοηθήσει περισσότερες εταιρείες και ιδιώτες στην Ελλάδα να γνωρίσουν τα οφέλη του CI και να το εφαρμόσουν στην πράξη.

1.5 Δομή

Δομή της διπλωματικής εργασίας (οργάνωση σε κεφάλαια και υποκεφάλαια).

Στα υπόλοιπα κεφάλαια που ακολουθούν:

- Στο κεφάλαιο 2, γίνεται η διατύπωση του θεωρητικού υποβάθρου του όρου Continuous Integration, αναλύοντας όλα τα στάδια που το απαρτίζουν, αλλά και τις πρακτικές που χρειάζονται να συνεργαστούν μαζί του ώστε να επιτευχθεί πλήρως η ανάπτυξη ενός λογισμικού. Στη συνέχεια, μέσα από παραδείγματα κορυφαίων εταιρειών του διαδικτύου που εφαρμόζουν CI για την ανάπτυξη των λογισμικών τους, επιβεβαιώνεται η σημαντικότητα της πρακτικής αυτής στις μέρες μας, ενώ το κεφάλαιο ολοκληρώνεται με την σύγκριση 8 δημοφιλών εργαλείων CI, ώστε να επιλεχθεί το κατάλληλο που θα χρησιμοποιηθεί στο κεφάλαιο 3 για τον σκοπό της παρούσας διπλωματικής εργασίας.

- Στο κεφάλαιο 3, παρουσιάζονται η μεθοδολογία και υλοποίηση μίας δοκιμαστικής εφαρμογής του εργαλείου CI που επιλέχθηκε στο 2^{ο} κεφάλαιο. Συγκεκριμένα, αναλύονται όλα τα βήματα από την προετοιμασία και τη δημιουργία του αποθετηρίου στο Github και του λογαριασμού στην πλατφόρμα Buddy, μέχρι και το τελευταίο στάδιο που απαιτείται για να ολοκληρωθεί η υλοποίηση ενός αγωγού CI μέσω του εργαλείου Buddy, ενώ ως τελευταίο βήμα για την δημοσίευση της νέας έκδοσης στην ιστοσελίδα παραγωγής, προστίθεται ως προέκταση του αγωγού και η πρακτική του Continuous Delivery.

- Στο κεφάλαιο 4 διατυπώνονται τα συμπεράσματα που εξήχθησαν από την διαδικασία υλοποίησης της δοκιμαστικής εφαρμογής και δίνονται απαντήσεις στα ερωτήματα που τέθηκαν ως στόχοι. Τέλος, καταγράφονται οι περιορισμοί που προέκυψαν κατά την διεκπεραίωση, αλλά και μερικές προτάσεις για μελλοντικά έρευνες.

2 ΚΕΦΑΛΑΙΟ 2ο : (Βιβλιογραφική Επισκόπηση)

Στο παρόν κεφάλαιο γίνεται αρχικά η διατύπωση του θεωρητικού υποβάθρου του όρου Continuous Integration, από την πρώτη του ιστορική αναφορά μέχρι και σήμερα, ενώ αναλύονται τα πλεονεκτήματα και μειονεκτήματά του.

Στη συνέχεια δίνεται ιδιαίτερη έμφαση στα στάδια αυτοματοποίησης που απαρτίζουν τον όρο CI καθώς και στα στάδια που χρειάζεται να συνεργαστούν μαζί του με σκοπό την επίτευξη δημοσίευσης ενός λογισμικού άμεσα και αυτοματοποιημένα χωρίς σφάλματα.

Έπειτα, μέσα από παραδείγματα εταιρειών που χρησιμοποιούν Continuous Integration επιβεβαιώνεται η σημαντικότητα και αναγκαιότητά του στις μέρες μας.

Τέλος, γίνεται ανάλυση και σύγκριση 8 εργαλείων με τα οποία μπορεί να δημιουργηθεί ένας αγωγός Continuous Integration και μέσα από τα πλεονεκτήματα και τα μειονεκτήματα του καθενός προκύπτει το κατάλληλο εργαλείο που θα χρησιμοποιηθεί στο κεφάλαιο 3 για την υλοποίηση Δοκιμαστικής Εφαρμογής.

2.1 Υπόβαθρο

2.1.1 Συνεχής Ενσωμάτωση (Continuous Integration - CI)

Για αρκετά χρόνια στην ευρύτερη αγορά της ανάπτυξης λογισμικού, η κύρια εστίαση δινόταν πάντα στο ίδιο το λογισμικό και στην συνεχή εξέλιξή του με σκοπό την βελτιστοποίηση αλλά και τη διαφοροποίησή του από τον ανταγωνισμό. Βέβαια, όντας πλέον σε μία περίοδο όπου ο χρόνος και το κόστος είναι δύο από τις πιο σημαντικές μεταβλητές που κρίνουν τη βιωσιμότητα και την κερδοφορία μίας επιχείρησης, η υποδομή στην οποία εκτελείται ένα λογισμικό, αποκτάει όλο και περισσότερη σημασία και προσοχή καθώς όλο και πιο συχνά φαίνεται πως είναι αυτή που καθορίζει σε μεγάλο βαθμό την πορεία και την επιτυχία του λογισμικού. Για τον λόγο αυτό λοιπόν, την τελευταία δεκαετία, έχει παρατηρηθεί η άνοδος κλάδων που σχετίζονται με την αυτοματοποίηση όσο το δυνατόν περισσότερων τμημάτων του κύκλου ζωής ενός προϊόντος λογισμικού [1].

Εικόνα 2-1 Continuous Integration

Ένας τέτοιος κλάδος είναι και το Continuous Integration, ο οποίος αποτελεί σήμερα τη βάση για την ανάπτυξη λογισμικού, καθώς είναι μία διαδικασία που μειώνει σημαντικά τόσο το χρόνο όσο και το κόστος υλοποίησής του. Βασικός στόχος του CI είναι να εντοπίζει άμεσα σφάλματα και ατέλειες του λογισμικού, ώστε να μπορούν στη συνέχεια να διορθωθούν και να αναθεωρηθούν το συντομότερο δυνατό [2].

Ταυτόχρονα, το CI επιτρέπει στους προγραμματιστές να συγχωνεύουν αυτόματα τον κώδικά τους με τον κώδικα άλλων, να ελέγχουν αν ο υποβαλλόμενος κώδικας παραβιάζει τον κώδικα των υπόλοιπων προγραμματιστών ή δημιουργεί σφάλμα στον κύριο κώδικα του λογισμικού. Αυτό μειώνει σημαντικά τη χειρωνακτική εργασία των προγραμματιστών που πρέπει να ελέγχουν τον κώδικα για εξαρτήσεις, αποτυχίες ή «σπάσιμο» κώδικα κάθε φορά που υποβάλλεται μια νέα έκδοση. Έτσι, ένας προγραμματιστής είναι πλέον σε θέση να εντοπίζει άμεσα το κάθε σφάλμα και πρόβλημα που έχει προκύψει κατά την υποβολή της νέας έκδοσης κώδικα [1].

Πριν όμως από την έλευση του CI, το μοντέλο που χρησιμοποιούνταν για την ανάπτυξη λογισμικού ήταν το μοντέλο ονόματι "Waterfall". Προϋπόθεση για την χρήση του συγκεκριμένου μοντέλου ήταν ότι από το στάδιο του σχεδιασμού ενός λογισμικού έπρεπε όλες οι απαιτήσεις να έχουν οριστικοποιηθεί εξαρχής καθώς δεν υπήρχε περιθώριο για μεταγενέστερες αλλαγές. Έπειτα, το στάδιο της ανάπτυξης, το στάδιο των δοκιμών αλλά και το στάδιο του build, έπρεπε να δρομολογηθούν ως ξεχωριστές εργασίες εφόσον δεν ήταν εφικτό να συμβούν παράλληλα με αποτέλεσμα ο κύκλος ζωής της κάθε έκδοσης ενός λογισμικού να έχει πολύ μεγάλη χρονική διάρκεια. Επιπλέον, το έργο της συγχώνευσης (merge) και δοκιμής (test) μίας έκδοσης ήταν μία αρκετά περίπλοκη και χρονοβόρα διαδικασία, καθώς ο κώδικας κάθε προγραμματιστή έπρεπε να συγχωνευθεί ξεχωριστά χωρίς πρώτα να έχει ελεγχθεί αν είναι συμβατός με τους κώδικες των υπόλοιπων προγραμματιστών της ομάδας, και ενώ μπορεί μεμονομένα να εμφάνιζε μηδενικά σφάλματα, μετά την συγχώνευση, συνήθως προκαλούσε σφάλματα που θα επηρέαζαν την λειτουργικότητα είτε του ίδιου είτε άλλων τμημάτων του λογισμικού [3].

Αυτά ήταν και μερικά από προβλήματα που αντιμετώπιζε και ο Grady Booch το 1991, όταν εφάρμοσε για πρώτη φορά τις αρχές του Continuous Integration, χωρίς όμως να εφαρμόσει 100% τη διαδικασία, αφού δεν το χρησιμοποίησε για την κατασκευή και αυτοματοποίηση ολόκληρης της υλοποίησής του. Αργότερα, το 1994, ο όρος CI καταγράφηκε και επίσημα για πρώτη φορά από τον ίδιο τον Booch στο βιβλίο του με τίτλο «Object-Oriented Analysis and Design with Applications» [2,4].

Η πλήρης εφαρμογή του CI συνέβη 3 χρόνια αργότερα, συγκεκριμένα το 1997, όπου ο Kent Beck και ο Ron Jeffries κατάφεραν να εφαρμόσουν εξ'ολοκλήρου την έννοια του CI, ενώ εργάζονταν στο έργο Chrysler Comprehensive Compensation System Project. Επινόησαν ένα πλαίσιο που ονομάστηκε Extreme programming (XP), το οποίο περιλάμβανε το CI ως πρακτική και υποστήριζε ένα πλήρες continuous building.

Δύο χρόνια μετά, το 1999, ο Beck δημοσίευσε το πρώτο βιβλίο για για το CI, ενώ το Cruise Control ήταν το πρώτο εργαλείο CI ανοιχτού κώδικα, το οποίο κυκλοφόρησε το 2001. Βέβαια, η αναγνωρισιμότητα της πρακτικής CI αυξήθηκε με την ανάπτυξη και την κυκλοφορία του εργαλείου Hudson CI, όπου λόγω της μεγάλης φήμης που είχε αποκτήσει στον τεχνολογικό τομέα αναγκάστηκε να το 2011 να μετονομαστεί σε Jenkins, μετά από διαμάχη με την Oracle, η οποία διεκδικούσε τα δικαιώματα του ονόματος [2].

2.1.2 Πλεονεκτήματα CI

Την τελευταία δεκαετία οι πρακτικές CI έχουν επηρεάσει θετικά τον ρυθμό με τον οποίο μπορούν να παραδοθούν τα προϊόντα λογισμικού βελτιώνοντας παράλληλα και την ποιότητά τους [5].

Με την υποστήριξη αυτοματοποιημένων διαδικασιών build, test και deploy, ένα από τα κύρια πλεονεκτήματα του CI είναι ότι οι οργανισμοί είναι σε θέση να διοχετεύουν περισσότερες δυνατότητες στους αγωγούς παραγωγής και παράδοσης ενώ ο αυτοματισμός κατάφερε να μειώσει επίσης την προσπάθεια που απαιτείται για τη ρύθμιση των εκδόσεων, δίνοντας τη δυνατότητα στους οργανισμούς να αναδίδουν τις εκδόσεις όσο συχνά απαιτείται. Επιπροσθέτως, έχει διαπιστωθεί ότι υψηλότερα επίπεδα αυτοματισμού οδηγούν σε βελτιωμένη διασφάλιση ποιότητας.

Ένας αυτοματοποιημένος αγωγός CI βοηθά στο να διασφαλιστεί ότι κάθε αλλαγή επαληθεύεται πριν προωθηθεί για παράδοση. Επειδή λοιπόν κάθε αλλαγή στον κώδικα ελέγχεται σε κάθε στάδιο της ανάπτυξης και τα σφάλματα ανακαλύπτονται και επιλύονται αμέσως, τα τελικά προϊόντα έχουν μηδενικά σφάλματα και το λογισμικό μπορεί να δημοσιευθεί εύκολα και γρήγορα.

Μία από τις σημαντικότερες θετικές επιπτώσεις του CI είναι ότι αναγκάζει τις ομάδες ανάπτυξης και χειρισμού υποδομών να αλληλεπιδρούν μεταξύ τους περισσότερο από πριν, κάτι που φαίνεται ότι οδηγεί σε μία αρκετά βελτιωμένη συνεργασία και επικοινωνία. Η αυξημένη συνεργασία επιταχύνει την ανταλλαγή γνώσεων και εμπειριών μεταξύ των ομάδων δημιουργώντας έτσι δυναμικές και λειτουργικές ομάδες με ποικίλες δυνατότητες που ενισχύουν την παραγωγικότητα και έχουν ως αποτέλεσμα τη μεγιστοποίηση των ικανοτήτων αφού έτσι αξιοποιείται ένα ευρύτερο σύνολο δεξιοτήτων [6].

Τέλος, λόγω των συχνών εκδόσεων, οι ομάδες ανάπτυξης και διαχείρισης είναι σε θέση να λαμβάνουν άμεσα σχόλια από τους τελικούς χρήστες και να μπορούν να κάνουν δοκιμές με πραγματικούς πελάτες, κάτι που βοηθά στη βελτίωση των τελικών προϊόντων. Αυτή λοιπόν η αλληλεπίδραση δίνει τη δυνατότητα στις εταιρείες να κάνουν εύκολα και σε πραγματικό χρόνο τα κατάλληλα tests και να ενημερωθούν άμεσα σχετικά με τις προτιμήσεις των πελατών. Αυτό στη συνέχεια, έχει ως αποτέλεσμα η εταιρεία να είναι σε θέση να προσαρμόσει τα προϊόντα της ώστε να ανταποκρίνονται ακριβώς στις απαιτήσεις της αγοράς.

Γι' αυτό λοιπόν, μία αποτελεσματική διαδικασία CI υποστηρίζει και επιτρέπει τον συνεχή πειραματισμό ο οποίος μπορεί να βοηθήσει τις εταιρείες να δοκιμάσουν γρήγορα διαφορετικές ιδέες και να λαμβάνουν ανάλογα τις αντίστοιχες αποφάσεις [5,6].

2.1.3 Μειονεκτήματα CI

Όσα πολλά και αν είναι τα πλεονεκτήματα, πάντα υπάρχουν και παράγοντες οι οποίοι μπορούν να οδηγήσουν στη λάθος υλοποίηση και εφαρμογή των πρακτικών ενός συστήματος CI.

Ένα από τα βασικά μειονεκτήματα που έχουν παρατηρηθεί σε αρκετές εταιρείες είναι η ανάθεση των εργασιών μεταξύ των ομάδων. Από τη μία οι προγραμματιστές πρέπει να αναλάβουν νέες εργασίες στις οποίες δεν είναι συνηθισμένοι, ενώ από την άλλη ενδέχεται να έχουν επιφυλάξεις σχετικά με την ανάθεση κάποιων ευθυνών και αρμοδιοτήτων τους σε νέα τμήματα που ίσως δεν έχουν την ίδια εξειδίκευση και εμπειρία [6].

Ταυτόχρονα, σε συγκεκριμένες περιπτώσεις, όπως για παράδειγμα σε ένα περιβάλλον μίας βάσης δεδομένων, μπορεί να χρησιμοποιηθούν συστήματα παραγωγής τα οποία μπορεί να είναι αρκετά περίπλοκα με αποτέλεσμα να κάνουν δύσκολη την αναπαραγωγή των δεδομένων της βάσης για την διαδικασία της επαλήθευσης και της δοκιμής. Κατά συνέπεια, οι αυτοματοποιημένες δοκιμές γίνονται λιγότερο αξιόπιστες με αποτέλεσμα ένα σύστημα CI να μην είναι πάντα επιτυχημένο.

Τέλος, το CI βασίζεται σε ποικιλία εργαλείων για την κατασκευή ενός αγωγού. Η ενσωμάτωση αυτών των εργαλείων μπορεί να αποδειχθεί προβληματική και δύσκολη καθώς ενδέχεται να υπάρχουν διαφοροποιήσεις στις εκδόσεις και τις ρυθμίσεις των εργαλείων που χρησιμοποιούν οι προγραμματιστές σε σχέση με τους χειριστές υποδομών. Τα εργαλεία αυτά λοιπόν θα πρέπει να λειτουργούν πάντα με βάση τις ίδιες εκδόσεις και ρυθμίσεις, διαφορετικά θα υπάρχουν σημαντικά προβλήματα διασύνδεσης μεταξύ τους [5,6].

2.2 Στάδια CI

Το CI εφαρμόζεται σε ένα περιβάλλον εξαιρετικά συντονισμένης ομαδικής λειτουργίας. Έχοντας λοιπόν, ένα τέλειο σενάριο συνεχών δοκιμών (continuous tests) και παραδόσεων, οι προγραμματιστές διαμορφώνουν ο καθένας ξεχωριστά το δικό τους τμήμα κώδικα για την ανάπτυξη μίας εφαρμογής και έπειτα, όλοι ανεξαιρέτως, οφείλουν να το προσθέσουν σε ένα κοινό αυτοματοποιημένο σύστημα διαχείρισης εκδόσεων (Version Control) ώστε μην υπάρξει κίνδυνος διαφορετικού ιστορικού μεταξύ των αρχείων κώδικα, ενώ ακόμη και στην περίπτωση που υπάρξει, να έχουν ανά πάσα στιγμή τη δυνατότητα να επαναφέρουν αυτόματα τα αρχεία στις σωστές εκδόσεις [9].

Μόλις λοιπόν δημιουργηθεί μία νέα έκδοση κώδικα στο αποθετήριο ανάπτυξης, στη συνέχεια αναλαμβάνει ένας μηχανικός CI, ο οποίος έχοντας δημιουργήσει έναν

μηχανισμό Continuous Integration ενημερώνει αυτόματα τον server του εργαλείου CI, ότι υπάρχει νέα έκδοση στο αποθετήριο και έτσι, με αυτοματοποιημένο τρόπο, το εργαλείο CI ανεβάζει τη νέα έκδοση κώδικα στον server και ξεκινάει το στάδιο του build. Παράλληλα, μόλις ανέβουν και τα αντίστοιχα αρχεία δοκιμών (test cases) στο αποθετήριο ποιοτικού ελέγχου, ο μηχανισμός CI ενημερώνεται αυτόματα και ξεκινάει αυτοματοποιημένα το στάδιο του testing φορτώνοντας τα test cases στον CI server και κάνοντας τους τελικούς δοκιμαστικούς ελέγχους (unit tests) [9,10].

Εικόνα 2-2 Στάδια Continuous Integration

Αφού ολοκληρωθούν αυτές οι διαδικασίες ανάλογα με την πολιτική της κάθε εταιρείας, επιλέγεται αν θα δημιουργηθεί ένα σύστημα Continuous Delivery, δηλαδή εάν η εφαρμογή θα περάσει σε στάδιο release για να αποφασιστεί από την εταιρεία πότε θα περάσει στο στάδιο deploy ή αν θα δημιουργηθεί σύστημα Continuous Deployment όπου η εφαρμογή θα δημοσιευτεί απευθείας σε περιβάλλον παραγωγής αυτόματα και χωρίς ανθρώπινη παρέμβαση [10].

Τέλος, οι βασικοί παράγοντες που θα αξιολογηθούν για να χαρακτηριστεί ένα σύστημα CI επιτυχημένο είναι:

- Η βελτίωση στην ποιότητα ανάπτυξης του λογισμικού
- Η συχνότητα κυκλοφορίας λογισμικού
- Η ανταπόκριση στις επιχειρηματικές ανάγκες
- Η ποιότητα του κώδικα
- Η ταχύτητα με την οποία επιδιορθώνονται τα σφάλματα.
- Η ευπάθεια των αναδυόμενων υποδομών σε διαδικτυακές επιθέσεις [1].

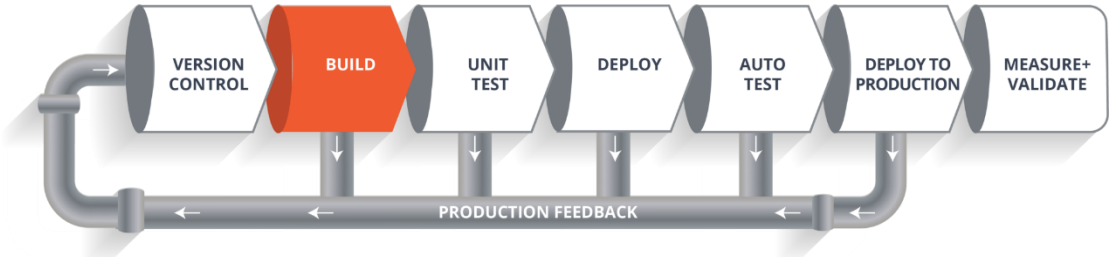
2.2.1 Έλεγχος Έκδοσης (Version Control)

Το Version Control είναι η διαδικασία παρακολούθησης κάθε έκδοσης κώδικα ή δεδομένων με τρόπο που διευκολύνει την παρακολούθηση του ιστορικού των αλλαγών και την επιστροφή στις προηγούμενες εκδόσεις ανά πάσα στιγμή, καθώς και την κοινή χρήση κώδικα με τους συνεργάτες. Αυτό ισχύει επίσης για το σύνολο δεδομένων που χρησιμοποιείται κατά τη διάρκεια κάθε επαναληπτικού κύκλου και πρέπει να παρακολουθούμε αυτά τα δεδομένα σε κάθε προσαύξηση προκειμένου να αναπαράγουμε τα ίδια αποτελέσματα στο μέλλον εάν χρειαστεί. Σε αυτό το βήμα, τυχόν αλλαγές στον πηγαίο κώδικα του μοντέλου ή αλλαγές στην έκδοση του συνόλου δεδομένων που χρησιμοποιείται θα προωθηθούν στο χώρο αποθήκευσης, ώστε να ενεργοποιηθεί στη συνέχεια η διαδικασία του build μέσω του αγωγού CI [1,11].

Η πιο διαδεδομένη πλατφόρμα διαχείρισης εκδόσεων, είναι το Git, το οποίο επιτρέπει σε κάθε προγραμματιστή να διατηρεί το δικό του τοπικό αποθετήριο και να προωθεί αλλαγές στο κεντρικό αποθετήριο ή να αντλεί αιτήματα από τον κεντρικό κόμβο, ενώ οι παλαιότερες εκδόσεις του κώδικα είναι πάντα διαθέσιμες όποτε χρειαστούν [11].

2.2.2 Συνεχής Κατασκευή (Continuous Building)

Στο στάδιο του Continuous Building, ο κώδικας κατασκευάζεται σε περιβάλλον ανάπτυξης ώστε να είναι δυνατή η δοκιμή και η διόρθωση σφαλμάτων.



Εικόνα 2-3 Continuous Building

Πιο συγκεκριμένα, μπορούμε να το χωρίσουμε σε τρία γενικά μέρη:

Βήμα 1: Μεταγλώτισση

Το πρώτο βήμα του σταδίου build μεταγλωττίζει την εφαρμογή. Στις μεταγλωττισμένες γλώσσες, αυτό σημαίνει ότι μπορούμε να δημιουργήσουμε ένα λειτουργικό δυαδικό αρχείο, ενώ στις διερμηνευμένες γλώσσες, επιβεβαιώνουμε ότι διαθέτουμε τις απαιτούμενες εξαρτήσεις και τα εργαλεία για ένα επιτυχημένο build της εφαρμογής.

Προγράμματα γραμμένα σε γλώσσες όπως η Java, η C/C++ ή η Go είναι αναγκαίο να μεταγλωττιστούν, ενώ προγράμματα σε γλώσσες όπως η Ruby, η Python και η JavaScript μπορούν να λειτουργήσουν και χωρίς αυτό το βήμα.

Βήμα 2: Ανάλυση Κώδικα

Ο προγραμματισμός απαιτεί την επίλυση δύσκολων προβλημάτων ακολουθώντας σωστές πρακτικές και τηρώντας ένα κοινό στυλ κωδικοποίησης. Σε γενικές γραμμές, υπάρχουν τρεις κατηγορίες εργαλείων ανάλυσης κώδικα:

Ανίχνευση (Linting): τα linters βελτιώνουν την ποιότητα του κώδικα υποδεικνύοντας προβληματικά και δύσκολα στη συντήρηση κομμάτια. Τα linters χρησιμοποιούν ένα σύνολο κανόνων για την επιβολή ενός ενιαίου προτύπου που βελτιώνει την αναγνωσιμότητα για την ομάδα.

Ποιότητα: αυτή η κατηγορία εργαλείων χρησιμοποιεί μετρικές για να βρει σημεία όπου ο κώδικας μπορεί να βελτιωθεί. Οι μετρικές που συλλέγονται περιλαμβάνουν τον αριθμό των γραμμών, την κάλυψη τεκμηρίωσης και το επίπεδο πολυπλοκότητας.

Ασφάλεια: τα εργαλεία ανάλυσης ασφάλειας σαρώνουν τον κώδικα, επισημαίνουν τμήματα που ενδέχεται να προκαλέσουν ευπάθειες και ελέγχουν ότι οι εξαρτήσεις δεν έχουν γνωστά προβλήματα ασφάλειας.

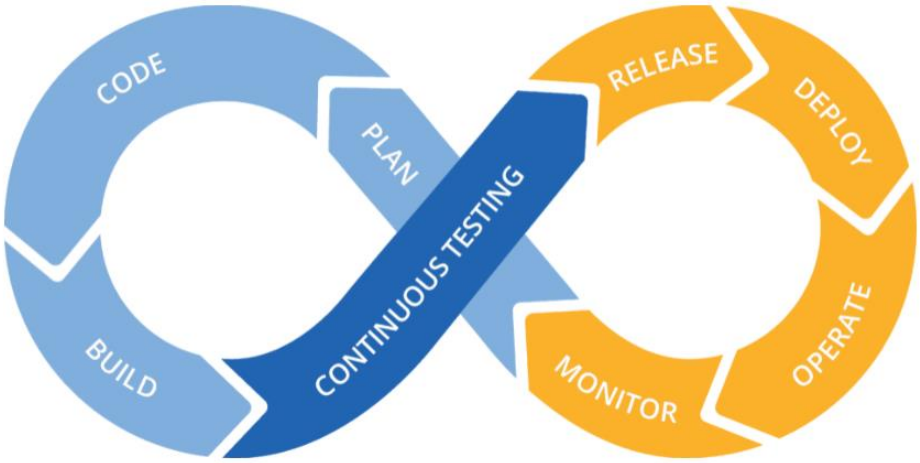
Βήμα 3: Δημιουργία Πακέτου Εφαρμογής

Το τρίτο και τελευταίο βήμα συσκευάζει την εφαρμογή για έκδοση ή ανάπτυξη. Σε αυτό το βήμα, όλα τα αρχεία και οι εξαρτήσεις του αγωγού μεταφέρονται σε ένα πακέτο κοντέινερ. Αυτό το βήμα πραγματοποιείται συνήθως χρησιμοποιώντας έναν διαχειριστή πακέτων όπως το Helm ή το Docker [1,12,13].

Τέλος, δεδομένου ότι κανείς δεν θέλει να στέλνει μη ασφαλή ή προβληματικό κώδικα, ο αγωγός CI πρέπει να σχεδιαστεί με τρόπο που να σταματά όταν εντοπίζονται σφάλματα. Η αποτυχία του αγωγού να περάσει το στάδιο του build αποτελεί ένδειξη ενός θεμελιώδους προβλήματος στη διαμόρφωση ενός έργου γι'αυτό θα πρέπει να λαμβάνετε σοβαρά υπόψη και να αντιμετωπίζεται άμεσα [14].

2.2.3 Συνεχής Δοκιμή (Continuous Testing)

Αυτό είναι το στάδιο όπου το αναπτυγμένο λογισμικό ελέγχεται συνεχώς για σφάλματα. Για continuous tests, χρησιμοποιούνται εργαλεία ελέγχου αυτοματισμού όπως Selenium, TestNG, JUnit κλπ. Αυτά τα εργαλεία επιτρέπουν στους Υπεύθυνους Διασφάλισης Ποιότητας (Quality Assurance – QA) να ελέγχουν πολλές βάσεις κώδικα ενδελεχώς, διασφαλίζοντας παράλληλα ότι δεν υπάρχουν ελαττώματα στη λειτουργικότητα. Σε αυτή τη φάση, συνήθως χρησιμοποιούνται τα λεγόμενα Δοχεία Docker (Docker Containers) ώστε να γίνει προσομοίωση του περιβάλλοντος δοκιμής [1].

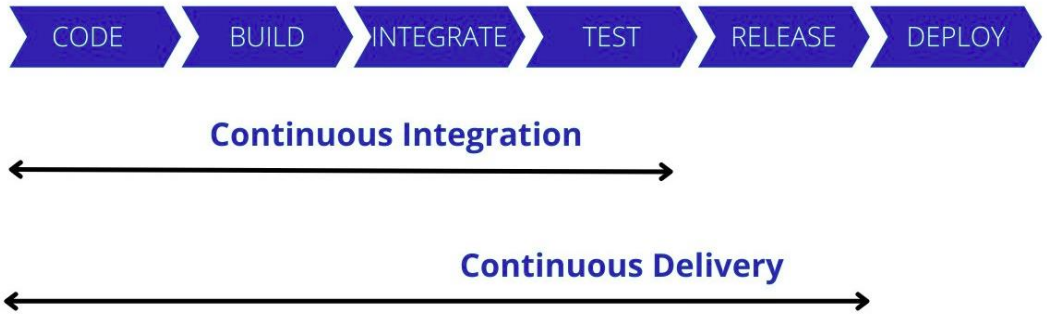


Εικόνα 2-4 Continuous Testing

Ολόκληρη αυτή η δοκιμαστική φάση μπορεί να αυτοματοποιηθεί με τη βοήθεια ενός εργαλείου CI. Τα Continuous Tests μέσω κάποιου εργαλείου CI εξοικονομούν πολύ χρόνο, προσπάθεια και εργασία για την εκτέλεσή τους ειδικά σε μακροπρόθεσμο πλαίσιο καθώς ένα μεγάλο μέρος της διαδικασίας χρειάζεται να δημιουργηθεί μία φορά και στη συνέχεια να χρησιμοποιείται επ'αόριστο. Έτσι, το έργο της αξιολόγησης των περιπτώσεων που απέτυχαν τα tests γίνεται πιο απλό και η εκτέλεση των δοκιμαστικών περιπτώσεων μπορεί να γίνεται ακόμη και σε προκαθορισμένους χρόνους ενώ μετά από κάθε, ο κώδικας θα ενσωματώνεται συνεχώς με τον υπάρχοντα κώδικα [15].

2.2.4 Συνεχής Παράδοση (Continuous Delivery - CD)

Το Continuous Delivery είναι ένας μηχανισμός ο οποίος ακολουθεί όλες τις πρακτικές και τα στάδια του Continuous Integration αλλά για να εφαρμοστεί πλήρως, χρειάζεται ακόμη ένα στάδιο, αυτό της κυκλοφορίας (release).



Εικόνα 2-5 Στάδια Continuous Delivery

Είναι μια πρακτική ανάπτυξης λογισμικού όπου οι αλλαγές κώδικα προετοιμάζονται αυτόματα για κυκλοφορία στην παραγωγή. Ένας πυλώνας της σύγχρονης ανάπτυξης εφαρμογών, το Continuous Delivery επεκτείνεται με τη συνεχή ενσωμάτωση με την ανάπτυξη όλων των αλλαγών κώδικα σε ένα περιβάλλον δοκιμής ή σε ένα περιβάλλον παραγωγής μετά το στάδιο build. Όταν εφαρμοστεί σωστά, οι προγραμματιστές θα έχουν πάντα ένα τεχνούργημα έτοιμο για ανάπτυξη που έχει περάσει από μια τυποποιημένη διαδικασία δοκιμής [1].

Το CD επιτρέπει στους προγραμματιστές να αυτοματοποιούν τα tests πέρα από τα δοκιμές μονάδων (unit tests), ώστε να μπορούν να επαληθεύουν τις ενημερώσεις εφαρμογών σε πολλές διαστάσεις πριν από την ανάπτυξη σε πελάτες. Αυτές οι δοκιμές μπορεί να περιλαμβάνουν δοκιμή διεπαφής χρήστη, δοκιμή φόρτωσης, δοκιμή ενσωμάτωσης, δοκιμή αξιοπιστίας API κλπ. Αυτό βοηθά τους προγραμματιστές να επικυρώνουν πιο διεξοδικά τις ενημερώσεις και να ανακαλύπτουν προληπτικά ζητήματα. Με το cloud, είναι εύκολο και οικονομικά αποδοτικό να αυτοματοποιηθεί η δημιουργία και η αναπαραγωγή πολλαπλών περιβαλλόντων για tests, κάτι που προηγουμένως ήταν δύσκολο να γίνει επί τόπου.

Οποιαδήποτε διαδικασία πρέπει να επαναληφθεί με την πάροδο του χρόνου θα πρέπει να αυτοματοποιηθεί. Η διαδικασία παράδοσης λογισμικού θα πρέπει να μπορεί να εκτελεί όλες τα tests σε κάθε έκδοση λογισμικού που δημιουργείται αυτόματα χωρίς καμία παρέμβαση του χρήστη, οδηγώντας έτσι προς τον απώτερο στόχο της γρήγορης αποστολής μιας ποιοτικής έκδοσης . Όλη αυτή η αρχή των continuous tests όχι μόνο μετακινεί τη διαδικασία των tests στην αρχή του κύκλου, αλλά επιτρέπει επίσης τη διεξαγωγή των tests σε σύστημα παρόμοιο με την παραγωγή (συμπληρωμένο με συνεχή ανάπτυξη) [16,17].

Για να δημοσιευτεί μία εφαρμογή σε περιβάλλον παραγωγής μέσω Continuous Delivery, θα πρέπει να κάνουμε τη συγκεκριμένη διαδικασία χειροκίνητα. Οι

περισσότερες εταιρείες, συνηθίζουν να επιλέγουν ως πρακτική το Continuous Delivery, καθώς τους δίνει τη δυνατότητα να κάνουν τους τελικούς ελέγχους που θέλουν, ώστε να αισθάνονται μεγαλύτερη σιγουριά και ασφάλεια, ότι το προϊόν τους θα δημοσιευτεί στο περιβάλλον παραγωγής όπως πρέπει.

Ένας ακόμη λόγος που οι εταιρείες προτιμούν τη διαδικασία του Continuous Delivery είναι η βελτιστοποίηση που παρέχει και στη διαχείριση της υποδομής αλλά και στην ανάγκη εξισορρόπησης χρόνου και πόρων. Η επικύρωση των προϊόντων λογισμικού περιλαμβάνει πολλαπλές μεταβλητές εισόδους, όπως εκδόσεις προϊόντων, διαφορετικά λειτουργικά συστήματα, διαφορετικές εκδόσεις λογισμικού, κλπ. Πιο συγκριμένα λοιπόν, η διαδικασία ενός συστήματος CD ξεκινά με ένα πακέτο εφαρμογών που ετοιμάζει το σύστημα CI, το οποίο θα εγκατασταθεί σύμφωνα με μια λίστα αυτοματοποιημένων εργασιών [16].

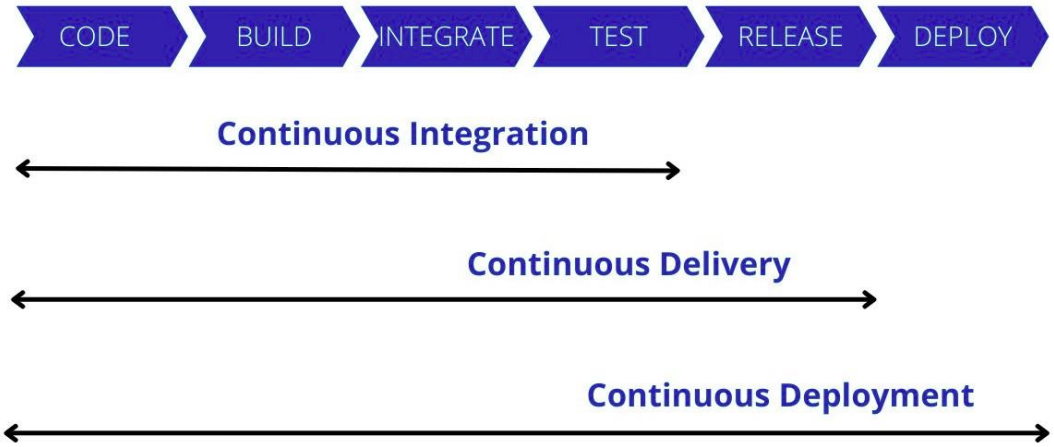
Αυτές οι εργασίες μπορούν να είναι οποιουδήποτε τύπου: αποσυμπίεση, διακοπή και επανεκκίνηση της υπηρεσίας, αντιγραφή αρχείων, αντικατάσταση διαμόρφωσης και ούτω καθεξής. Η εκτέλεση λειτουργικών δοκιμών (functional tests) και δοκιμών αποδοχής (acceptance tests) μπορεί επίσης να πραγματοποιηθεί κατά τη διαδικασία του CD. Σε αντίθεση με το CI, το CD στοχεύει να δοκιμάσει ολόκληρη την εφαρμογή με όλες τις εξαρτήσεις της.

Αυτό είναι πολύ ορατό σε εφαρμογές μικροϋπηρεσιών που αποτελούνται από πολλές υπηρεσίες και API. Το CI θα δοκιμάσει μόνο την υπό ανάπτυξη μικρουπηρεσία, ενώ, μόλις αναπτυχθεί, θα είναι δυνατό να δοκιμαστεί και να επικυρωθεί ολόκληρη η εφαρμογή καθώς και τα API και οι μικροϋπηρεσίες από τις οποίες αποτελείται. Στην πράξη, σήμερα, είναι πολύ συνηθισμένο να συνδέουμε το CI με το CD σε ένα περιβάλλον ολοκλήρωσης.

Είναι πράγματι απαραίτητο ώστε οι προγραμματιστές να μπορούν να έχουν σε κάθε δέσμευση όχι μόνο την εκτέλεση unit tests αλλά και επαλήθευση της εφαρμογής στο σύνολό της (UI και λειτουργική), με την ενσωμάτωση των εξελίξεων των άλλων μελών της ομάδας. Είναι πολύ σημαντικό το πακέτο που δημιουργείται κατά τη διάρκεια του CI και που θα αναπτυχθεί κατά τη διάρκεια του CD να είναι το ίδιο που θα εγκατασταθεί σε όλα τα περιβάλλοντα, και αυτό θα πρέπει να ισχύει μέχρι την παραγωγή. Ωστόσο, ενδέχεται να υπάρχουν μετασχηματισμοί αρχείων διαμόρφωσης που διαφέρουν ανάλογα με το περιβάλλον, αλλά ο κώδικας της εφαρμογής (δυαδικά, DLL και JAR) πρέπει να παραμείνει αμετάβλητος. Αυτός ο αμετάβλητος, αμετάβλητος χαρακτήρας του κώδικα είναι η μόνη εγγύηση ότι η εφαρμογή που επαληθεύεται σε ένα περιβάλλον θα είναι της ίδιας ποιότητας με την έκδοση που έχει αναπτυχθεί στο προηγούμενο περιβάλλον και την ίδια που θα αναπτυχθεί στο επόμενο περιβάλλον. Εάν πρόκειται να γίνουν αλλαγές (βελτιώσεις ή διορθώσεις σφαλμάτων) στον κώδικα μετά την επαλήθευση σε ένα από τα περιβάλλοντα, μόλις γίνει, η τροποποίηση θα πρέπει να περάσει ξανά από τον κύκλο CI και CD [1,17].

2.2.5 Συνεχής Ανάπτυξη (Continuous Deployment - CD)

Το Continuous Deployment είναι ο μηχανισμός ο οποίος αυτοματοποιεί τη διαδικασία από την ανάπτυξη του κώδικα μέχρι και τη δημοσίευσή του στο περιβάλλον παραγωγής. Είναι επίσης η διαδικασία με την οποία κάθε αλλαγή στον κώδικα λογισμικού ή την αρχιτεκτονική δημοσιεύεται αυτοματοποιημένα στην παραγωγή χωρίς ανθρώπινη παρέμβαση [1].



Εικόνα 2-6 Στάδια Continous Deployment

Η πρακτική του Continuous Deployment προχωρά ένα στάδιο μετά το τελευταίο στάδιο του Continuous Delivery συνεχή παράδοση. Είναι ουσιαστικά μια διαδικασία που αυτοματοποιεί ολόκληρο τον αγωγό CI/CD από τη στιγμή που ο προγραμματιστής δεσμεύει τον κώδικά του στην ανάπτυξη σε όλα τα βήματα επαλήθευσης. Αυτή η πρακτική εφαρμόζεται σπάνια σε επιχειρήσεις επειδή απαιτεί ευρεία κάλυψη δοκιμών (μονάδας, λειτουργικότητας, αποδοχής, απόδοσης κλπ.) για την εφαρμογή και η επιτυχής εκτέλεση αυτών των tests είναι αρκετή για να επικυρώσει την ορθή λειτουργία της εφαρμογής με όλες αυτές οι εξαρτήσεις, αλλά και αυτοματοποιημένη ανάπτυξη σε περιβάλλον παραγωγής χωρίς καμία ενέργεια έγκρισης.

Η συνεχής ανάπτυξη είναι ένας εξαιρετικός τρόπος για να επιταχύνετε τον κύκλο των σχολίων με τους πελάτες σας και να αποβάλλετε την πίεση από την ομάδα, καθώς δεν υπάρχει πλέον ημέρα κυκλοφορίας. Οι προγραμματιστές μπορούν να επικεντρωθούν στην κατασκευή λογισμικού και βλέπουν τη δουλειά τους να βγαίνει ζωντανά λίγα λεπτά μετά την ολοκλήρωση της εργασίας τους σε αυτό.

Η διαδικασία Continuous Deployment πρέπει επίσης να λαμβάνει υπόψη όλα τα βήματα για την επαναφορά της εφαρμογής σε περίπτωση προβλήματος παραγωγής. Η συνεχής ανάπτυξη μπορεί να υλοποιηθεί με τη χρήση και την εφαρμογή τεχνικών εναλλαγής χαρακτηριστικών, που περιλαμβάνει την ενθυλάκωση (encapsulation) των λειτουργιών της εφαρμογής και την ενεργοποίηση των δυνατοτήτων της κατά παραγγελία, απευθείας στην παραγωγή, χωρίς να χρειάζεται να ανανεωθεί ο κώδικας της εφαρμογής [18].

2.3 Εταιρείες που Εφαρμόζουν CI Σήμερα

Τα μεταβαλλόμενα επιχειρηματικά μοντέλα και ο αυξημένος ανταγωνισμός έχουν οδηγήσει σε ραγδαία αύξηση του ενδιαφέροντος για το CI την τελευταία δεκαετία ενώ κάθε χρόνο όλο και περισσότερες εταιρείες ενδιαφέρονται να βελτιώσουν την ταχύτητα και την ποιότητα της ανάπτυξης λογισμικού. Οι 4 παρακάτω οργανισμοί έχουν σημειώσει τεράστια εξέλιξη και επιτυχία στους μετασχηματισμούς CI που πραγματοποίησαν.

2.3.1 Amazon

Η Amazon φημίζεται για το εντυπωσιακό της σύστημα CI. Η μετάβαση στο CI αποτέλεσε μέρος της μετακίνησης της εταιρείας το 2010 από τους φυσικούς διακομιστές στο cloud της Amazon Web Services (AWS), μια αλλαγή που της επέτρεψε να εξοικονομήσει πόρους με την κλιμάκωση της χωρητικότητας σε βήματα ενός μόνο διακομιστή. Η Amazon άρχισε επίσης να χρησιμοποιεί μια διαδικασία CI που διαχειρίζεται ένα εσωτερικό σύστημα που ονομάζεται Apollo, το οποίο δίνει στους προγραμματιστές τη δυνατότητα να αναπτύσσουν κώδικα όποτε και όπου θέλουν. Μέχρι τον Μάιο του 2011, ανέπτυσσε νέο λογισμικό σε διακομιστές παραγωγής κατά μέσο όρο κάθε 11,6 δευτερόλεπτα ενώ στις ώρες αιχμής, 1.079 νέες αναπτύξεις στέλνονταν στο περιβάλλον παραγωγής.

Το CI έχει σχεδιαστεί για να αυξάνει τη συχνότητα της ανάπτυξης, αλλά οι πιο επιτυχημένες εταιρείες γνωρίζουν ότι η ποιότητα και η διαχείριση των κινδύνων είναι εξίσου σημαντικές με την ταχύτητα. Η εφαρμογή ενός νέου εξισορροπιστή φορτίου από την Amazon εξασφάλισε ότι μόνο μία από τις 100.000 αναπτύξεις θα μπορούσε να οδηγήσει σε διακοπή λειτουργίας στον ιστότοπο της Amazon. Σύμφωνα με τον John Jenkins, πρώην επικεφαλής μηχανικό της Amazon, η μετάβαση στο ευέλικτο CI εξοικονομεί πλέον εκατομμύρια [23].

Ένας μηχανικός της AWS έγραψε πρόσφατα για το πώς μοιάζουν οι αγωγοί ανάπτυξης της Amazon και ποιες πρακτικές ακολουθούνται για τη συνεχή ανάπτυξη σε ένα περιβάλλον παραγωγής. Ένας αγωγός επικυρώνει τις αλλαγές σε πολλαπλά προπαραγωγικά περιβάλλοντα που εκτελούν unit tests και χρησιμοποιούν στάδια για την κλιμάκωση των αναπτύξεων στην παραγωγή. Οι προγραμματιστές δεν εξετάζουν ενεργά τις αναπτύξεις, καθώς ο αγωγός παρακολουθεί βασικές μετρήσεις και μπορεί να κάνει rollback αν χρειαστεί. Επίσης, οι προγραμματιστές διαμορφώνουν τους αγωγούς τους ως κώδικα που μπορεί να κληρονομήσει κοινές διαμορφώσεις.

Στο στάδιο του build, ο αγωγός μεταγλωττίζει τον κώδικα, εκτελεί unit tests, εκτελεί στατική ανάλυση, αξιολογεί την κάλυψη του κώδικα και αποθηκεύει τα τεχνουργήματα. Στο στάδιο των tests, ο αγωγός επικυρώνει τις αλλαγές σε πολλαπλά προπαραγωγικά περιβάλλοντα, όπως το άλφα, το βήτα και το γάμμα- και εδώ είναι που εξασκούνται στις αναπτύξεις σε διαφορετικές ζώνες και περιοχές. Τέλος, η ομάδα χωρίζει τον αγωγό σε μικρότερα στάδια ανάπτυξης για να περιορίσει τον αντίκτυπο σε περίπτωση αποτυχίας. Πριν από τη μετάβαση στο επόμενο στάδιο, ο αγωγός

παρακολουθεί την υπηρεσία για κάποιο χρονικό διάστημα για να διασφαλίσει τη διαθεσιμότητά της [24].

Όταν ένας προγραμματιστής ολοκληρώνει την κωδικοποίηση και έχει επικυρώσει τις αλλαγές σε έναν προσωπικό λογαριασμό AWS χρησιμοποιώντας unit tests, κάποιος άλλος στην ομάδα πρέπει να τις επανεξετάσει. Ο αναθεωρητής βεβαιώνεται ότι ο κώδικας είναι σωστός, ότι υπάρχουν αρκετές δοκιμές, ότι είναι επαρκώς ενορχηστρωμένος και ότι οι αλλαγές είναι συμβατές προς τα πίσω ώστε να διασφαλίζεται η επαναφορά. Οι αναθεωρήσεις κώδικα είναι η τελευταία χειροκίνητη διαδικασία και ένα κρίσιμο βήμα, διότι όταν ο κώδικας καταλήξει στον κύριο κλάδο, ο αγωγός ανάπτυξης ενεργοποιείται αυτόματα. Αν και, μπορεί να υπάρξουν φορές που χρειάζονται πρόσθετες εγκρίσεις πριν από τη λειτουργία, όπως μια αλλαγή στη διεπαφή χρήστη.

Η Clare Liguori, επικεφαλής μηχανικός λογισμικού από την Amazon Web Services μοιράστηκε περισσότερες λεπτομέρειες σχετικά με τη διαδικασία αναθεώρησης σε προσωπική συνέντευξη που είχε δώσει τον Ιούλιο του 2020:

"Κάνουμε ανάπτυξη βασισμένοι στον κορμό, οπότε ο αγωγός ασχολείται μόνο με τον κύριο κλάδο. Σπάνια δημιουργούμε κλάδους. Ένα εργαλείο αυτόματης δημιουργίας αντιγράφων ασφαλείας ωθεί τις τοπικές μας δεσμεύσεις σε ένα απομακρυσμένο git ref, και η υποβολή αναθεώρησης κώδικα δημιουργεί ένα απομακρυσμένο git ref των προτεινόμενων αλλαγών μας, ενώ οι αναθεωρήσεις κώδικα εκτελούν unit tests [23,24].

Επιπλέον, ορισμένοι αγωγοί ανάπτυξης μπορούν να έχουν χρονικά παράθυρα για να υποδεικνύουν πότε μπορεί να ξεκινήσει μια ανάπτυξη ώστε να μειωθεί ο κίνδυνος πρόκλησης δυσμενών επιπτώσεων. Σύμφωνα με την Liguori, αυτό το σύνολο των χρονικών παραθύρων τους βοηθά να έχουν "λιγότερη εμπλοκή με τους μηχανικούς που εφημερεύουν εκτός των κανονικών ωρών εργασίας και να διασφαλίζουν ότι ένας μικρός αριθμός αλλαγών ομαδοποιείται μαζί, όσο τα χρονικά παράθυρα είναι κλειστά" [25].

2.3.2 Netflix

Το Netflix θεωρείται εδώ και καιρό ηγέτης στην παροχή περιεχομένου που μπορεί να μεταδοθεί σε απευθείας σύνδεση ως υπηρεσία. Ένας από τους βασικούς παράγοντες διαφοροποίησης στην παροχή της υπηρεσίας online streaming είναι η δυναμική υποδομή τους. Αυτή η υποδομή τους επιτρέπει να διατηρούν τα SLAs ανεξάρτητα από ενημερώσεις λογισμικού, δραστηριότητες συντήρησης ή απροσδόκητες προκλήσεις του συστήματος.

Οι προκλήσεις που έπρεπε να αντιμετωπίσουν για τη δημιουργία μιας δυναμικής υποδομής απαιτούσαν την ανάπτυξη μιας pluggable αρχιτεκτονικής. Αυτή η αρχιτεκτονική έπρεπε να υποστηρίζει καινοτομίες από την κοινότητα προγραμματιστών και να κλιμακώνεται για να προσεγγίσει νέες αγορές. Έπρεπε να υποστηρίζονται περιβάλλοντα πολλαπλών περιοχών με εξελιγμένες στρατηγικές

ανάπτυξης - στρατηγικές που επέτρεπαν μπλε/πράσινες αναπτύξεις, καναρίνια έκδοσης και ροές εργασίας CI/CD. Τα εργαλεία και το μοντέλο CI που προέκυψαν επέκτειναν και επηρέασαν επίσης τη βασική τους δραστηριότητα της δημιουργίας, επικύρωσης και ανάπτυξης περιεχομένου [26].

Το Netflix λοιπόν, λειτουργεί μια υποδομή βασισμένη στο cloud που αποτελείται από εκατοντάδες μικρουπηρεσίες. Οι προγραμματιστές μπορούν να κατασκευάζουν αυτόματα κομμάτια κώδικα σε εικόνες ιστού που μπορούν να αναπτυχθούν χωρίς να βασίζονται σε λειτουργίες πληροφορικής. Καθώς οι εικόνες ενημερώνονται, ενσωματώνονται στην υποδομή του Netflix χρησιμοποιώντας μια προσαρμοσμένη, βασισμένη στο διαδίκτυο πλατφόρμα. Υπάρχει αυτοματοποιημένη παρακολούθηση, έτσι ώστε εάν η ανάπτυξη των εικόνων αποτύχει, οι νέες εικόνες ανατρέπονται και η κυκλοφορία ανακατευθύνεται στην παλιά έκδοση.

Η επιτυχία του Netflix στο CI δεν πέρασε απαρατήρητη. Το 2015, όταν η εταιρεία επιλέχθηκε ομόφωνα για το Ειδικό Βραβείο της Κριτικής Επιτροπής JAX, ο συντάκτης του JAXenter Coman Hamilton δήλωσε: "Ο ρυθμός με τον οποίο αυτή η εταιρεία αλλάζει το παιχνίδι της ψυχαγωγίας, έχει υιοθετήσει νέες τεχνολογίες και τις έχει εφαρμόσει στην προσέγγιση CI θέτοντας νέα πρότυπα στην πληροφορική" [27].

Το εργαλείο Jenkins αποτελεί κεντρικό μέρος της υποδομής δημιουργίας και ανάπτυξης του Netflix εδώ και αρκετά χρόνια και χρησιμοποιείται ως εργαλείο για συνεχή ενσωμάτωση και ανάπτυξη. Μόλις μια γραμμή κώδικα έχει κατασκευαστεί και δοκιμαστεί τοπικά με τη χρήση του Nebula (Μια συλλογή από πρόσθετα Gradle που έχει κατασκευαστεί από το Netflix). Το πρώτο βήμα είναι να προωθηθεί ο ενημερωμένος πηγαίος κώδικας σε ένα αποθετήριο git. Μόλις γίνει η δέσμευση της αλλαγής, ενεργοποιείται μια εργασία Jenkins. Το Netflix ξεκίνησε με ένα μόνο τεράστιο Jenkins master στο κέντρο δεδομένων του και τώρα τρέχει 25 Jenkins master στο AWS. Το Jenkins Job ρυθμίζεται ώστε να καλεί το Nebula για τη δημιουργία, τη δοκιμή και το build του κώδικα της εφαρμογής. Εάν το αποθετήριο που κατασκευάστηκε είναι μια βιβλιοθήκη, το Nebula δημοσιεύει το .jar στο αποθετήριο τεχνουργημάτων και εάν το αποθετήριο που κατασκευάστηκε είναι μια εφαρμογή, τότε θα εκτελεστεί το plugin ospackage(operation system package) του Nebula. Χρησιμοποιώντας το plugin ospackage του Nebula, το τεχνούργημα κατασκευής μιας εφαρμογής θα ομαδοποιηθεί είτε σε ένα πακέτο Debian είτε σε ένα πακέτο RPM, του οποίου τα περιεχόμενα ορίζονται μέσω μιας απλής DSL (Domain Specific Language) βασισμένης στο Gradle [28].

Πώς το Netflix χτίζει κώδικα χρησιμοποιώντας το Jenkins;

- Ο κώδικας χτίζεται και δοκιμάζεται τοπικά χρησιμοποιώντας το Nebula
- Οι αλλαγές δεσμεύονται σε ένα κεντρικό αποθετήριο git
- Μια εργασία Jenkins εκτελεί το Nebula, το οποίο χτίζει, δοκιμάζει και πακετάρει την εφαρμογή για ανάπτυξη
- Τα builds "ψήνονται" σε Amazon Machine Images
- Οι σωληνώσεις Spinnaker χρησιμοποιούνται για την ανάπτυξη και την προώθηση της αλλαγής του κώδικα
- Παρατηρήστε πώς οι διαφορετικές τεχνολογίες συνεργάζονται αρμονικά για να μας παρέχουν τελική υπηρεσία όπως το Netflix. Έτσι, απώτερος στόχος είναι η ενσωμάτωση των διαφόρων τεχνολογιών [29].

2.3.3 Facebook

Το Facebook χρησιμοποιούσε τη συνεχή ανάπτυξη ήδη από το 2005. Το Facebook διαθέτει χιλιάδες μηχανικούς και ένα σύνολο υπηρεσιών που χρησιμοποιούνται από τουλάχιστον ένα δισεκατομμύριο χρήστες, με τους backend servers να πρέπει να επεξεργάζονται δισεκατομμύρια ερωτήματα ανά δευτερόλεπτο. Οι προγραμματιστές, αντί να σχεδιάζουν την εργασία σε έργα ή να την χωρίζουν σε χρονικά καθορισμένα Sprints, κάνουν το μεγαλύτερο μέρος της εργασίας τους σε ανεξάρτητες, μικρές αλλαγές που κυκλοφορούν συχνά. Αυτό έχει νόημα στο διαδικτυακό επιχειρηματικό μοντέλο του Facebook, όλοι συντονίζουν συνεχώς την πλατφόρμα και δοκιμάζουν νέες επιλογές και εφαρμογές σε διαφορετικές κοινότητες χρηστών, βλέποντας τι μένει. Είναι προς τιμήν της αρχιτεκτονικής τους ότι τόσες πολλές μικρές, ανεξάρτητες αλλαγές μπορούν πραγματικά να γίνουν ανεξάρτητα και φτηνά [30].

Το Facebook λέει ότι ακολουθεί τη Συνεχή Ανάπτυξη, αλλά η αλήθεια πως δεν πρόκειται για Continuous Deployment, δηλαδή κάθε αλλαγή να μεταφέρεται αμέσως στους πελάτες. Στο Facebook, ο κώδικας μπορεί να κυκλοφορεί δύο φορές την ημέρα, αλλά αυτό γίνεται κυρίως για διορθώσεις σφαλμάτων και εσωτερικό κώδικα. Ο νέος κώδικας παραγωγής κυκλοφορεί μία φορά την εβδομάδα: χιλιάδες αλλαγές από εκατοντάδες προγραμματιστές συσκευάζονται από μία μικρή ομάδα έκδοσης τις Κυριακές, περνούν από αυτοματοποιημένες δοκιμές παλινδρόμησης (regression tests) και κυκλοφορούν την Τρίτη, αν οι προγραμματιστές που συνέβαλαν στις αλλαγές είναι παρόντες. Οι μηχανικοί έκδοσης αξιολογούν τον κίνδυνο των αλλαγών με βάση το μέγεθος της αλλαγής, τον όγκο της συζήτησης που γίνεται στις αναθεωρήσεις κώδικα (η οποία καταγράφεται μέσω ενός εσωτερικού εργαλείου αναθεώρησης κώδικα) και το πόσα προβλήματα κώδικα έχουν δει στο παρελθόν. Ένα εργαλείο που ονομάζεται "Gatekeeper" ελέγχει ποια χαρακτηριστικά είναι διαθέσιμα σε ποιους πελάτες για την υποστήριξη της σκοτεινής εκτόξευσης, και όλος ο κώδικας κυκλοφορεί σταδιακά - στο staging, στη συνέχεια σε ένα υποσύνολο χρηστών, και ούτω καθεξής. Οι αλλαγές μπορούν να ανακληθούν εάν είναι απαραίτητο - μεμονωμένα ή, ως έσχατη λύση, μια ολόκληρη έκδοση κώδικα [31].

Ένα βασικό στοιχείο της κουλτούρας στο Facebook είναι ότι οι προγραμματιστές είναι ατομικά υπεύθυνοι για τον κώδικα που έγραψαν, για τη δοκιμή του και την υποστήριξή του στην παραγωγή. Το Facebook δεν διαθέτει ανεξάρτητη ομάδα δοκιμών, επειδή, όπως λέει, δεν χρειάζεται. Πρώτον, εξαρτώνται πολύ από τις αναθεωρήσεις κώδικα για την εύρεση σφαλμάτων. Κάθε γραμμή κώδικα που γράφεται αναθεωρείται από διαφορετικό μηχανικό από τον αρχικό συγγραφέα. Αυτό εξυπηρετεί πολλαπλούς σκοπούς: ο αρχικός μηχανικός έχει κίνητρο να διασφαλίσει ότι ο κώδικας είναι υψηλής ποιότητας, ο αναθεωρητής έρχεται με φρέσκο μυαλό και μπορεί να βρει ελαττώματα ή να προτείνει εναλλακτικές λύσεις και, γενικά, η γνώση σχετικά με τις πρακτικές κωδικοποίησης και τον ίδιο τον κώδικα διαδίδεται σε όλη την εταιρεία. Οι προγραμματιστές είναι επίσης υπεύθυνοι για τη συγγραφή unit tests και των δικών τους regression tests και αυτοματοποιημένων δοκιμών απόδοσης (performance tests).[30]

Το Facebook εξαρτάται επίσης από τους πελάτες που δοκιμάζουν το λογισμικό για λογαριασμό του. Το λογισμικό κυκλοφορεί σταδιακά για A/B tests και "ζωντανό πειραματισμό" σε υποσύνολα της βάσης χρηστών, είτε οι πελάτες θέλουν να συμμετάσχουν σε αυτές τις δοκιμές είτε όχι. Επειδή η πελατειακή του βάση είναι τόσο μεγάλη, μπορεί να λάβει ουσιαστική ανατροφοδότηση από τα testsακόμη και με ένα μικρό ποσοστό χρηστών, γεγονός που ελαχιστοποιεί τουλάχιστον τον κίνδυνο και την ταλαιπωρία των πελατών [32].

Τέλος, ενώ οι επιδόσεις είναι ένα σημαντικό στοιχείο για τους προγραμματιστές του Facebook, δεν υπάρχει καμία αναφορά σε ελέγχους ασφαλείας ή δοκιμές πουθενά σε αυτή την περιγραφή του τρόπου με τον οποίο το Facebook αναπτύσσει και αναπτύσσει λογισμικό. Δεν υπάρχει κάποια ανάλυση ή εξήγηση του τρόπου με τον οποίο η ομάδα ασφαλείας και οι προγραμματιστές συνεργάζονται, ούτε καν για τον "ευαίσθητο στην προστασία της ιδιωτικής ζωής κώδικα" - αν και αυτός ο κώδικας "υπόκειται σε υψηλότερα πρότυπα", δεν εξηγείται ποιο είναι αυτό το "υψηλότερο πρότυπο". Πιθανότατα βασίζεται στη χρήση βιβλιοθηκών και πλαισίων για να χειριστεί τουλάχιστον ορισμένα προβλήματα AppSec, και ενδεχομένως για να αναζητήσει σφάλματα ασφαλείας στις αναθεωρήσεις κώδικα [31,33].

2.3.4 Etsy

Στα πρώτα χρόνια της λειτουργίας του, το Etsy πάλευε με αργές, επώδυνες αναπτύξεις. Η εταιρεία είχε πολλά σιλό και ομάδες που δεν συνεργάζονταν καλά. Παρόλο που οι αναπτύξεις της Etsy δύο φορές την εβδομάδα ήταν στην πραγματικότητα πιο συχνές από τις περισσότερες του κλάδου εκείνη την εποχή, ένιωθαν ότι θα μπορούσαν να είναι ταχύτερες. Η Etsy άρχισε να επιτρέπει στους προγραμματιστές να αναπτύσσουν τον δικό τους κώδικα γύρω στα τέλη του 2009. Σύμφωνα με τον πρώην αντιπρόεδρο τεχνικών λειτουργιών Michael Rembetsy, σε συνέντευξή του στο Network World το 2015, συνειδητοποίησαν ότι όταν οι προγραμματιστές αισθάνονται την ευθύνη για την ανάπτυξη, θα αναλάμβαναν επίσης την ευθύνη για την απόδοση της εφαρμογής, τον χρόνο διαθεσιμότητας και άλλους στόχους [34].

Το Etsy αναπτύσσει πλέον πάνω από 60 φορές την ημέρα. Ο Rembetsy λέει ότι η κουλτούρα μάθησης αποτελεί βασικό μέρος του CI στο Etsy, ακόμη και μια αποτυχία μπορεί να είναι επιτυχία, αν αντληθούν διδάγματα.

Συγκεκριμένα, αναφέρει ότι η δουλειά ενός μηχανικού (ως υπεύθυνος λειτουργίας, CI, QA, υποστήριξη κλπ) είναι να ενεργοποιεί τους επιχειρηματικούς στόχους. Για να επιτευχθεί αυτό θα πρέπει όλοι να έχουν την ικανότητα να αναπτύσσουν κώδικα γρήγορα και με ασφάλεια. Ακόμα και αν οι επιχειρηματικοί στόχοι είναι να αναπτυχθεί κώδικας με αυστηρά QA, η ύπαρξη αξιόπιστης και εύκολης ανάπτυξης πρέπει να είναι αδιαπραγμάτευτη.

Μόλις λοιπόν η ανάπτυξη γίνεται εύκολα και γρήγορα, πολλά πράγματα μπορούν να αλλάξουν. Οι αλλαγές μπορούν να δημοσιεύονται κομμάτι-κομμάτι και όχι όλες μαζί, ενώ οι επιλογές διαμόρφωσης της εφαρμογής, μπορούν να βρίσκονται στον κώδικα - και να αλλάζουν γρήγορα.

Στο Etsy, μόλις ο κώδικάς είναι έτοιμος, ακολουθεί το λεγομένο Deployinator το οποίο οδηγείται απευθείας στο QA. Όταν η εταιρεία έφερε για πρώτη φορά το Deployinator στο διαδίκτυο, ήταν απλώς ένα web frontend για τα shell scripts που μετακινούσαν τα πάντα στη σωστή θέση. Αυτό που τελικά απέφερε κέρδος βάζοντας μια οθόνη μπροστά του ήταν η δυνατότητα επανάληψης του backend χωρίς να αλλάξει η εμπειρία για τους ανθρώπους που κάνουν deploying. Το Deployinator χρησιμοποιεί επί του παρόντος το svn για την ενημέρωση του κώδικα και στη συνέχεια το rsync για τη μετακίνησή του μεταξύ των περιβαλλόντων.

Ένα άλλο σημαντικό μέρος του Deployinator είναι ότι τα περιβάλλοντα είναι μονόδρομος. Ο κώδικας που πηγαίνει στην παραγωγή δεν επηρεάζεται από οποιεσδήποτε μεταβιβάσεις έτσι ώστε κάθε φορά όλοι να είναι ενήμεροι και να γνωρίζουν το τι ακριβώς αναπτύσσεται χωρίς δυσάρεστες εκπλήξεις [35].

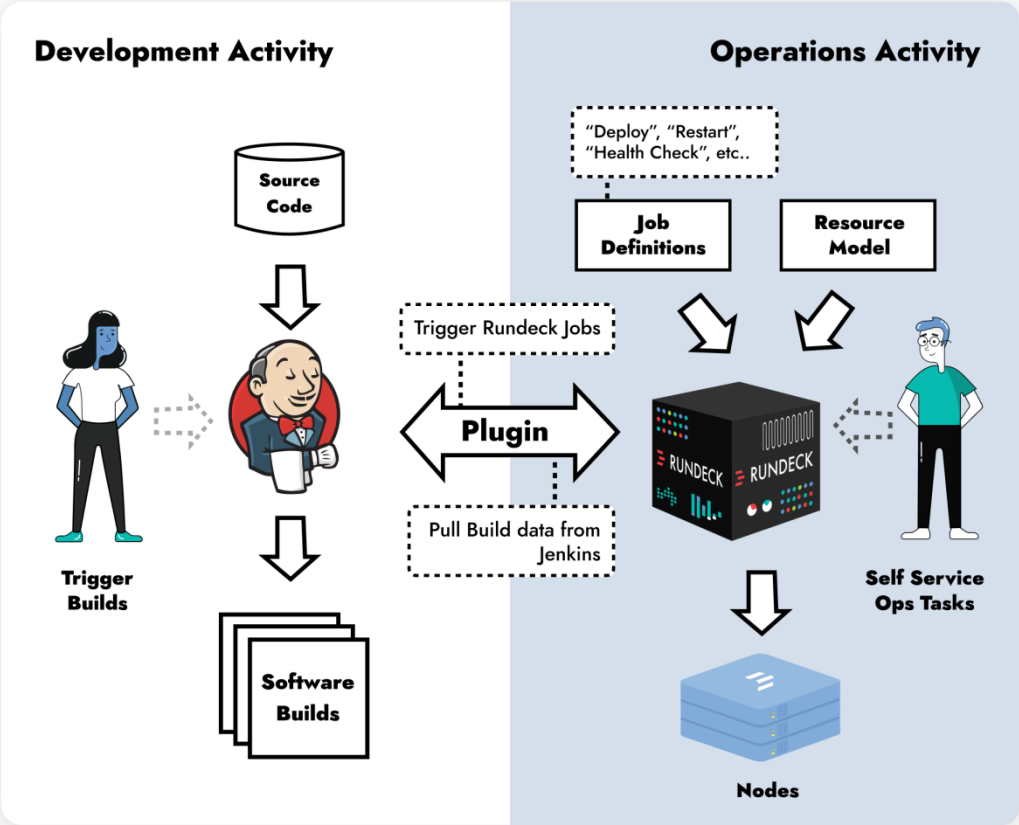
2.4 Εργαλεία CI

Λόγω της ραγδαίας ανάπτυξης και της μεγάλης ποικιλίας επιλογών στην αγορά της τεχνολογίας πληροφοριών, η επιλογή των σωστών εργαλείων γίνεται πολύ δύσκολη υπόθεση. Συχνά, τα κριτήρια δεν είναι σαφή και δεν ορίζονται στην αρχή της διαδικασίας επιλογής εργαλείων για την ανάπτυξη CI. Είναι σημαντικό λοιπόν να γνωρίζουμε ποιες ιδιότητες του CI απαιτούνται κατά την επιλογή των εργαλείων. Λόγω αυτού του προβλήματος, σε αυτό το κεφάλαιο, γίνεται αναφορά στα κριτήρια, τις απαιτήσεις και τα χαρακτηριστικά των εργαλείων CI με σκοπό να καταλήξουμε στο εργαλείο που θα περιλαμβάνει τα περισσότερα θετικά στοιχεία μέσα από έναν συγκριτικό πίνακα, ο οποίος θα παρουσιαστεί στο τέλος του κεφαλαίου [3,36].

2.4.1 Εργαλείο Jenkins

Το Jenkins είναι μια πλατφόρμα CI ανοικτού κώδικα. της οποίας ο αρχικός στόχος ήταν η αυτοματοποίηση των διαδικασιών build και test. Το σύστημα build είναι πλήρως

γραμμένο σε Java και είναι εύκολα επεκτάσιμο χάρη στην αρχιτεκτονική των πρόσθετων (plugins). Αυτό καθιστά το Jenkins ένα εξαιρετικά παραμετροποιημένο και ευέλικτο εργαλείο, ικανό να καλύψει πολλά πιθανά σενάρια και απαιτήσεις χάρη στα χιλιάδες πρόσθετα που αναπτύσσονται από την τεράστια κοινότητα ανοικτού κώδικα [37].



Εικόνα 2-7 Εργαλείο Jenkins

Το Jenkins αναπτύχθηκε αρχικά από τον Kohsuke Kawaguchi ενώ εργαζόταν στη Sun Microsystems το 2004, ωστόσο εκείνη την εποχή ονομαζόταν Hudson. Μετά την αρχική του κυκλοφορία το 2005, έγινε γρήγορα ένα από τα αγαπημένα συστήματα κατασκευής ανοικτού κώδικα server. Το 2010, άρχισαν να προκύπτουν ζητήματα μεταξύ της ανοικτής κοινότητας που εργάζεται στο Hudson, και της Oracle (η οποία είχε εξαγοράσει τη Sun). Τελικά, μετά από μία ψηφοφορία για το αν θα έπρεπε ή όχι να διασπαστεί το project, η συντριπτική πλειοψηφία της κοινότητας όντας υπέρ της διάσπασης έκανε το Jenkins και επίσημα ανεξάρτητο.

Η πλειοψηφία αυτών που εργάζονταν ή χρησιμοποιούσαν το Hudson εκείνη την εποχή, τελικά μετέβησαν στο Jenkins. Αυτή τη στιγμή υπάρχουν τουλάχιστον 127.000 εγκαταστάσεις του Jenkins. Εκτός του ότι είναι ανοικτού κώδικα, το Jenkins είναι εύκολο στην εγκατάσταση και εξαιρετικά παραμετροποιήσιμο μέσω της διεπαφής ιστού. Ενώ το Jenkins προσφέρει πολλά από μόνο του, είναι επίσης εξαιρετικά επεκτάσιμο μέσω του plug-ins του εργαλείου. Επί του παρόντος, διαθέτει 1350+

πρόσθετα από 580+ συνεργάτες, για την εκτέλεση μιας πληθώρας διαφορετικών εργασίων, επιτρέποντας σε πολλά εργαλεία τρίτων να αξιοποιήσουν το Jenkins [38].

Από τη συνεχή ολοκλήρωση στη συνεχή παράδοση με Jenkins

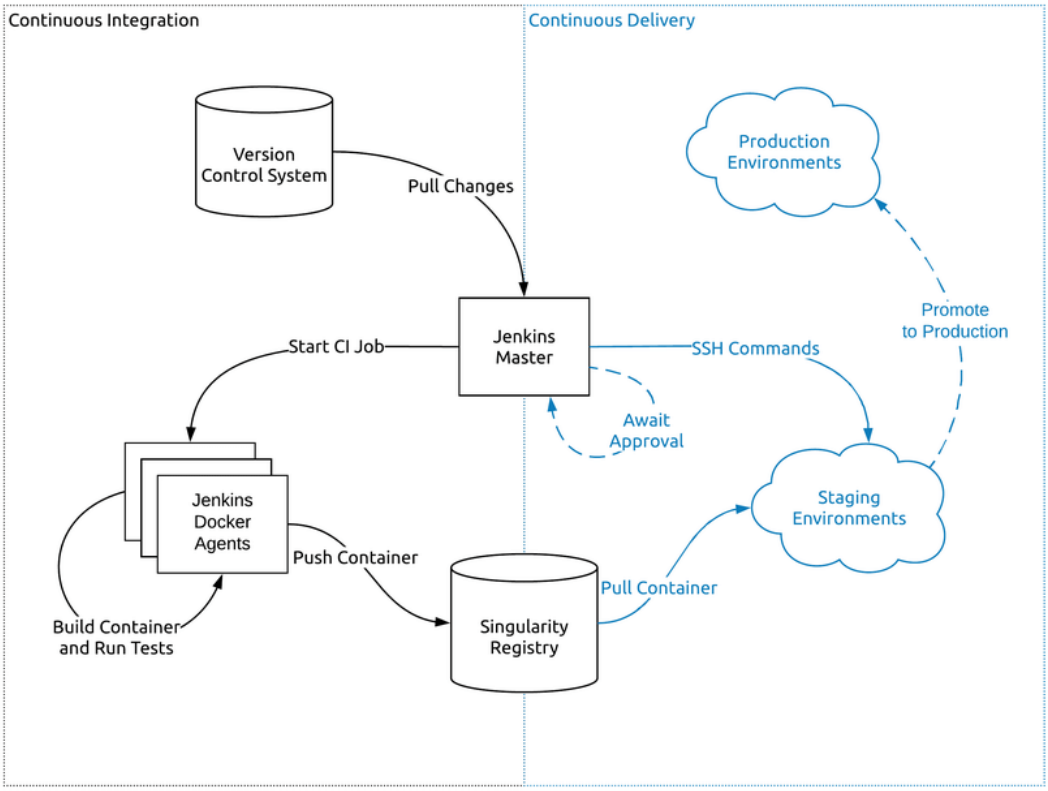
Ο παραδοσιακός τρόπος διαχείρισης της ανάπτυξης λογισμικού, γνωστός ως μοντέλο καταρράκτη, είναι ένα λογικό και διαδοχικό μοντέλο το οποίο έχει το μεγάλο ελάττωμα ότι δεν εφαρμόζεται εύκολα στην πραγματικότητα: μία από τις βασικές έννοιες αυτού του μοντέλου είναι ο ορισμός των παγωμένων απαιτήσεων σε προκαταρκτική και απομονωμένη κατάσταση. Αυτό το προκαταρκτικό βήμα θα δρομολογούσε τους καταρράκτες των επόμενων βημάτων, τα οποία θα μπορούσαν να διαρκέσουν εβδομάδες ή μήνες για να που περιλαμβάνουν σχεδιασμό, υλοποίηση, επαλήθευση και ανάπτυξη. Κατά τη διάρκεια του χρόνου που απαιτείται για να διανύσει το σύνολο των βημάτων, οι απαιτήσεις πιθανόν να αλλάξουν, λόγω της αστάθεια της αγοράς, σε νέους ανταγωνιστές, ή απλώς σε μια βελτίωση του αρχικού έργου.

Συνειδητοποιώντας ότι η παραδοσιακή προσέγγιση αποτύγχανε πολύ συχνά, το 2001 μια μικρή ομάδα ανθρώπων συγκεντρώθηκε για να συζητήσει τις νέες αξίες που απαιτούσε η ανάπτυξη λογισμικού. Έκτοτε, οι εταιρείες έχουν προχωρήσει πολλά βήματα προς έναν "ευέλικτο μετασχηματισμό", προσπαθώντας να είναι πάντα πιο δυναμικές, ανταγωνιστικές και να ανταποκρίνονται στις αλλαγές από ποτέ άλλοτε.

Το CI άρχισε κυριολεκτικά να αλλάζει τον τρόπο με τον οποίο οι εταιρείες αντιμετώπιζαν τη διαχείριση του build, του source control, του deploy και την ενορχήστρωση των tests και εμφανίστηκαν διάφορα εργαλεία για να εκπληρώσουν αυτές τις νέες προσδοκίες. Μεταξύ αυτών, το Jenkins, που γεννήθηκε ως μικρό ερασιτεχνικό έργο του K. Kawaguchi, έγινε ξαφνικά το πιο χρησιμοποιούμενο εργαλείο CI, χάρη στην αρχιτεκτονική των πρόσθετων και τη φύση του ως OSS, που παρακίνησε πολλούς προγραμματιστές σε όλο τον κόσμο να συνεισφέρουν στα χαρακτηριστικά του. Έτσι, χάρη στην ελαστικότητα και την ευελιξία του, το Jenkins μπόρεσε να προσαρμόζεται συνεχώς στις νέες ανάγκες, παρέχοντας πάντα περισσότερες λειτουργίες και ενσωματώσεις με εξωτερικά εργαλεία και τεχνολογίες, εκπληρώνοντας πλήρως τη συνεχιζόμενη επανάσταση του CI [39].

Ωστόσο, αν το CI ξεκίνησε ως αυτοματοποίηση μόνο της φάσης ανάπτυξης, πολύ σύντομα η επανάσταση αγκάλιασε τη φάση δοκιμής της ομάδας QA και την ανάπτυξη σε διάφορα περιβάλλοντα από την ομάδα των Ελεγκτών (Operators), εμπλέκοντας ολόκληρο τον κύκλο ζωής ενός προϊόντος και εισάγοντας μια νέα αντίληψη: Continuous Delivery (CD). Το Jenkins βελτιώθηκε και πάλι και επεκτάθηκε, και από εργαλείο CI που ήταν αποκλειστικά και μόνο, μετατράπηκε σε πλατφόρμα CD, επιτρέποντας στις Dev, QA και Ops να συνεργάζονται στενά χρησιμοποιώντας τον ίδιο ενορχηστρωτή και υιοθετώντας τα βασικά σημεία του CD:

- Συνεργασία μεταξύ όλων των ομάδων που εμπλέκονται στην κύκλο ζωής του προϊόντος
- Εκτεταμένη αυτοματοποίηση της διαδικασίας παράδοσης



Εικόνα 2-8 CI/CD με Jenkins

Οφέλη του Jenkins

- Το Jenkins είναι ένα εργαλείο ανοικτού κώδικα που είναι εξαιρετικά εύκολο στην εγκατάσταση και τη χρήση. Δεν χρειάζεστε επιπλέον εξαρτήματα για να το χρησιμοποιήσετε
- Είναι δωρεάν και διαθέσιμο για χρήση με διάφορες πλατφόρμες, όπως Windows, Linux, macOS και άλλες
- Χρησιμοποιείται ευρέως, οπότε η εύρεση υποστήριξης σε διαδικτυακές κοινότητες δεν αποτελεί μεγάλο πρόβλημα
- Αυτοματοποιεί όλες τις εργασίες ενσωμάτωσης. Τα ζητήματα ενσωμάτωσης είναι σπάνια και, έτσι, βοηθά στην εξοικονόμηση χρόνου και χρημάτων κατά τη διάρκεια του κύκλου ζωής ενός έργου.
- Είναι εύκολο στη διαμόρφωση, την επέκταση και την τροποποίηση. Επιτρέπει την άμεση δημιουργία tests καθώς και την κατασκευή, αυτοματοποίηση και ανάπτυξη κώδικα σε διαφορετικές πλατφόρμες
- Το Jenkins μπορεί να ρυθμιστεί ώστε να εκτελεί σωστά τις έννοιες CI και CD
- Μπορεί εύκολα να εντοπίσει και να διορθώσει προβλήματα.
- Υποστηρίζει μια ποικιλία από πρόσθετα, τα οποία επιτρέπουν μεγαλύτερη ευελιξία και οργάνωση.

- Βοηθά στον εντοπισμό σφαλμάτων πολύ νωρίς, εξοικονομώντας έτσι στους προγραμματιστές αρκετό χρόνο τον οποίο μπορούν να αξιοποιήσουν στην περαιτέρω ανάπτυξη ενός έργου [37,39].

2.4.2 Εργαλείο TravisCI

Το TravisCI είναι μια αποδεδειγμένη λύση CI που είναι κατάλληλη για έργα ανοιχτού κώδικα. Αυτή η πλατφόρμα απρόσκοπτης ενσωμάτωσης παρέχει μια σειρά επιλογών αυτοματοποίησης CI. Δεδομένου το TravisCI φιλοξενεί μια υπηρεσία cloud, δεν απαιτείται διακομιστής. Το TravisCI διατίθεται επίσης σε έκδοση που βασίζεται σε επιχειρήσεις. Ένα από τα σπουδαιότερα πράγματα με αυτό το εργαλείο είναι ότι κάθε φορά που εκτελείται μία νέα έκδοση, υποστηρίζει το πιο πρόσφατο build.

Διαμόρφωση του Travis CI

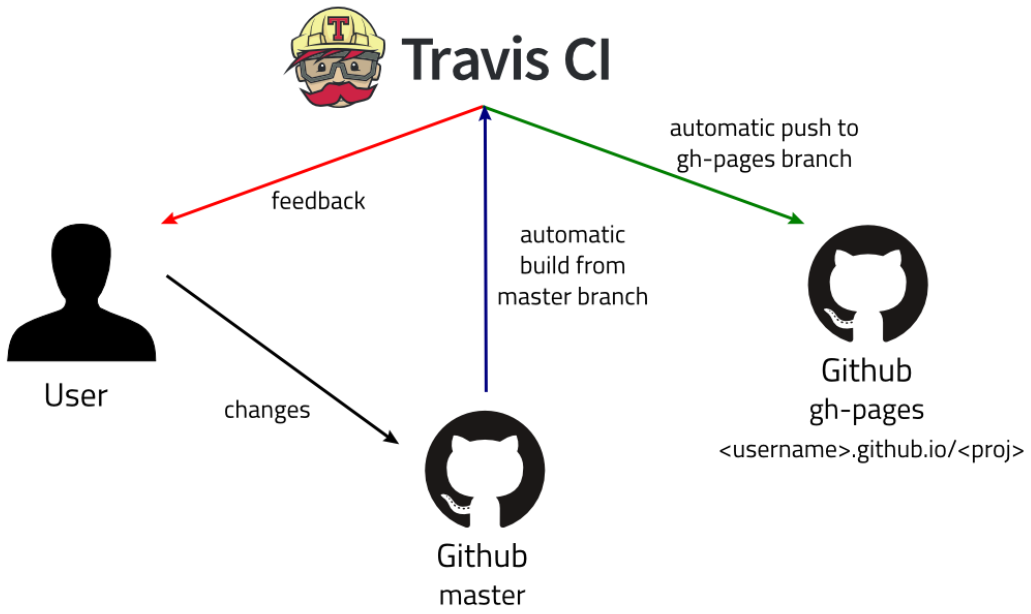
Οι χρήστες του TRAVIS CI μπορούν να ορίσουν ποια εργαλεία χρειάζονται και με ποια σειρά πρέπει να εκτελεστούν για να ολοκληρωθεί μια εργασία build. Αυτές οι λεπτομέρειες διαμόρφωσης αποθηκεύονται σε ένα αρχείο .travis.yml, το οποίο εμφανίζεται στον ριζικό κατάλογο ενός αποθετηρίου Github. Το αρχείο .travis.yml μπορεί επίσης να καθορίζει προγράμματα εκτέλεσης γλωσσών προγραμματισμού και άλλα περιβάλλοντα ρυθμίσεις διαμόρφωσης που απαιτούνται για την εκτέλεση εργασιών Continuous Building.

Λειτουργία

Όταν το Travis CI έχει ενεργοποιηθεί για ένα δεδομένο αποθετήριο, το GitHub θα το ειδοποιεί κάθε φορά που προωθούνται νέες δεσμεύσεις σε αυτό το αποθετήριο ή υποβάλλεται αίτημα έλξης . Μπορεί επίσης να ρυθμιστεί ώστε να εκτελείται μόνο για συγκεκριμένους κλάδους ή κλάδους των οποίων τα ονόματα ταιριάζουν με ένα συγκεκριμένο μοτίβο. Στη συνέχεια, το Travis CI θα ελέγξει τον σχετικό κλάδο και θα εκτελέσει τις εντολές που καθορίζονται στο .travis.yml , το οποίο συνήθως δημιουργεί το λογισμικό και εκτελεί τυχόν αυτοματοποιημένα tests. Όταν ολοκληρωθεί αυτή η διαδικασία, το Travis ειδοποιεί τον προγραμματιστή με τον τρόπο που έχει ρυθμιστεί να το κάνει, για παράδειγμα, στέλνοντας ένα email που περιέχει τα αποτελέσματα των tests(δείχνοντας επιτυχία ή αποτυχία) ή δημοσιεύοντας ένα μήνυμα σε IRC κανάλι. Στην περίπτωση αιτημάτων έλξης, το αίτημα έλξης θα σχολιαστεί με το αποτέλεσμα και έναν σύνδεσμο προς το αρχείο καταγραφής build, χρησιμοποιώντας μια ενοποίηση GitHub.

Το Travis CI μπορεί να ρυθμιστεί για να εκτελεί τα testsσε μια σειρά διαφορετικών μηχανημάτων, με εγκατεστημένο διαφορετικό λογισμικό (όπως παλαιότερες εκδόσεις εφαρμογής μιας γλώσσας προγραμματισμού, για έλεγχο συμβατότητας), και υποστηρίζει την κατασκευή λογισμικού σε πολλές γλώσσες, όπως C, C++ , C# , Clojure , D , Dart, Erlang, F#, Go , Apache,Groovy, Haskell, Java, JavaScript , Julia , Nim , Perl , PHP, Python , R , Ruby , Rust, Scala, Swift και Visual Basic . Αρκετά έργα ανοιχτού κώδικα υψηλού προφίλ το χρησιμοποιούν για την εκτέλεση εκδόσεων και

δοκιμών σε κάθε δέσμευση, όπως το Plone , το Ruby on Rails και το Ruby. Από τον Φεβρουάριο του 2013, περισσότερα από 8056 έργα JavaScript το χρησιμοποιούν [40].



Εικόνα 2-9 Εργαλείο TravisCI

Χαρακτηριστικά

Το Travis CI μπορεί να χαρακτηριστεί από τα ακόλουθα χαρακτηριστικά. Πρώτον, είναι δωρεάν για έργα ανοικτού κώδικα. Στη συνέχεια, ένα από τα κύρια χαρακτηριστικά του είναι ότι περιλαμβάνει τις πιο δημοφιλείς γλώσσες προγραμματισμού και τεχνολογίες διακομιστών. Εχει σαφείς προδιαγραφές διαμόρφωσης και διαθέτει ευρεία κοινότητα χρηστών. Παρέχει γρήγορη υποστήριξη και διαδικασία ενημέρωσης του λογισμικού στη στοίβα. Επιπλέον, ένα άλλο καλό χαρακτηριστικό είναι η απρόσκοπτη ενσωμάτωση με το GitHub. Κάθε pull request και commit παρουσιάζει την τρέχουσα κατάσταση build στο GitHub. Επιπλέον, εξοικονομεί χρόνο με την εκτέλεση μεγάλων συνόλων δοκιμών σε ένα απομακρυσμένο σύστημα που επιτρέπει στους προγραμματιστές να συνεχίσουν την εργασία τους χωρίς να σταματούν για αναμονή τους δικούς τους σταθμούς εργασίας για την εκτέλεση των tests.

Η διεπαφή χρήστη είναι πολύ ευέλικτη, οι περισσότερες χρήστες λένε ότι είναι εύκολο να χρησιμοποιήσουν το Travis CI και να κάνουν τις εργασίες, τη διαμόρφωση, εύκολα παρακολούθηση των builds. Το Travis CI επιτρέπει την ταυτόχρονη ανάπτυξη εύκολα εκτελέσιμων δοκιμών. Επιπλέον, είναι δωρεάν η ύπαρξη ενός ιδιωτικού αποθετηρίου για τους φοιτητές [41].

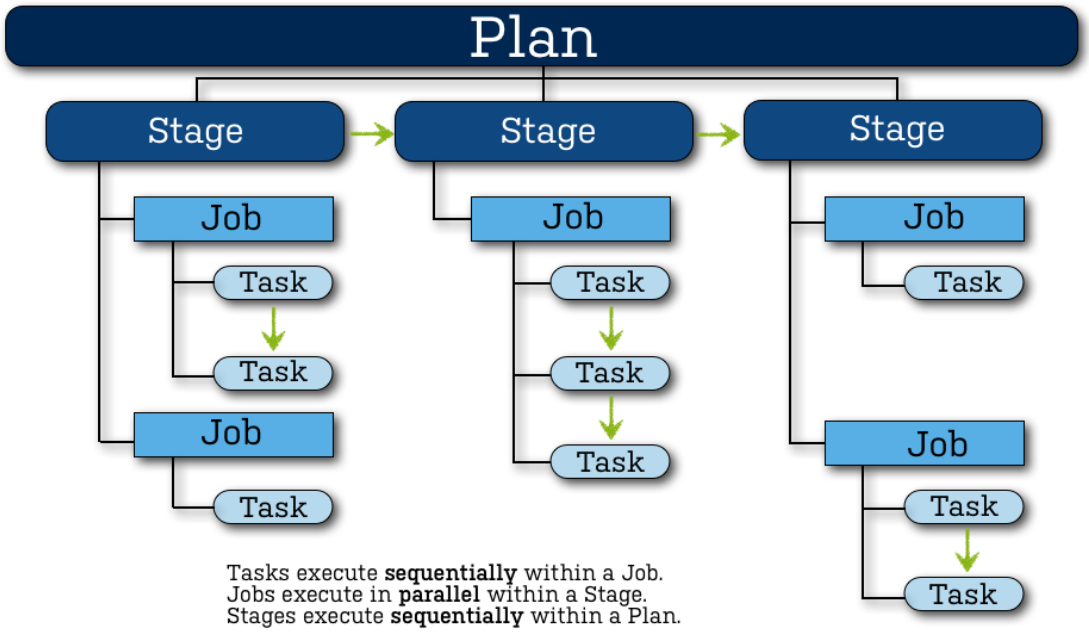
Εκτός από θετικά χαρακτηριστικά παρουσιάζει και κάποια αρνητικά. Αρκετοί αξιολογητές σημείωσαν ότι σε ορισμένες περιπτώσεις η διαδικασία ρύθμισης απαιτούσε σημαντικό χρόνο και δεν ήταν εύκολο να κατανοηθεί όλη η τεκμηρίωση.

Μπορεί να μην είναι σταθερό μερικές φορές. Κατά τη διάρκεια της μεταγλώττισης ή της εκτέλεσης δοκιμών, μπορεί να παγώσει για άγνωστους λόγους που απαιτούν χειροκίνητες διακοπές. Τις περισσότερες φορές αυτό το πρόβλημα δεν συμβαίνει, αλλά εμφανίζεται περιστασιακά. Ορισμένοι χρήστες λένε ότι θα ήθελαν να προσθέσουν SSH στη μηχανή κατασκευής για να αποσφαλματώσουν μια αποτυχία build, και στη συνέχεια να έχουν τη δυνατότητα να ενεργοποιήσουν χειροκίνητα ένα νέο build και να οργανώσουν εκ νέου ή να επαναπροσδιορίσουν τις προτεραιότητες των κατασκευών. Δεν υπάρχει καλή υποστήριξη για το Docker. Η ρύθμιση των αρχείων .travis.yml μπορεί να είναι δύσκολη και να απαιτεί πολλές υπερβολικές δεσμεύσεις για να λειτουργήσει σωστά ένα build.

Η εταιρεία έχει την έδρα της στο Βερολίνο της Γερμανίας και ιδρύθηκε το 2011. Το 2012 το έργο γνώρισε σημαντική ανάπτυξη και ξεκίνησε μια εκστρατεία χρηματοδότησης από κοινού για τη χρηματοδότηση περαιτέρω ανάπτυξης που χρηματοδοτήθηκε από δεκάδες εταιρείες τεχνολογίας. Τον Ιανουάριο του 2019 ανακοινώθηκε ότι η εταιρεία εξαγοράστηκε από την Idera, Inc. Λίγες εβδομάδες μετά την απόκτηση, μια μεγάλη μερίδα της παλιάς ομάδας του Τράβις απολύθηκε. Τον Μάρτιο του 2019 η υποδομή Travis CI υπέστη μαζική διακοπή λειτουργίας που διήρκεσε από τις 27 Μαρτίου έως τις 29 Μαρτίου. Τον Μάρτιο του 2020, ο Travis CI παρουσίασε το "The Cookbook" γραμμένο από τη Montana Mendy με μαθήματα για περιπτώσεις κοινής χρήσης. Τον Νοέμβριο του 2020, ο Travis CI ανακοίνωσε τον τερματισμό λειτουργίας travis-ci.orgέως τις 31 Δεκεμβρίου 2020, με όλους τους υπάρχοντες και νέους λογαριασμούς να μετεγκαθίστανται στο travis-ci.com. Παρά την επίσημη δέσμευση να διατηρηθούν οι "λογαριασμοί ανοιχτού κώδικα εντελώς ελεύθεροι κάτω από travis-ci.com", έργα ανοιχτού κώδικα αναφέρουν ότι οι εργασίες κατασκευής τους σταμάτησαν [40,41].

2.4.3 Εργαλείο Bamboo

Το Bamboo είναι ένα εμπορικό εργαλείο CI που αναπτύχθηκε από την Atlassian, αλλά είναι δωρεάν για έργα ανοικτού κώδικα. Συνδέει τα αυτοματοποιημένα builds και tests καθώς και τις εκδόσεις, όλα μαζί σε μια ενιαία ροή εργασίας.



Εικόνα 2-10 Εργαλείο Bamboo

Χαρακτηριστικά

Το Bamboo υποστηρίζει οποιεσδήποτε γλώσσες και άλλες τεχνολογίες διάδοσης, όπως AWS CodeDeply, Docker, Amazon S3, Maven, Ant, Git, Mercurial και SVN. Βρίσκει νέους κλάδους στα αποθετήρια Git, Mercurial και SVN και στη συνέχεια εφαρμόζει αυτόματα το σχήμα CI της κύριας γραμμής σε αυτούς. Το Bamboo παρέχει στις αναπτύξεις την πρώτης τάξης αντιμετώπιση. Η ενσωμάτωση με το Amazon S3, τα έτοιμα προς χρήση docker containers και οι εργασίες AWS CodeDeploy καθιστούν τη διαδικασία ανάπτυξης ταχύτερη και ευκολότερη. Το Bamboo παρέχει καλή ενσωμάτωση με το Jira με δυνατότητα περαιτέρω επέκτασης. Μπορεί να επεκταθεί με plugins με τον ίδιο τρόπο όπως το Jenkins. Οι ειδοποιήσεις build μπορούν να τροποποιηθούν ανάλογα με τον τύπο του συμβάντος και αποστέλλονται μέσω ηλεκτρονικού ταχυδρομείου, άμεσων μηνυμάτων, RSS ή αναδυόμενων παραθύρων.

Οι ακόλουθες εταιρείες χρησιμοποιούν το Bamboo: Εθνικό Δημόσιο Ραδιόφωνο, LeapFrog, WilliamsSonoma, Cisco, BMW, Kaiser Permanente, NASA. Σύμφωνα με χρήστες του Bamboo αναφέρεται ότι είναι ένα πολύ ευέλικτο εργαλείο με καλή ενσωμάτωση προϊόντων Atlassian, όπως το Jira και το Fisheye. Η τεκμηρίωση του Bamboo είναι καλή και λεπτομερής. Αναφέρεται ότι η Atlassian παρέχει εξαιρετική υποστήριξη. Το Bamboo λειτουργεί γρήγορα και αποτελεσματικά όταν η διαμόρφωση και το περιβάλλον έχουν ρυθμιστεί σωστά [42].

Από την άλλη πλευρά, παρουσιάζει και αρνητικά χαρακτηριστικά. Αρχικά αν το έργο δεν είναι ανοικτού κώδικα, το Bamboo θα είναι δωρεάν μόνο για τις πρώτες επτά ημέρες. Ορισμένοι από τους κριτικούς λένε ότι η διεπαφή χρήστη δεν είναι καλή και ελπίζουν ότι η Atlassian θα επικεντρωθεί περισσότερο στις εφαρμογές για κινητά

τηλέφωνα. Η πρώτη εργασιακή εμπειρία με το Bamboo μπορεί να είναι περίπλοκη. Για ορισμένους χρήστες η διαδικασία εγκατάστασης ήταν δύσκολη και δεν ήταν αρκετά σαφής για να την κατανοήσουν.

Οργάνωση μιας ροής εργασίας Bamboo

Το Bamboo χρησιμοποιεί την έννοια ενός «πλάνου» με «θέσεις εργασίας» και «εργασίες» για να διαμορφώσει και να διατάξει τις ενέργειες στη ροή εργασίας.

Project

- Δεν έχει κανένα, ένα ή περισσότερα σχέδια.
- Παρέχει αναφορές (για παράδειγμα, χρησιμοποιώντας τον πίνακα τοίχου) σε όλα τα σχέδια του έργου.
- Παρέχει συνδέσμους προς άλλες εφαρμογές.
- Επιτρέπει τη ρύθμιση δικαιωμάτων για όλα τα σχέδια που περιέχει.

Plan

- Έχει ένα μόνο στάδιο, από προεπιλογή, αλλά μπορεί να χρησιμοποιηθεί για την ομαδοποίηση εργασιών σε πολλαπλά στάδια.
- Επεξεργάζεται μια σειρά από ένα ή περισσότερα στάδια που εκτελούνται διαδοχικά χρησιμοποιώντας το ίδιο αποθετήριο.
- Καθορίζει το προεπιλεγμένο αποθετήριο.
- Καθορίζει τον τρόπο ενεργοποίησης του buildκαι τις εξαρτήσεις ενεργοποίησης μεταξύ του σχεδίου και άλλων σχεδίων στο έργο.
- Καθορίζει τις ειδοποιήσεις για τα αποτελέσματα του build.
- Καθορίζει ποιος έχει δικαίωμα προβολής και διαμόρφωσης του πλάνου και των εργασιών του.
- Παρέχει τον ορισμό των μεταβλητών του σχεδίου

Stage

- Από προεπιλογή έχει μία μόνο εργασία, αλλά μπορεί να χρησιμοποιηθεί για την ομαδοποίηση πολλαπλών εργασιών.
- Επεξεργάζεται τις εργασίες του παράλληλα, σε πολλαπλούς πράκτορες (όπου είναι διαθέσιμοι).
- Πρέπει να ολοκληρώσει επιτυχώς όλες τις εργασίες του πριν από την επεξεργασία του επόμενου σταδίου του σχεδίου.
- Μπορεί να παράγει τεχνουργήματα που μπορούν να διατεθούν για χρήση από ένα επόμενο στάδιο.

Job

- Επεξεργάζεται μια σειρά από μία ή περισσότερες εργασίες που εκτελούνται διαδοχικά στον ίδιο πράκτορα.
- Ελέγχει τη σειρά με την οποία εκτελούνται οι εργασίες.
- Συλλέγει τις απαιτήσεις των επιμέρους εργασιών της εργασίας, ώστε οι απαιτήσεις αυτές να μπορούν να αντιστοιχιστούν με τις δυνατότητες του πράκτορα.
- Καθορίζει τα τεχνουργήματα που θα παράγει το build.
- Μπορεί να χρησιμοποιήσει μόνο τεχνουργήματα που έχουν παραχθεί σε προηγούμενο στάδιο.
- Καθορίζει τυχόν ετικέτες με τις οποίες θα επισημανθεί το αποτέλεσμα του build ή τα τεχνουργήματά του.

Task

- Είναι μια μικρή διακριτή μονάδα εργασίας, όπως η ολοκλήρωση του ελέγχου του πηγαίου κώδικα, η εκτέλεση ενός στόχου του Maven, η εκτέλεση ενός σεναρίου ή η ανάλυση των αποτελεσμάτων των δοκιμών.
- Εκτελείται διαδοχικά στο πλαίσιο μιας εργασίας σε έναν κατάλογο εργασίας του Bamboo [43].

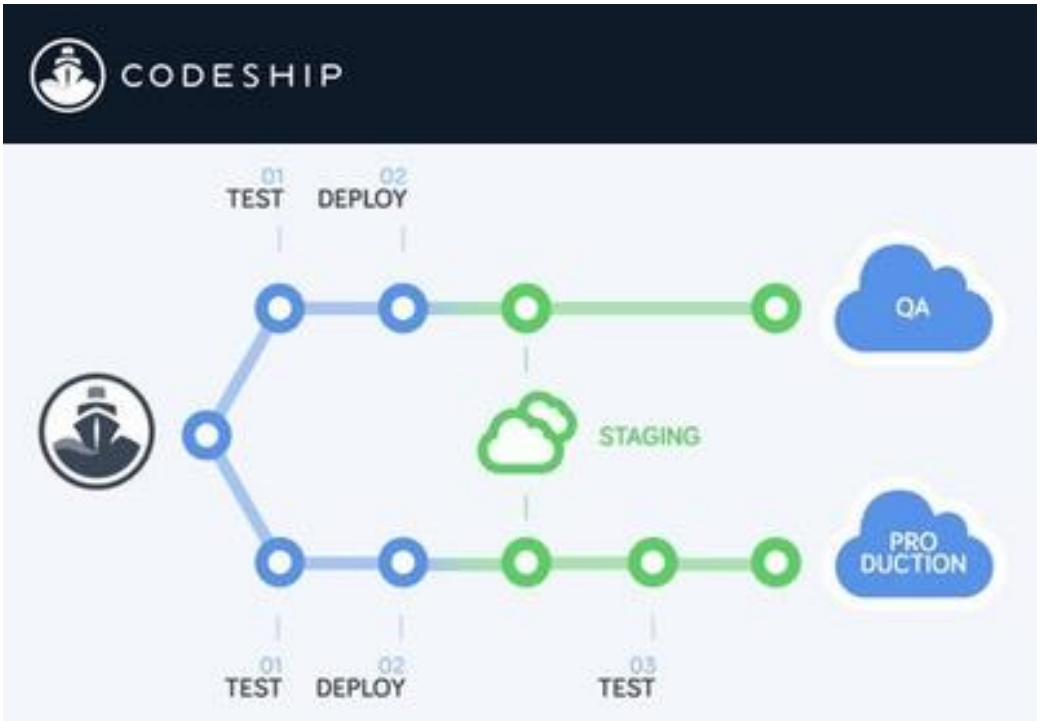
2.4.4 Εργαλείο Codeship

Το Codeship είναι ένα εργαλείο CI για διαδικτυακές εφαρμογές, συμπεριλαμβανομένων των php, rails, java, python ή go apps. Εκτελεί continuous tests και ρυθμίζει την ανάπτυξη μετά από αλλαγές σε ένα αποθετήριο. Συνεχίζει να διαχειρίζεται και να κλιμακώνει την υποδομή, επιτρέποντας έτσι στους προγραμματιστές να μπορούν να δοκιμάζουν και να κυκλοφορούν συχνότερα και να λαμβάνουν γρήγορη ανατροφοδότηση για τη βελτίωση της διαδικασίας δημιουργίας ενός αναπτυξιακού λογισμικού.

Η πρακτική CI περιλαμβάνει επαναλαμβανόμενα continuous testsκαι συγχώνευση κώδικα με ειδοποιήσεις για να ειδοποιούν τους προγραμματιστές για προβλήματα στον κώδικά τους σε πραγματικό χρόνο. Το Codeship το συνδυάζει με την πρακτική του Continuous Delivery , η οποία προβλέπει την τακτική ανάπτυξη κώδικα μόλις οι αλλαγές περάσουν τα continuous tests. Η συνεργασία με μια υπηρεσία όπως το Codeship μπορεί να ελαχιστοποιήσει τις επιπτώσεις από σφάλματα, να επιταχύνει τον εντοπισμό τους και να κάνει μία ομάδαπιο άνετη και πιο ικανή στην κυκλοφορία τακτικών σταδιακών βελτιώσεων σε κάθε πλατφόρμα.

Σε σύγκριση με το Jenkins, το Codeship έχει καλύτερο UI και λειτουργεί καλύτερα στην περίπτωση που έχετε εγκαταστήσει τεράστιο αριθμό plugins στο Jenkins, επειδή το επιβραδύνει. Επιπλέον, το Codeship διαθέτει καλύτερη υπηρεσία υποστήριξης. Οι

χρήστες λένε ότι σε περίπτωση που αντιμετώπιζαν οποιοδήποτε πρόβλημα και χρειάζονταν υποστήριξη, την είχαν πάντα απίστευτα σύντομα. Παρά το γεγονός ότι το Codeship παρέχει τόσες πολλές καλές υπηρεσίες, διαθέτει ένα δωρεάν πακέτο με μειωμένη λειτουργικότητα για έργα ανοικτού κώδικα [44].



Εικόνα 2-11 Εργαλείο Codeship

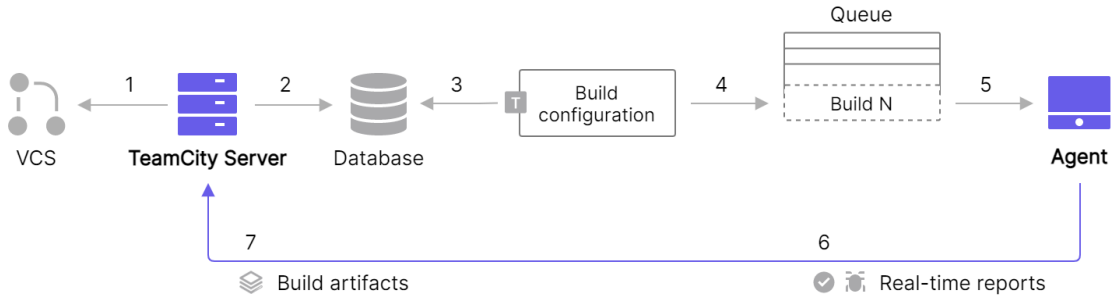
Χαρακτηριστικά

Είναι εύκολη η εγκατάσταση με το GitHub και εύκολη η διαμόρφωση των δοκιμών, επίσης είναι πολύ βολικό να αναπτυχθεί στις υπηρεσίες Amazon Web Services. Ορισμένες από τις κριτικές υπονοούν ότι η εμπειρία του χρήστη είναι πολύ καλή στο Codeship. Στους ανθρώπους άρεσε ο τρόπος με τον οποίο υλοποιείται η κοινή εγκατάσταση, η δοκιμή και η ανάπτυξη στο Codeship. Οι περισσότεροι από αυτούς τονίζουν ότι η διαδικασία εργασίας με το Codeship είναι απλή και φιλική προς το χρήστη. Οι άνθρωποι περιγράφουν τη διεπαφή χρήστη ως διαισθητική και περίτεχνη, επειδή όλα τα εργαλεία είναι ισχυρά και λειτουργούν με συνέπεια και δεν χρειάζεται πολύς χρόνος για να εξοικειωθεί ο χρήστης με το GUI. Για παράδειγμα, εάν το build αποτύχει, ο προγραμματιστής γνωρίζει ότι το πρόβλημα είναι με έναν κώδικα ή μια δοκιμή και όχι με την πλατφόρμα. Αυτή η τεχνική επιτρέπει την αντιμετώπιση προβλημάτων και τον προσδιορισμό των προβλημάτων πολύ πιο γρήγορα. Ορισμένες κριτικές έδωσαν έμφαση στη δυνατότητα υποστήριξης του Secure Shell σε περιβάλλον build. Κάποιοι από αυτούς συνιστούν σε όλους να δοκιμάσουν το Codeship, ειδικά σε μικρές ομάδες με περιορισμένο προϋπολογισμό, καθώς το Codeship προσφέρει μια λογική λειτουργικότητα στο δωρεάν πακέτο του.

Παρ' όλα αυτά, έχει αρνητικές ιδιότητες. Σε ορισμένες κριτικές, οι χρήστες πρότειναν να προστεθεί υποστήριξη αποθήκευσης αντικειμένων. Στη συνέχεια, υπάρχουν μικρά προβλήματα με τις ειδοποιήσεις. Οι περισσότερες κριτικές αναφέρουν ότι υπάρχει έλλειψη τεκμηρίωσης και θα πρέπει να επεκταθεί, επειδή τους ενοχλεί, ειδικά σε σύγκριση με το Jenkins. Επιπλέον, ορισμένοι λένε ότι η τιμή είναι υψηλή [45].

2.4.5 Εργαλείο TeamCity

Το TeamCity είναι ένας πολυλειτουργικός διακομιστής CI, ο οποίος είναι έτοιμος να λειτουργήσει αμέσως μετά την εγκατάσταση. Υποστηρίζει πολλά συστήματα ελέγχου εκδόσεων, έλεγχο ταυτότητας, ανάπτυξη και δοκιμές. Το TeamCity είναι επεκτάσιμο, για πολλές λειτουργίες δεν χρειάζεται καν να υποστηρίζει Java. Είναι εύκολο στην εγκατάσταση και είναι δωρεάν για μικρές ομάδες και έργα ανοικτού κώδικα. Το TeamCity χρησιμοποιείται από τις ακόλουθες γνωστές εταιρείες: Apple, Ebay, Jvm stack, Yammer, Stack exchange, NationBuilder, Stack Overflow [46].



Εικόνα 2-12 Εργαλείο TeamCity

Χαρακτηριστικά

Τα σημαντικότερα χαρακτηριστικά του Team City περιγράφονται παρακάτω.

- Άμεση ανατροφοδότηση για προβλήματα και αποτυχίες δοκιμών κατά τη διάρκεια της προόδου του build.
- Δεν απαιτεί από τον χρήστη να περιμένει το τέλος του build για να μπορέσει να δει σφάλματα στη μεταγλώττιση ή αποτυχίες δοκιμών.
- Δυνατότητα εκτέλεσης διαδικασιών μεταγλώττισης και δοκιμών τροποποιημένου κώδικα χωρίς δέσμευση στο σύστημα ελέγχου εκδόσεων, απευθείας από το IDE.
- Απλοποιημένη εγκατάσταση - επιτρέπει τη δημιουργία έργων μόνο από μια διεύθυνση URL ενός αποθετηρίου VCS.
- Εκτέλεση πολλαπλών έργων και δοκιμών υπό διαφορετικές διαμορφώσεις και πλατφόρμες ταυτόχρονα. Η ιεραρχική δομή των έργων, που καθιστά εύκολη τη ρύθμιση των δικαιωμάτων και τη διαμόρφωση του διακομιστή.
- Μεγάλες στατιστικές αναφορές για τα αποτελέσματα των έργων με πληροφορίες σχετικά με τη διάρκεια του έργου, το ποσοστό επιτυχίας, την ποιότητα του κώδικα και προσαρμοσμένες μετρήσεις.
- Μεγάλη οπτική αναπαράσταση του έργου.

- Παρακολουθεί κάθε αλλαγή που πραγματοποιείται από οποιονδήποτε χρήστη στο σύστημα, φιλτράρει τα έργα και επιλέγει το στυλ της οπτικής αναπαράστασης της κατάστασης αλλαγών.
- Η διαχείριση κοινών πόρων επιτρέπει την εύκολη πρόσβαση σε κοινές βάσεις δεδομένων, συσκευές δοκιμών κλπ.
- Ρυθμιζόμενες συνθήκες αποτυχιών σε μια πρόοδο του build με βάση μια ποικιλία μετρήσεων, συμπεριλαμβανομένου του αριθμού των αποτυχημένων δοκιμών, του αριθμού των ακάλυπτων κλάσεων και ενοτήτων, επιπλέον μετρήσεις που αποκλείουν την υποβάθμιση της ποιότητας του κώδικα.
- Μοναδικά χαρακτηριστικά για την υποστήριξη διακομιστή σε κατάλληλες συνθήκες: ενσωματωμένες ιστορίες καθαρισμού build, αναφορές σχετικά με τον καταλαμβανόμενο χώρο στο δίσκο και αναφορές σχετικά με την υγεία του διακομιστή.
- Υποστήριξη μικτού ελέγχου ταυτότητας που επιτρέπει τη χρήση διαφορετικών μεθόδων ελέγχου ταυτότητας (LDAP, Windows Domain, ενσωματωμένη) ταυτόχρονα.
- Άριστη ενσωμάτωση με συστήματα ελέγχου εκδόσεων: υποστήριξη πολλαπλών συστημάτων σε ένα μόνο έργο, δυνατότητα διακλάδωσης για το Mercurial και το Git, προηγμένοι κανόνες για την έναρξη του build με βάση τις αλλαγές στα συστήματα ελέγχου εκδόσεων.
- Οι ρόλοι και οι ομάδες χρηστών επιτρέπουν τη ρύθμιση εύκολης πρόσβασης στο διακομιστή για όλους τους χρήστες.
- Υποστήριξη έργων Ruby και XCode, διαισθητική διεπαφή ιστού, υποστήριξη μηνυμάτων υπηρεσίας, περισσότερα από 100 διαθέσιμα δωρεάν πρόσθετα, εύκολα επεκτάσιμη λειτουργικότητα του TeamCity και νέες ενσωματώσεις με Java API.
- Σύμφωνα με τις κριτικές, η εγκατάσταση του TeamCity είναι πολύ εύκολη και γρήγορη. Επιπλέον, η διεπαφή χρήστη του TeamCity είναι απλή και εύκολη στην εξοικείωση και παρέχει μεγάλη ποικιλία επιλογών. Το TeamCity αναφέρει την κατάσταση του έργου και τις σχετικές πληροφορίες για διάφορους χρήστες, ακόμη και για τους ενδιαφερόμενους του έργου. Το TeamCity περιλαμβάνει τεκμηρίωση με πολλά παραδείγματα και σεμινάρια για διάφορες περιπτώσεις. Επιπλέον, οι χρήστες λένε ότι είναι εύκολο να βρουν απαντήσεις στο Διαδίκτυο, συμπεριλαμβανομένων θεμάτων όπως η εγκατάσταση της πλατφόρμας, λόγω του γεγονότος ότι το TeamCity έχει παρουσία στην αγορά για μεγάλο χρονικό διάστημα και έχει αποκτήσει δημοτικότητα μεταξύ των χρηστών.
- Υπάρχει μια μακρά αλλά πολύ απλή διαδικασία ρύθμισης των έργων, η οποία καθιστά δυνατή την τροποποίηση επιλογών, όπως η δημιουργία ενός χρονοδιαγράμματος για την ενεργοποίηση ενός έργου.
- Ένα από τα πιο δημοφιλή χαρακτηριστικά του TeamCity είναι η αυτοματοποίηση διαφόρων πτυχών του κύκλου ζωής της ανάπτυξης

λογισμικού. Με την αυτοματοποίηση της διαδικασίας ανάπτυξης οι χρήστες είναι σε θέση να αποκτήσουν μεγαλύτερη παραγωγικότητα. Η διαδικασία επιτρέπει τη μείωση του χρόνου ανάπτυξης και παρακολουθεί εύκολα τις διαμορφώσεις της διαδικασίας ανάπτυξης. Η διαμόρφωση του build είναι πολύ ευέλικτη καθώς ο πηγαίος του κώδικας περιλαμβάνεται με διαφορετικούς τρόπους (πχ μέσω shell scripting κλπ). Επιπλέον, οι κριτικές παρουσιάζουν το TeamCity ως ένα καλό προϊόν για ομάδες ανάπτυξης οποιουδήποτε μεγέθους.

- Επιπλέον, ταιριάζει στις εταιρείες με διαφορετικά περιβάλλοντα και υπηρεσίες, καθώς μπορεί να επεκταθεί για να ταιριάζει στα έργα.
- Ωστόσο, έχει ορισμένες ελλείψεις. Ορισμένοι από τους αξιολογητές λένε ότι δεν ήταν ικανοποιημένοι με την τεκμηρίωση και ότι η διαδικασία εγκατάστασης και αναβάθμισης ήταν επίσης δύσκολη. Σε έναν από τους αξιολογητές δεν άρεσε η επεξήγηση της ειδικής ορολογίας του λογισμικού, που ως επί το πλείστον πρέπει να αναζητήσει τις έννοιες των όρων στο Διαδίκτυο. Λέγεται ότι οι εργαλειοθήκες (tooltips) μπορούν να προσφέρουν πιο ουσιαστική και άμεση ανατροφοδότηση από το άνοιγμα παραθύρων με μεγάλο όγκο πληροφοριών [46,47].

2.4.6 Εργαλείο CircleCI

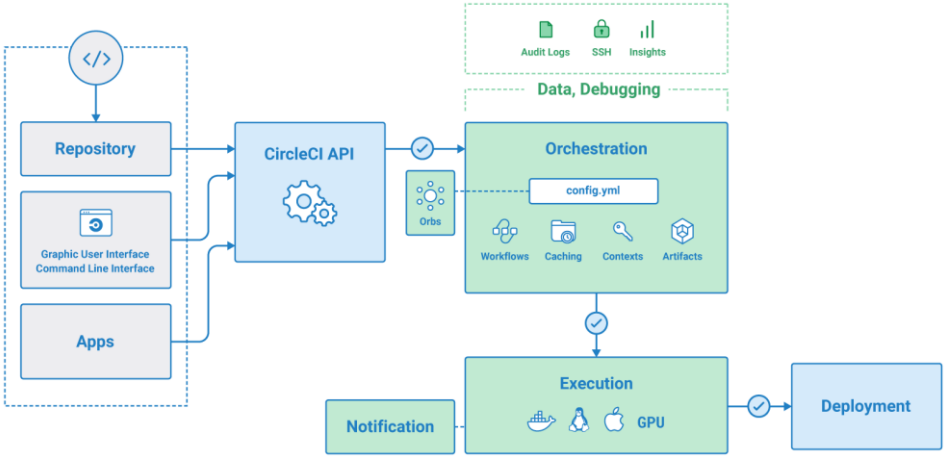
Το CircleCI είναι μια ενσωμάτωση που είναι συνεχής και λειτουργεί σαν δίκτυο παράδοσης. Μπορεί να κατασκευαστεί τοπικά και να χρησιμοποιηθεί στο cloud, υποστηρίζει πολλές γλώσσες κώδικα. Αυτό το εργαλείο διευκολύνει τα tests, τα builds και τα deployments. Η βασική διεπαφή χρήστη έχει πολλά επιλογές διαμόρφωσης. Οι προγραμματιστές μπορούν να μειώσουν το αριθμό των σφαλμάτων με το CircleCI και να βελτιώσουν γρήγορα την εφαρμογή ποιότητα. Το CircleCI παρέχει ένα δωρεάν σχέδιο για open -source έργα, αν και πρόκειται για ένα εμπορικό εργαλείο [48].

Χαρακτηριστικά

Το CircleCI είναι ένα από τα πιο δημοφιλή και αξιόπιστα προϊόντα που διαθέτει:

- Γρήγορη εγκατάσταση. Η διαδικασία εγκατάστασης είναι περίπου 20 δευτερόλεπτα. Το CircleCI ανιχνεύει δοκιμαστικές ρυθμίσεις για ένα ευρύ φάσμα εφαρμογών ιστού με ένα μόνο κλικ, και στη συνέχεια τις ρυθμίσεις σε CircleCI διακομιστές κάθονται αυτόματα.
- Βαθιά προσαρμογή. Οι ρυθμίσεις του CircleCI είναι πολύ ευέλικτες, επιτρέποντας την εύκολη ρύθμιση να ρυθμίσετε σχεδόν οτιδήποτε απαιτείται.
- Η αποσφαλμάτωση γίνεται με ευκολία. Όταν οι δοκιμές είναι κατεστραμμένες, το CircleCI βοηθάει στη διόρθωση τους. Εντοπίζει αυτόματα τα σφάλματα και παρέχει μεγάλη υποστήριξη.
- Έξυπνες ειδοποιήσεις. Μπορεί να ειδοποιεί έξυπνα μέσω ηλεκτρονικού ταχυδρομείου, Hipchat, Campfire και άλλες πηγές. Οι ειδοποιήσεις περιέχουν μόνο σχετικές πληροφορίες.

- Γρήγορη υποστήριξη. Η CircleCI ανταποκρίνεται στα αιτήματα υποστήριξης πολύ γρήγορα, οι χρήστες δεν χρειάζεται να περιμένουν πάνω από 12 ώρες για μια απάντηση.
- Αυτόματη παραλληλοποίηση. Το CircleCI μπορεί να παραλληλοποιήσει αυτόματα τις δοκιμές με έως και 4 παράλληλες ροές μέσω πολλαπλών μηχανών.
- Συνεχής ανάπτυξη, αντικείμενα δημιουργίας, καθαρά περιβάλλοντα δημιουργίας, ενσωμάτωση στο GitHub, γρήγορες δοκιμές, δωρεάν υποστήριξη ανοικτού κώδικα. Σύμφωνα με τις κριτικές, το CircleCI είναι ένα διαισθητικό και εξαιρετικά χρήσιμο εργαλείο, μάλιστα ένας από τους κριτές ισχυρίζεται ότι είναι ταχύτερο από το Travis και απλούστερο από το Jenkins. Το χαρακτηρίζουν ως γρήγορο και εύκολο στην εγκατάσταση και ωφέλιμο για τις διαδικασίες ανάπτυξης. Επιπλέον, ένας αναθεωρητής διαπίστωσε ότι η τεκμηρίωσή τους είναι καλή και όταν είχε απορίες, η υποστήριξη της CircleCI τον βοήθησε πολύ γρήγορα. Υπάρχει μια δωρεάν δοκιμαστική περίοδος 14 ημερών για να δοκιμάσετε το Circle CI, αφού έχει χρόνο να δει αν ανταποκρίνεται σε όλες τις απαιτήσεις της ομάδας ή όχι. Η διεπαφή χρήστη είναι πολύ καθαρή και εύκολα κατανοητή. Ένας από τους αξιολογητές τόνισε την αυξημένη ταχύτητα των παράλληλων tests. Υπάρχουν έτοιμες διαμορφώσεις με καλή ενσωμάτωση στο GitHub. Οι ειδοποιήσεις είναι έγκαιρες και χρήσιμες, μπορούν να αποσταλούν μέσω ηλεκτρονικού ταχυδρομείου, συνομιλίας και ειδοποιήσεων webhook [49].



Εικόνα 2-13 Εργαλείο CircleCI

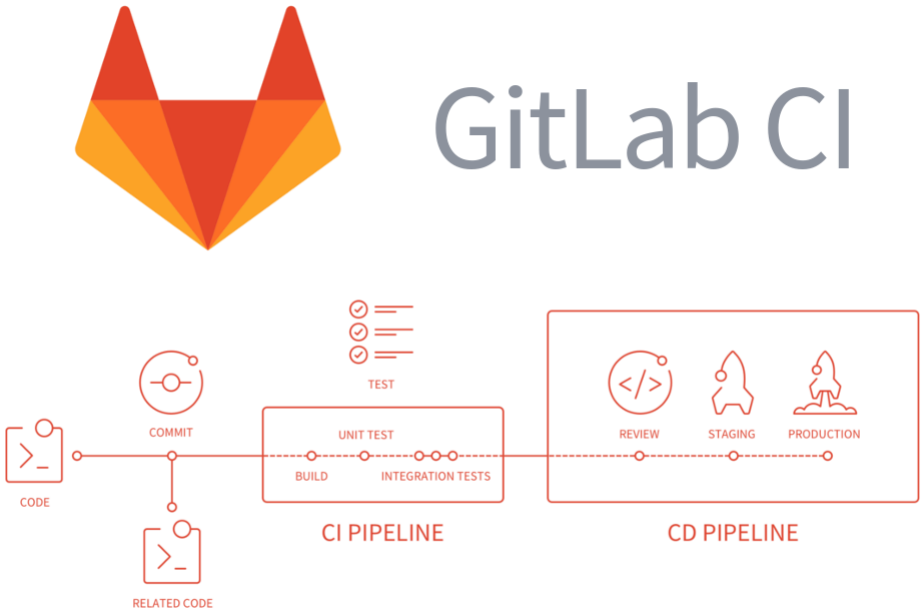
Από την άλλη υπάρχουν ορισμένα μειονεκτήματα. Για παράδειγμα, υπήρχαν ορισμένες περιπτώσεις όπου τα testsχάλασαν εξαιτίας των αναβαθμίσεων της CircleCI. Αν και πρόκειται για ένα μικρό παράπονο, αλλά θα ήταν καλύτερα αν ήταν δυνατόν να ορίσετε μεταβλητή περιβάλλοντος για όλες τις έργα. Μερικές φορές οι δοκιμές δεν μπορούν να περάσουν επειδή κάποια επιπλέον διαμόρφωση είναι απαιτείται. Επίσης, το CircleCI στέλνει ειδοποιήσεις σε πολλά μηνύματα ηλεκτρονικού ταχυδρομείου κάθε φορά που

ένα push πραγματοποιείται. Ένας από τους αξιολογητές δήλωσε ότι η εμπειρία του χρήστη δεν ήταν τόσο άνετη και θα μπορούσε να βελτιωθεί. Επιπλέον, δεν ενημερώνεται πάντα με τις τελευταίες java εκδόσεις του κιτ ανάπτυξης της java.

Το CircleCI είναι παρόμοιο στην ιδέα με το Jenkins, εκτός από το ότι είναι μια ιδιόκτητη λύση με μοντέλο άδειας χρήσης freemium. Το CircleCI χρησιμοποιείται από μεγάλες εταιρείες όπως η Samsung, η Spotify, η Lyft, η Ford Motor Company και η Coinbase . Παρά το γεγονός ότι είναι μια επί πληρωμή λύση, το CircleCI υποστηρίζει σε μεγάλο βαθμό την κοινότητα ανοιχτού κώδικα, προσφέροντας επιπλέον 400.000 μονάδες ανά μήνα για εκδόσεις ανοιχτού κώδικα. Τα React, Vue.js, Helm, PyTorch και OpenMCT είναι μερικά από τα μεγάλα έργα ανοιχτού κώδικα που χρησιμοποιούν αυτήν τη στιγμή το CircleCI [48,49].

2.4.7 Εργαλείο GitLab

Το GitLab είναι μια διαδικτυακή πλατφόρμα διαχείρισης αποθετηρίων git και δημιουργήθηκε από τους Dmitriy Zaporozhets και Valery Sizov. Ήταν αρχικά γραμμένο σε Ruby και μερικά μέρη σε Go. Το GitLab προσφέρει δωρεάν ιδιωτικά αποθετήρια, εγγραφές container, webhooks και μία ολοκληρωμένη υπηρεσία CI και CD που ονομάζεται GitLab CI [50].



Εικόνα 2-14 Εργαλείο GitLab CI

Το εργαλείο GitLab CI προσφέρει ορισμένα πλεονεκτήματα για αγωγούς CI/CD, από την εύκολη εγκατάσταση έως την προηγμένη αυτοματοποίηση. Το διαδικτυακό εργαλείο ενθαρρύνει τις σωστές πρακτικές κώδικα εντός της ομάδας και την ασφαλή ανάπτυξη στην παραγωγή. Δεν χρειάζεται να εγκαταστήσετε τίποτα για συνεχή ολοκλήρωση και συνεχή παράδοση, καθώς διαθέτει ήδη και τα δύο ως ενσωματωμένο χαρακτηριστικό.

- Ένα ισχυρό πλεονέκτημα του Gitlab CI είναι ότι δεν χρειάζεται να εγκατασταθεί κανένα πρόσθετο ή πρόγραμμα για να επιτευχθεί Continuous Integration καθώς ως εργαλείο έχει ήδη ενσωματωμένο λογισμικό CI αλλά και CD (Continuous Delivery).
- Για όλες τις εφαρμογές που αναπτύσσονται μέσω του Gitlab, εμφανίζονται μετρήσεις επιδόσεων για κάθε μία από αυτές ενώ χωρίς να απομακρυνθούν από το Gitlab, οι προγραμματιστές μπορούν να αναλύσουν τον αντίκτυπο κάθε μικρής αλλαγής στην παραγωγή.
- Οι προγραμματιστές έχουν τη δυνατότητα να δημιουργούν ειδοποιήσεις επιπέδου υπηρεσίας για κάθε κοινοποιημένο συμβάν. Στον ίδιο κώδικα αλλά και στην ίδια ροή εργασίας.
- Είναι το πιο ασφαλές για ένα container στην περίπτωση των εικόνων docker (docker images), εξαιτίας του οποίου μπορείτε εύκολα να κατεβάσετε και να ανεβάσετε εικόνες καθώς υπάρχει πλήρης ενσωμάτωση με τη διαχείριση αποθετηρίου Git.
- Για κάθε νέο κλάδο(branch), μπορεί να δημιουργηθεί ένα νέο περιβάλλον μέσω του Gitlab CI, το οποίο θα βοηθήσει στην ενίσχυση της διαδικασίας ανάπτυξης, ενώ μπορεί πάντα να γίνει προεπισκόπηση των αλλαγών σε ζωντανό περιβάλλον.
- Καθώς τα γραφήματα περιέχουν συχνά πολύπλοκες δομές, το Gitlab διαθέτει ένα χαρακτηριστικό γράφημα αγωγού που επιτρέπει στους χρήστες να βλέπουν μέσω του γραφήματος τι συμβαίνει στη διαδικασία σε πραγματικό χρόνο και ποια είναι η τρέχουσα κατάσταση.
- Παρέχει μια λειτουργία για τον έλεγχο της ποιότητας του κώδικα. Ακόμα και αν θέλουμε να κάνουμε μια αλλαγή σε έναν κώδικα, τότε θα μας προτείνει επίσης τον αντίκτυπο αυτής της αλλαγής στον κώδικα μέσω ενός ελέγχου ποιότητας. Η συγκεκριμένη λειτουργία είναι ενσωματωμένη στο Gitlab CI και δεν χρειάζεται να εγκατασταθεί κάποιο επιπλέον πρόσθετο.
- Τέλος, μπορεί να εκτελέσει οποιονδήποτε αριθμό εργασιών μέσω του Gitlab runner. Μπορούμε εύκολα να δημιουργήσουμε, να κάνουμε τα απαραίτητα tests και να αναπτύξουμε οποιοδήποτε έργο με βάση το Linux και μέσω σεναρίων Powershell και γραμμής εντολών [51,52].
- Ωστόσο, ως εργαλείο εμφανίζει ορισμένες ελλείψεις. Το σημαντικότερο απ'όλα είναι ότι για κάθε νέα εργασία είναι απαραίτητο να ορίζουμε και να ανεβάζουμε/κατεβάζουμε τεχνουργήματα (αρχεία, δέσμες που βρίσκονται μόνο μέσα στον αγωγό διασφαλίζοντας την ομαλή και συνεκτική λειτουργία του) [52].

- Τέλος, ένα ακόμη μείον του συγκεκριμένου εργαλείου είναι ότι ενώ η πραγματική συγχώνευση θα πρέπει να συμβαίνει πριν δοκιμαστεί η κατάσταση συγχώνευσης ενός κλάδου, στο Gitlab CI συμβαίνει αναγκαστικά μετά την δοκιμή [51,52].

2.4.8 Buddy

Το Buddy, έχει αναγνωριστεί από κορυφαίους προγραμματιστές παγκοσμίως, καθώς συνδυάζει άψογη εμπειρία χρήστη με κορυφαίες επιδόσεις σε μια λύση για ομάδες που θέλουν να επιταχύνουν τον κύκλο ζωής της ανάπτυξης του λογισμικού τους. Είναι ένα εργαλείο CI/CD, με πολύ γρήγορη διαδικασία ανάπτυξης, περίπου 12 δευτερόλεπτα κατά μέσο όρο, υποστηρίζοντας τη χρήση Docker καθώς και ταυτόχρονη χρήση ενεργειών σε έναν αγωγό CI/CD.

Εικόνα 2-15 Εργαλείο Buddy

Είναι ενσωματωμένο με πολλές υπηρεσίες φιλοξενίας αποθετηρίων source control, όπως το GitLab, το Bitbucket και το GitHub, όπου ο πηγαίος κώδικας του έργου μπορεί να αποθηκευτεί εύκολα, ενώ διαθέτει πάνω από 100 έτοιμες προς χρήση ενέργειες που μπορούν να καταστήσουν πολύ εύκολα τα στάδια build, test και deploy οποιουδήποτε έργου. Σε περίπτωση που δεν θέλετε να παρέχετε στο Buddy πρόσβαση ανάγνωσης και εγγραφής στον λογαριασμό σας στο GitHub, μπορείτε απλά να φιλοξενήσετε το έργο σας μέσα στο ίδιο το Buddy.

Επιπλέον, το Buddy δεν χρειάζεται να εγκατασταθεί σε υπολογιστή ή σε κάποια συσκευή. Όλη η διαδικασία γίνεται δημιουργώντας έναν λογαριασμό μέσα από την ιστοσελίδα της εταιρείας και έτσι μπορεί άμεσα να ξεκινήσει κάποιος την πρώτη του εργασία, μετά την εισαγωγή και διασύνδεση του αποθετηρίου. Είναι ένα επιχειρηματικό εργαλείο που διαθέτει ένα απλό και εύχρηστο περιβάλλον εργασίας με βελτιωμένο και σύγχρονο σχεδιασμό υλικού ενώ παρέχεται υποστήριξη 24 ώρες το 24ωρο από ζωντανό προσωπικό είτε είσαι συνδρομητής σε κάποιο πλάνο είτε όχι.

Όσον αφορά τις αναπτύξεις στο Buddy, βασίζονται σε σύνολα αλλαγών, πράγμα που σημαίνει ότι μόνο τα τροποποιημένα αρχεία αναπτύσσονται, και ο ίδιος ο μηχανισμός

ανάπτυξης είναι βελτιστοποιημένος στα όρια. Με άλλα λόγια, δεν χρειάζεται να ανεβαίνει ολόκληρο το αποθετήριο κάθε φορά, κάτι που καθιστά το ανέβασμα ταχύτατο. Ωστόσο, η αυτοματοποίηση δεν περιορίζεται μόνο στην ανάπτυξη. Με το Buddy είναι εφικτός ο έλεγχος και η μεταγλώτισση κώδικα JS/CSS καθώς και η δοκιμή των αλλαγών σε κάθε push στο Git. Δυνατότητα τροποποίησης του κώδικα που αναπτύσσεται.

Το Buddy υποστηρίζει όλους τους τύπους δοκιμών - από δοκιμές PHPUnit μέχρι δοκιμές ολοκλήρωσης, browser και οπτικές δοκιμές (visual tests). Μπορείτε επίσης να συνδέσετε τη MySQL και να εγκαταστήσετε οποιαδήποτε πακέτα απαιτούνται από τις δοκιμές σας απευθείας στο δοχείο [53].

Ένα από τα μεγαλύτερα πλεονεκτήματα του εργαλείου Buddy είναι η άριστη συμβατότητά του με πλατφόρμες CMS όπως το Wordpress και το Shopify. Το Wordpress είναι μία πλατφόρμα μέσα από την οποία έχει κατασκευαστεί το 48,1% των ιστοσελίδων σε όλο το διαδίκτυο (από απλές ιστοσελίδες, blogs μέχρι ηλεκτρονικά καταταστήματα) ενώ το Shopify διαθέτει ένα ποσοστό 9,9% της αγοράς το οποίο όμως αφορά αποκλειστικά και μόνο ηλεκτρονικά καταστήματα. Δηλαδή με λίγα λόγια, το Buddy καλύπτει πλήρως και μπορεί να εξυπηρετήσει με αποκλειστικές λειτουργίες και ενέργειες σχεδόν το 60% της αγοράς των ιστοσελίδων.

Με το Buddy μπορεί να αυτοματοποιηθεί η ανάπτυξη σε όλους τους τύπους διακομιστών bare-metal και cloud σε μία ιστοσελίδα Wordpress, κάτι που αλλάζει τα δεδομένα όσον αφορά την παραγωγικότητα, ενώ όσον αφορά την ανάπτυξη θεμάτων, πρόσθετων ή API μπορεί να ρυθμιστεί ώστε το κάθε ανέβασμα στην παραγωγή να γίνεται αυτοματοποιημένα. Στη συνέχεια, έρχονται οι εντολές WP-CLI που επιτρέπουν την εκτέλεση αντιγράφων ασφαλείας, την ενημέρωση πυρήνα του WP και μεταφορά των βάσεων δεδομένων μετά το ανέβασμα. Επίσης, με το Buddy μπορούν να προστεθούν visual tests, ώστε να είναι εφικτή η εξέταση και ο έλεγχος του τελικού αποτελέσματος των αλλαγών καθώς και ειδοποιήσεις που επιτρέπουν την ενημέρωση των πελατών για νέες εκδόσεις.

Όσον αφορά τη διαδικασία CI του Shopify με το εργαλείο Buddy, η διαδικασία build εκτελείται εξ ολοκλήρου σε απομονωμένα κοντέινερ, γεγονός που εξαλείφει τους κινδύνους κατασκευής που αφορούν συγκεκριμένους μηχανικούς. Τα σχέδια δεν βασίζονται σε λεπτά ανάπτυξης κάτι που μειώνει το κόστος για τις εταιρείες που πραγματοποιούν μεγάλους όγκους ενημερώσεων. Σε συνδυασμό με το 99,9% uptime (εξαιρουμένης της προγραμματισμένης συντήρησης), το Buddy αποτελεί τον απόλυτο σύντροφο ανάπτυξης σε περιβάλλον Shopify [54].

Τέλος, το Buddy διαθέτει ακόμη και στο δωρεάν πλάνο τη δυνατότητα ανάπτυξης σε FTP και SMTP κάτι που άλλα εργαλεία είτε δεν το διαθέτουν ή το παρέχουν με επιπλέον κόστος. Το Buddy λοιπόν επιτρέπει την εφαρμογή FTP με 100+ έτοιμες προς χρήση ενέργειες για να αυτοματοποιήσετε την ανάπτυξή σας και να δημιουργήσετε καλύτερες εφαρμογές γρηγορότερα.

Η ενέργεια FTP του εργαλείου Buddy έχει προσαρμοστεί έτσι ώστε ναμεταφορτώνει τα αρχεία από ένα αποθετήριο Git / GitHub / Bitbucket ή σύστημα αρχείων αγωγού του Buddy σε έναν διακομιστή FTP. Το σύστημα αρχείων του αγωγού περιέχει αρχεία που έχουν ληφθεί από το αποθετήριο, αρχεία του αποθετηρίου που αντικαθίστανται από τις ενέργειες του αγωγού, αντικείμενα που δημιουργούνται από τις ενέργειες και στατικά αρχεία που μεταφορτώνονται στο σύστημα αρχείων και δεν θα έπρεπε να βρίσκονται στο αποθετήριο.

Όταν λοιπόν ο αγωγός φτάσει στο σημείο να εκτελέσει την ενέργεια FTP, εκείνη τη στιγμή το σύστημα γνωρίζει τι υπάρχει στον διακομιστή και ανεβάζει μόνο ό,τι έχει αλλάξει μεταξύ αναθεωρήσεων/προωθήσεων. Με λίγα λόγια, μπορεί να ανεβάζει νέα αρχεία, να ενημερώνει τα αλλαγμένα και να αφαιρεί ό,τι έχει διαγραφεί από το αποθετήριο [53].

Βέβαια, όπως κάθε εργαλείο έτσι και το Buddy δεν μπορεί να έχει μόνο πλεονεκτήματα. Σύμφωνα με πολλές κριτικές, παρόλο που διαθέτει δωρεάν έκδοση, θεωρείται ακριβό εργαλείο σε σύγκριση με άλλα εργαλεία CI/CD αν ένα έργο απαιτεί την εκτέλεση πολλών παράλληλων εργασιών [53,54].

2.5 Σύγκριση Εργαλείων CI

Όλα αυτά τα εργαλεία έχουν σχεδιαστεί για να διευκολύνουν και να αυτοματοποιήσουν την ανάπτυξη λογισμικού εξασφαλίζοντας την ποιότητα και τη μείωση του χρόνου. Η επιλογή ενός καλού εργαλείου CI για ένα έργο είναι κρίσιμη και μερικές φορές δύσκολη, ενώ μία λανθασμένη επιλογή και εκτίμησή του, θα μπορούσε να δημιουργήσει προβλήματα μειώνοντας τη συνολική ευελιξία, τον χρόνο υλοποίησης ενός λογισμικού καθώς και την ποιότητά του. Τα αποτελέσματα της έρευνας πρόκειται να δώσουν τη δυνατότητα σε ορισμένους προγραμματιστές να λάβουν μια απόφαση για ένα εργαλείο ή τουλάχιστον να αποκτήσουν κάποιες γνώσεις σχετικά με τα εργαλεία στην ανάπτυξη λογισμικού.

Παράμετρος	Jenkins	Travis	Codeship	Bamboo	TeamCity	CircleCI	Gitlab CI	Buddy
Κόστος	Δωρεάν	Δωρεάν για 30 ημέρες, μετά από $69/μήνα	Δωρεάν για 100 αναπτύξεις/μήνα, μετά από $49/μήνα	Δωρεάν για 30 ημέρες, μετά από $1200/έτος	Δωρεάν για 14 ημέρες, μετά ξεκινάει από $54/μήνα	Δωρεάν για 6000 λεπτά/μήνα, υπάρχει πλάνο από $15/μήνα αλλά απαιτούνται add-ons επί πληρωμή.	Δωρεάν για 400 λεπτά/μήνα, υπάρχει πλάνο από $19/μήνα και $99/μήνα.	Δωρεάν για 100 αναπτύξεις/μήνα, υπάρχει πλάνο από $75/μήνα
Συμβατότητα Github	Εξαιρετική	Εξαιρετική	Πρέπει να εγκατασταθεί η εφαρμογή Codeship GitHub App	Όχι. Εξαιρετικό για το JIRA ή το Stash	Καλή	Πολύ καλή	Ιδανική	Εξαιρετική
Χρήση / Περιβάλλον Εργασίας	Όχι τόσο εύκολο. Χρειάζεται χρόνος εκμάθησης	Περίπλοκο Χρειάζεται χρόνος εκμάθησης	Εύκολο	Περίπλοκο Χρειάζεται χρόνος εκμάθησης	Εύκολο	Εύκολο	Όχι τόσο εύκολο. Χρειάζεται χρόνος εκμάθησης	Πολύ Εύκολο
Υποστήριξη	Ευρεία υποστήριξη από την κοινότητα	Περιορισμένη υποστήριξη	Υποστήριξη μέσω Live Chat ακόμη και με το δωρεάν πλάνο	Απαιτείται Πληρωμή	Απαιτείται Πληρωμή	Απαιτείται Πληρωμή	Για τεχνική υποστήριξη απαιτείται πληρωμή	Υποστήριξη μέσω Live Chat ακόμη και με το δωρεάν πλάνο.
Μηχανή διακομιστή	Βάσει διακομιστή	Cloud-based	Βάσει διακομιστή	Cloud-based	Cloud-based	Cloud-based	Cloud-based	Cloud-based
Συμβατότητα Wordpress	Συμβατό με προσθήκη Plugin.	Μικρή. Απαιτούνται γνώσεις κώδικα.	Μικρή. Απαιτούνται γνώσεις κώδικα.	Μικρή. Απαιτούνται γνώσεις κώδικα.	Μικρή. Απαιτούνται γνώσεις κώδικα.	Μικρή. Απαιτούνται γνώσεις κώδικα.	Μικρή. Απαιτούνται γνώσεις κώδικα.	100% Συμβατό. Δεν απαιτούνται γνώσεις κώδικα.
Ανάπτυξη μέσω FTP/SFTP	Απαιτείται Plugin	Απαιτούνται γνώσεις CMD	Απαιτούνται γνώσεις CMD	Απαιτούνται γνώσεις CMD	Ναι	Απαιτείται Plugin	Απαιτείται Plugin	Ναι
Χειροκίνητη έγκριση ανάπτυξης (Release)	Ναι	Όχι	Ναι	Ναι	Ναι	Ναι	Απαιτείται επιπλέον πληρωμή	Ναι
Ενσωματωμένο Περιβάλλον Docker	Ναι	Ναι	Ναι	Όχι	Ναι	Απαιτείται πληρωμή μηνιαίου πλάνου	Ναι	Ναι
Οπτικές Δοκιμές (Visual Tests)	Όχι	Όχι	Όχι	Όχι	Όχι	Όχι	Όχι	Ναι
Υποστηριζόμενες γλώσσες	Python, Ruby, Java, Android, C/C++	Android, C, C#, C++, Java, JavaScript, Pearl, PHP, Python, Ruby	Java, JVM, JavaScript με NodeJS, PHP, Python, Ruby, Dart	Όλες	C#, C++, JavaScript, Pearl, PHP, Python	Android, iOS, Java, JavaScript με NodeJS, Python, Ruby	Go, JavaScript, Vue.js, Ruby	Go, JavaScript, PHP, Python, Ruby, Java, Android, Node.js, React.js, React Native, Scala, Django, Angular, .NET Core

Πίνακας 1 Σύγκριση Εργαλείων CI

2.6 Συμπέρασμα και Επιλογή Εργαλείου CI

Βάσει λοιπόν του πίνακα σύγκρισης (Πίνακας 1), η επιλογή δεν ήταν τόσο εύκολη, καθώς τόσο το Jenkins, το Gitlab όσο και το Buddy προσφέρουν πολλές πολύτιμες υπηρεσίες και διαθέτουν πολύ σημαντικά χαρακτηριστικά, γι'αυτό και ξεχωρίσαν σε σχέση με τα υπόλοιπα εργαλεία. Αρκετά Ωστόσο, το Buddy ήταν αυτό που επιλέχθηκε έναντι όλων των άλλων εργαλείων για τους σκοπούς αυτής της διπλωματικής εργασίας.

Όντας ένα από τα νεότερα εργαλεία CI της αγοράς, οι ιδρυτές του Buddy φαίνεται πως έχουν ερευνήσει σε βάθος τον ανταγωνισμό, εντοπίζοντας αδυναμίες και προβλήματα που οι ίδιοι φρόντισαν να επιλύσουν ώστε να είναι τα σημεία που θα τους ξεχωρίζουν στην αγορά.

Πρώτο και σημαντικότερο, έχοντας ως απόφθεγμα τη φράση "The Easiest CI/CD EVER" φαίνεται πως έχουν δώσει ιδιαίτερη σημασία στην ίδια την πλατφόρμα, σχεδιάζοντας ένα πολύ εύχρηστο και σύγχρονο περιβάλλον εργασίας που καθιστά πολύ εύκολη και ξεκάθαρη την διαδικασία δημιουργίας ενός αγωγού CI. Μάλιστα, σε περίπτωση που προκύψει οποιοδήποτε πρόβλημα ή απορία, διαθέτει ολοκληρωμένους οδηγούς, videos καθώς και υποστήριξη 24/7 μέσω Live Chat, κάτι που το καθιστά ακόμη πιο εύκολο στη χρήση και στην εκμάθησή του.

Επίσης, το Buddy μέσω της έξυπνης ανίχνευσης αλλαγών, της υπερσύγχρονης προσωρινής αποθήκευσης δεδομένων καθώς και της δυνατότητας παράλληλων ενεργειών φαίνεται πως είναι το εργαλείο CI με την γρηγορότερη ανάπτυξη.

Επιπλέον, με την μεγάλη έκρηξη των πλατφορμών CMS τα τελευταία χρόνια, η ομάδα του Buddy φαίνεται να έχει προνοήσει, παρέχοντας 100% συμβατότητα με έτοιμες ενέργειες και λειτουργίες για Wordpress και Shopify, ένας λόγος που έπαιξε σημαντικό ρόλο στο να ξεχωρίσει από τα εργαλεία Jenkins και Gitlab που μέχρι εκείνη τη στιγμή δεν είχαν απορριφθεί, καθώς το πρακτικό μέρος της εργασίας, από τα αρχεία, τον δοκιμαστικό server μέχρι και τον server ανάπτυξης βασίστηκαν στην πλατφόρμα και την αρχιτεκτονική του Wordpress.

Ένας ακόμη λόγος που ξεχώρισε το Buddy έναντι των εργαλείων Jenkins και Gitlab ήταν η ευκολία που παρέχει στο ανέβασμα των αρχείων μέσω FTP/SMTP. Στο Buddy η συγκεκριμένη ενέργεια προστίθεται στον αγωγό με το πάτημα ενός κουμπιού, ενώ δίνεται και η δυνατότητα φιλτραρίσματος των αρχείων ώστε να μην ανέβουν αρχεία που δεν χρειάζονται στην παραγωγή (π.χ. αρχεία test). Στο εργαλείο Jenkins, η διαδικασία είναι πιο σύνθετη και περίπλοκη και απαιτείται μεγαλύτερη εξειδίκευση, ενώ στο εργαλείο Gitlab πάλι η διαδικασία δεν είναι τόσο απλή όσο στο Buddy αλλά το σημαντικότερο είναι ότι υπάρχει περιορισμός χρήσης καθώς επιτρέπει ανέβασμα συγκεκριμένου όγκου αρχείων.

Τέλος, οι δυνατότητες που παρέχει το δωρεάν πλάνο του Buddy είναι μεγαλύτερες από το δωρεάν πλάνο του Gitlab, ενώ συγκρίνεται επάξια και με το πλάνο των 19$ το μήνα. Όσον αφορά το Jenkins σίγουρα το ότι είναι ανοιχτού κώδικα και δωρεάν για όλους

είναι από μόνο του ένα μεγάλο πλεονέκτημα, αλλά δεν είναι πάντα ο καθοριστικός παράγοντας ειδικά στην επιλογή ενός εργαλείου CI, όπου η προτεραιότητα και το κριτήριο επιλογής ενός μηχανικού, συνήθως είναι η συμβατότητα του εργαλείου με τις προδιαγραφές της εφαρμογής, καθώς και η ανάγκη κατασκευής, ελέγχου και ανάπτυξης του έργου στην παραγωγή όσο πιο εύκολα και γρήγορα είναι εφικτό να συμβεί, χωρίς να υπάρξουν προβλήματα και καθυστερήσεις.

Ο μόνος παράγοντας που θα μπορούσε να απορρίψει την επιλογή του Buddy θα ήταν η ενσωμάτωση μίας πολύ μικρής και απλής εφαρμογής με πολύ συχνή ανάπτυξη, π.χ. πολλές φορές μέσα στην ημέρα. Μία τέτοια περίπτωση θα ξεπερνούσε σίγουρα τις παροχές του δωρεάν πλάνου του Buddy και εφόσον πρόκειται για μία εφαρμογή με απλή υλοποίηση, η επιλογή του Jenkins ως εργαλείο ανοιχτού κώδικα, ίσως να ήταν η επικρατέστερη επιλογή εφόσον τα βήματα της διαδικασίας θα ήταν εύκολα ακόμη και για κάποιον που δεν έχει την εμπειρία με το συγκεκριμένο εργαλείο, επομένως η εφαρμογή δεν θα έχανε σε χρόνο, ευκολία και ταχύτητα.

Στο επόμενο κεφάλαιο θα παρουσιαστεί η μεθοδολογία δημιουργίας μίας δοκιμαστικής εφαρμογής ενός εργαλείου CI, αναλύοντας όλα τα βήματα από την προετοιμασία και τη δημιουργία του αποθετηρίου στο Git και του λογαριασμού στην πλατφόρμα Buddy, μέχρι και το τελευταίο στάδιο που απαιτείται για να ολοκληρωθεί η υλοποίηση ενός αγωγού CI μέσω του εργαλείου Buddy καθώς και τη συνεργασία του με την πρακτική του Continuous Delivery για την επίτευξη ανάπτυξης στον διακομιστή παραγωγής.

3 ΚΕΦΑΛΑΙΟ 3ο : (Μεθοδολογία και Δοκιμαστική Εφαρμογή ενός Εργαλείου CI)

Στο παρόν κεφάλαιο παρουσιάζεται η μεθοδολογία δημιουργίας μίας δοκιμαστικής εφαρμογής ενός εργαλείου CI. Αρχικά, παρουσιάζονται οι πλατφόρμες οι οποίες χρειάστηκαν για την υλοποίηση του συστήματος. Στη συνέχεια, όπως είδαμε και στο θεωρητικό υπόβαθρο το πρώτο στάδιο της υλοποίησης ήταν η δημιουργία αποθετηρίου για τον έλεγχο των εκδόσεων (Version Control) χρησιμοποιώντας ως πλατφόρμα το Github και η διαδικασία με την οποία έγινε το ανέβασμα όλων των αρχείων κώδικα από τον τοπικό διακομιστή στο νέο αποθετήριο που δημιουργήθηκε.

Έπειτα, έχοντας καταλήξει στο Buddy ως κατάλληλο εργαλείο CI για τη δημιουργία του συστήματος, δημιουργήθηκε λογαριασμός στην διαδικτυακή πλατφόρμα του Buddy και ακολούθησαν όλα τα απαραίτητα βήματα για τη δημιουργία νέου Project και τη διασύνδεσή του με το αποθετήριο που δημιουργήθηκε στο Github σε προηγούμενο βήμα.

Τέλος, με την ολοκλήρωση της διασύνδεσης του νέου Project με το αποθετήριο, τελευταίο βήμα ήταν η δημιουργία του αγωγού CI όπου αναλύθηκαν όλα τα βήματα και οι ενέργειες που χρειάζονται για να επιτευχθούν όλα τα στάδια που απαιτούνται για την υλοποίησή του, ενώ για την επίτευξη ανάπτυξης της νέας έκδοσης στον διακομιστή παράγωγής χρειάστηκε η προέκταση του αγωγού από CI σε CI/CD εφαρμόζοντας την πρακτική του Continuous Delivery.

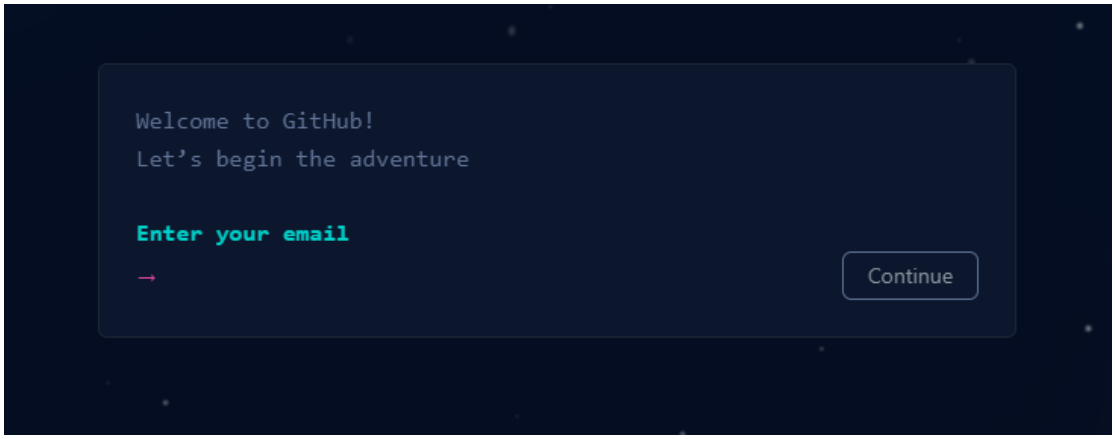
3.1 Προετοιμασία

Για την υλοποίηση του συστήματος χρησιμοποιήθηκαν οι ακόλουθες πλατφόρμες:

- Πλατφόρμα Source Control: Github
- Εργαλείο CI/CD: Buddy
- Πλατφόρμα Λειτουργίας Ιστοσελίδας: Wordpress
- Test Server: test.weblix.gr
- Production Server: production.weblix.gr

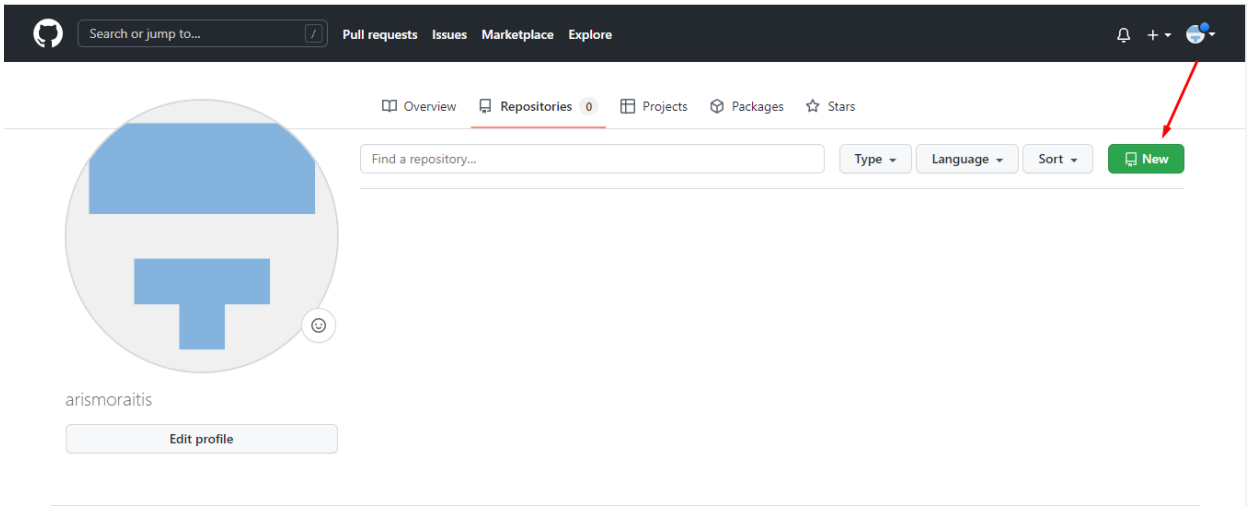
3.2 Δημιουργία Αποθετηρίου και Ανέβασμα Αρχείων

Για τη δημιουργία αποθετηρίου προϋπόθεση ήταν η δημιουργία λογαριασμού στην πλατφόρμα Github μέσα από το https://github.com/signup.

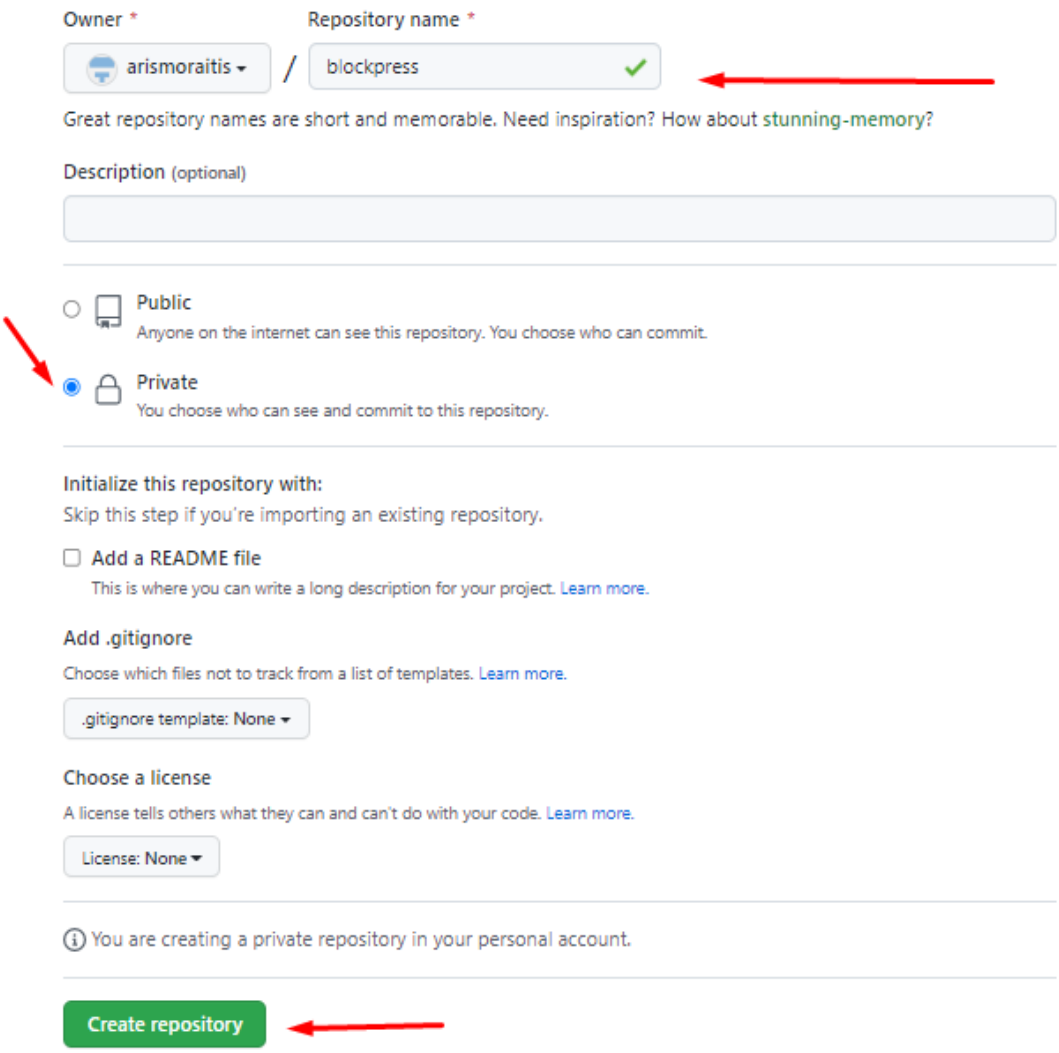


Εικόνα 3-1 Δημιουργία Λογαριασμού Github

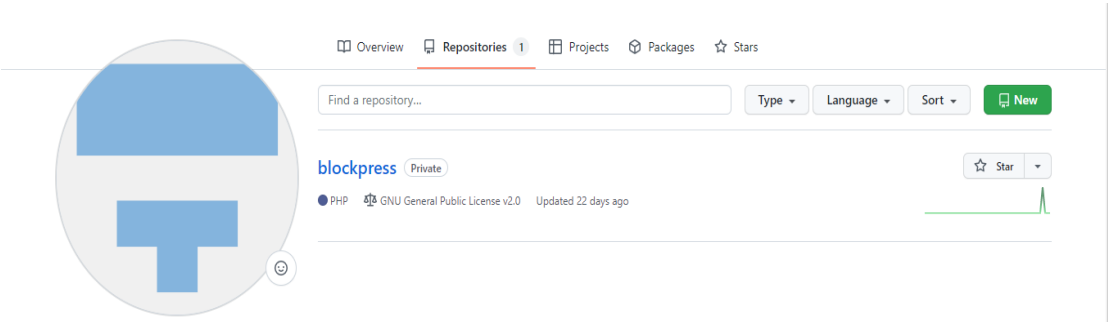
Μόλις έγινε η δημιουργία του νέου λογαριασμού σειρά είχε η δημιουργία ενός νέου ιδιωτικού αποθετηρίου που ονομάστηκε blockpress.



Εικόνα 3-2 Δημιουργία Αποθετηρίου - Βήμα 1



Εικόνα 3-3 Δημιουργία Αποθετηρίου -Βήμα 2



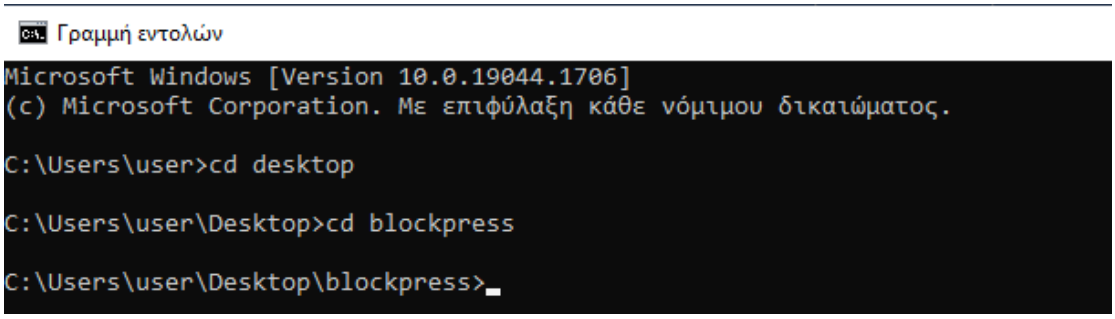
Εικόνα 3-4 Δημιουργία Αποθετηρίου -Βήμα 3

Επόμενο βήμα ήταν το ανέβασμα των αρχείων του θέματος στο αποθετήριο. Επειδή το GitHub δεν επιτρέπει το απευθείας ανέβασμα αρχείων μεγάλου όγκου (άνω των 25MB), ο μόνος τρόπος για να επιτευχθεί η συγκεκριμένη διαδικασία ήταν η χρήση της γραμμής εντολών (cmd). Οι εντολές που χρησιμοποιήθηκαν αναγράφονται παρακάτω και είναι οι προεπιλεγμένες εντολές που δίνει το ίδιο το Github για τη διαδικασία ανεβάσματος μεγάλων αρχείων σε ένα αποθετήριο.

Εντολές Github:

- git init
- git status
- git add **.** (η τελεία δίνει την εντολή προσθήκης όλων των αρχείων που βρίσκονται στον φάκελο που έχει δημιουργηθεί το αρχείο git).
- git commit -m "first commit"
- git remote add origin (url of github rep)
- git push -u origin master

Συγκεκριμένα λοιπόν, το πρώτο βήμα ήταν να διευκρινιστεί σε ποιο φάκελο βρίσκονται τα αρχεία που θα ανέβουν στο αποθετήριο. Ο φάκελος blockpress ήταν στην επιφάνεια εργασίας επομένως στη γραμμή εντολών γράφτηκαν αρχικά οι εντολές cd desktop και cd blockpress.



Εικόνα 3-5 Εφαρμογή Εντολών Github μέσω CMD – Βήμα 1

Στη συνέχεια ήταν η στιγμή να χρησιμοποιηθεί η εντολή git init όπου δημιουργήθηκε ένας κρυφός φάκελος .git μέσα στο φάκελο blockpress. Το git status έδωσε την αναφορά στο Github για το ποια αρχεία βρίσκονται στον φάκελο και με την εντολή git add . έγινε η προσθήκη των αρχείων μέσα στον φάκελο (.git).

Έπειτα, με την εντολή **git commit -m "first commit"** έγινε η ενσωμάτωση όλων των αρχείων σε ένα, ακολούθησε η εντολή **git remote add origin https://github.com/arismoraitis/blockpress.git** όπου το ενσωματωμένο αρχείο ήταν πλέον έτοιμο για ανέβασμα, και τέλος, με την εντολή **git push -u origin master** ξεκίνησε και ολοκληρώθηκε η διαδικασία του ανεβάσματος αποσυμπιέζοντας το αρχείο και επαναφέροντας τα αρχεία του θέματος στην αρχική τους μορφή.

Με την παραπάνω διαδικασία το αποθετήριο blockpress είχε πλέον όλα τα αρχεία της τελευταίας έκδοσης του έργου.

```
C:\Users\user\Desktop\blockpress>git init
Initialized empty Git repository in C:/Users/user/Desktop/blockpress/.git/

C:\Users\user\Desktop\blockpress>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        LICENSE
        assets/
        bin/
        build/
        composer.json
        functions.php
        inc/
        index.php
        languages/
        package-lock.json
        package.json
        parts/
        phpunit.xml.dist
        readme.txt
        screenshot.png
        style.css
        templates/
        tests/
        theme.json
        vendor/
        woocommerce/

nothing added to commit but untracked files present (use "git add" to track)

C:\Users\user\Desktop\blockpress>git add .
```

Εικόνα 3-6 Εφαρμογή Εντολών Github μέσω CMD – Βήμα 2

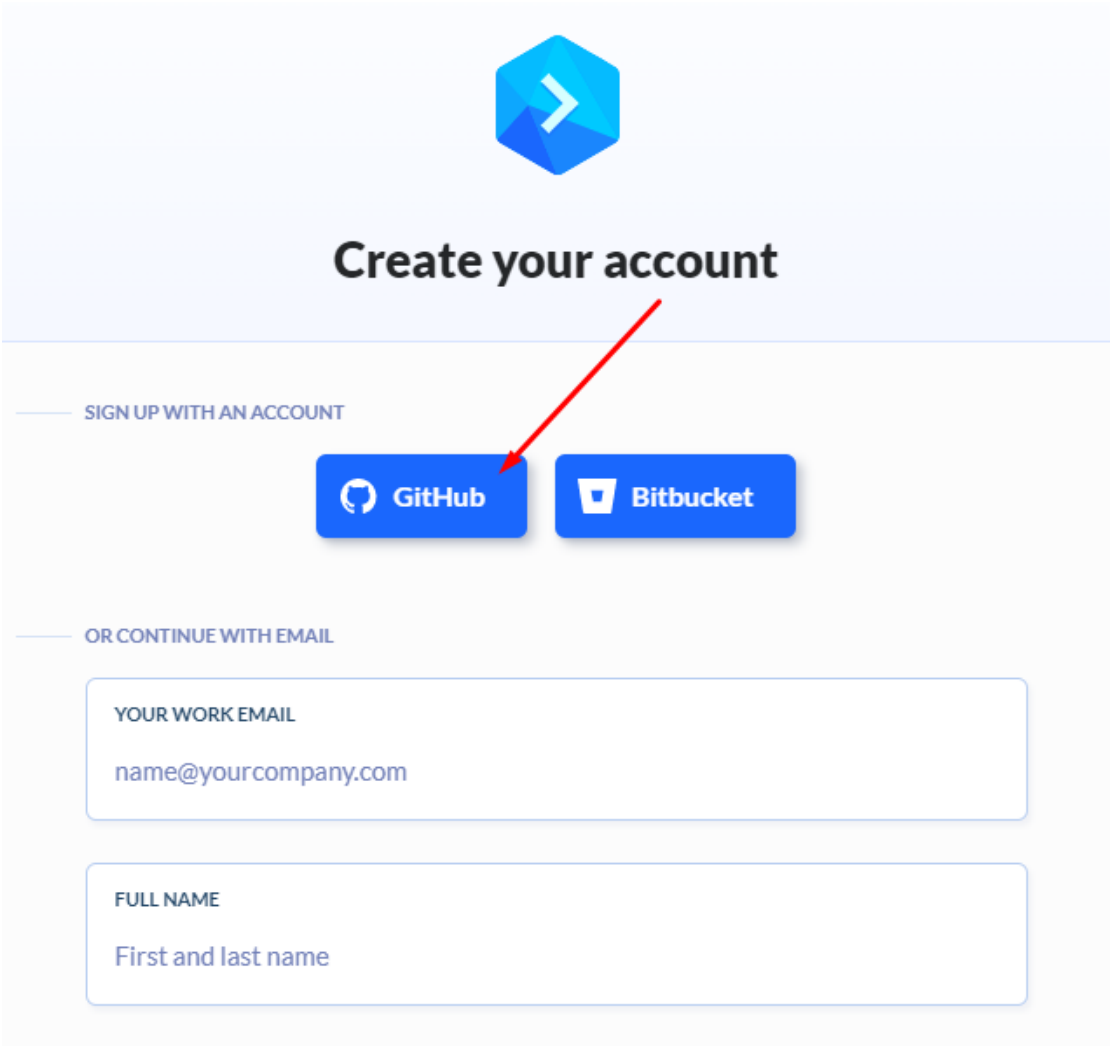
```
C:\Users\user\Desktop\blockpress>git push -u origin master
Enumerating objects: 307, done.
Counting objects: 100% (307/307), done.
Delta compression using up to 4 threads
Compressing objects: 100% (298/298), done.
Writing objects: 100% (307/307), 3.25 MiB | 1.19 MiB/s, done.
Total 307 (delta 45), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (45/45), done.
To https://github.com/arismoraitis/blockpress.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

C:\Users\user\Desktop\blockpress>
```

Εικόνα 3-7 Εφαρμογή Εντολών Github μέσω CMD – Βήμα 3

3.3 Δημιουργία Λογαριασμού στο Buddy

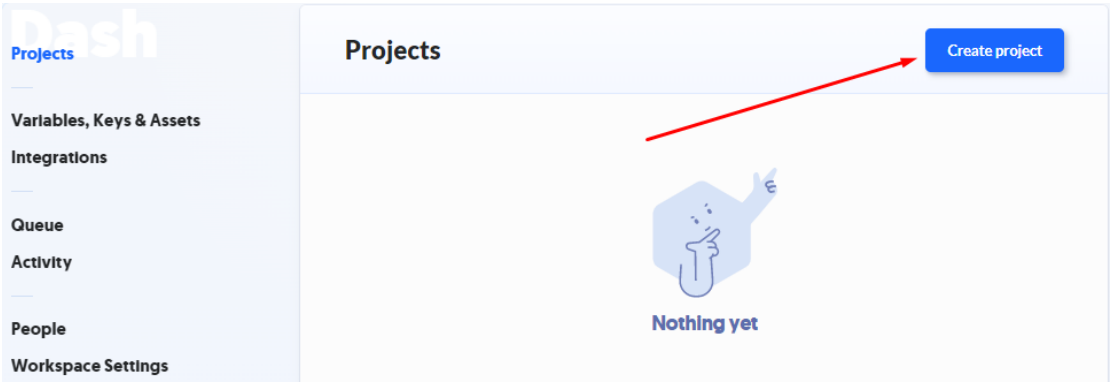
Αρχικά, για τη δημιουργία λογαριασμού μπήκαμε στο https://buddy.works/sign-in. Η δημιουργία ήταν πολύ απλή καθώς μπορούσε να γίνει κάνοντας απευθείας διασύνδεση με τον λογαριασμό Github που είχαμε ήδη.



Εικόνα 3-8 Δημιουργία Λογαριασμού στην Πλατφόρμα Buddy

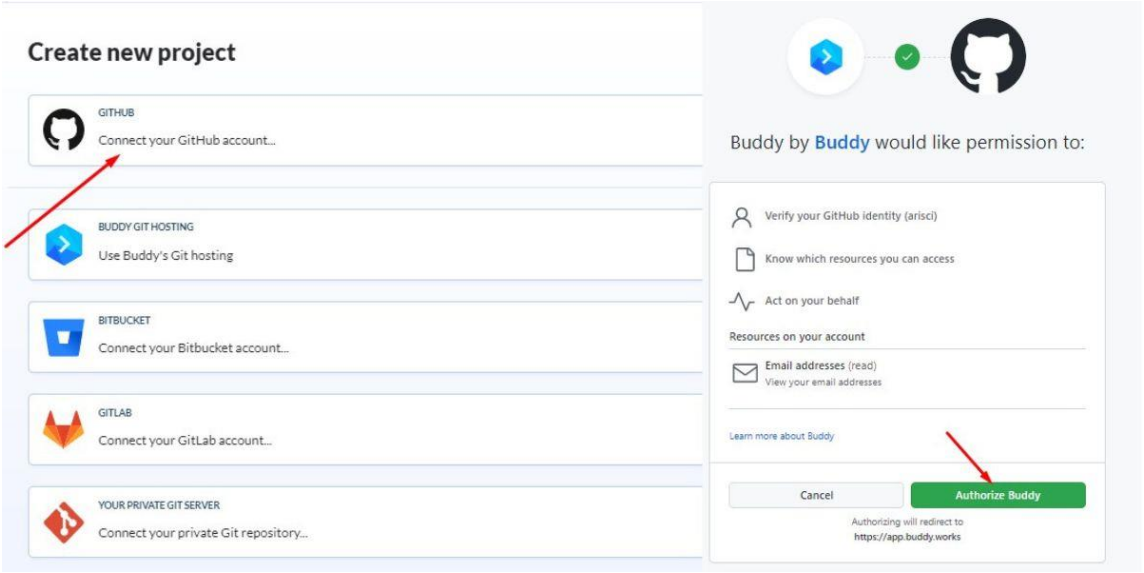
3.4 Δημιουργία Project & Σύνδεση Αποθετηρίου στο Buddy

Μόλις ολοκληρώσαμε την εγγραφή, μας εμφανίστηκε το περιβάλλον διαχείρισης του Buddy. Το πρώτο βήμα ήταν η δημιουργία Project.



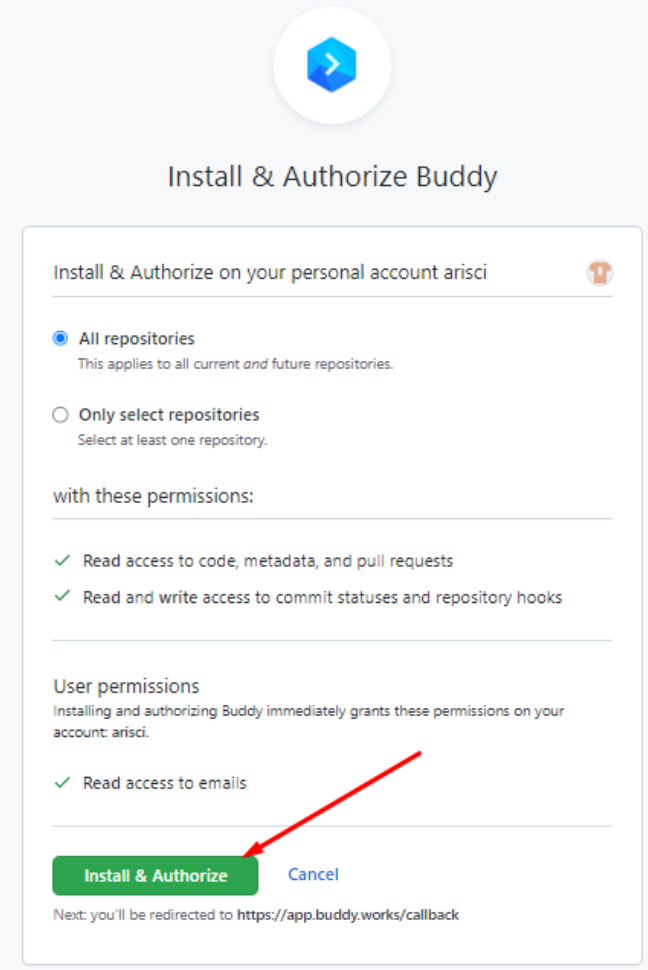
Εικόνα 3-9 Δημιουργία Project στο Buddy

Το Buddy μας έδινε τη δυνατότητα να φιλοξενήσουμε το project μας σε διάφορες πλατφόρμες ή ακόμη και μέσα σε δικό του εσωτερικό server. Έχοντας όμως δημιουργήσει λογαριασμό στο Buddy μέσω του λογαριασμού στο Github και έχοντας τα αρχεία κώδικα εκεί, η φιλοξενία του project επιλέχθηκε να γίνει μέσω αυτού.



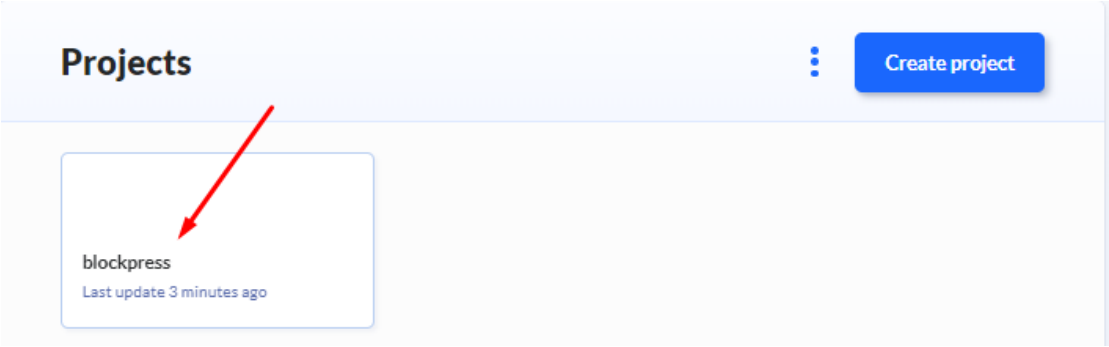
Εικόνα 3-10 Διασύνδεση Github με Buddy

Ολοκληρώνοντας τη σύνδεση του νέου project με το Github, σειρά είχε η διασύνδεση του project με τα αποθετήρια που είχαμε δημιουργήσει στο Github. Επομένως, έχοντας δημιουργήσει μόνο το αποθετήριο blockpress, το Buddy συνδέθηκε με το συγκεκριμένο αποθετήριο.



Εικόνα 3-11 Διασύνδεση του Αποθετηρίου στο Github με το Buddy

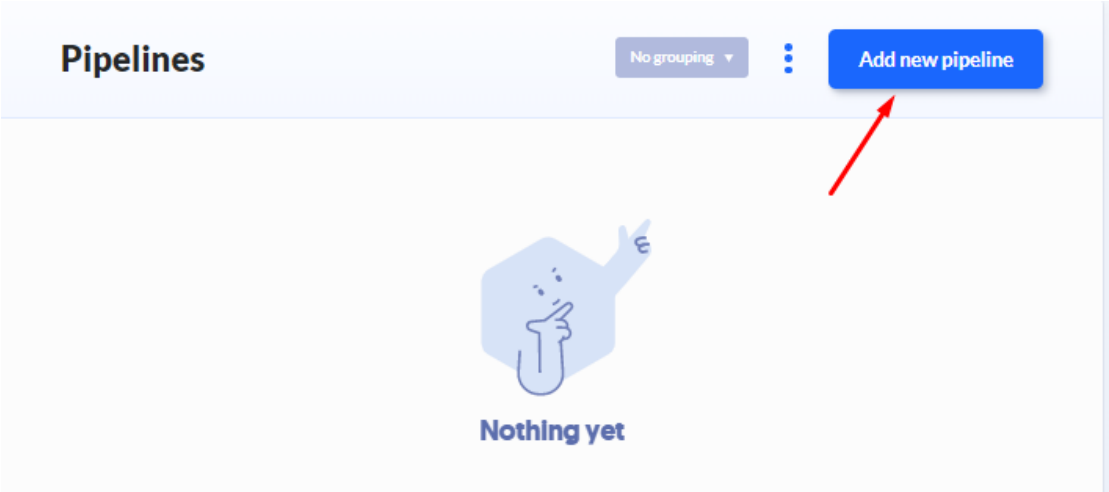
Ανανεώνοντας τη σελίδα είδαμε το νέο project να είναι πλέον διαθέσιμο και έτοιμο για να υποδεχθεί τον αγωγό CI που θα δημιουργηθεί για τους σκοπούς της εργασίας.



Εικόνα 3-12 Εμφάνιση Νέου Project στο Περιβάλλον Εργασίας του Buddy

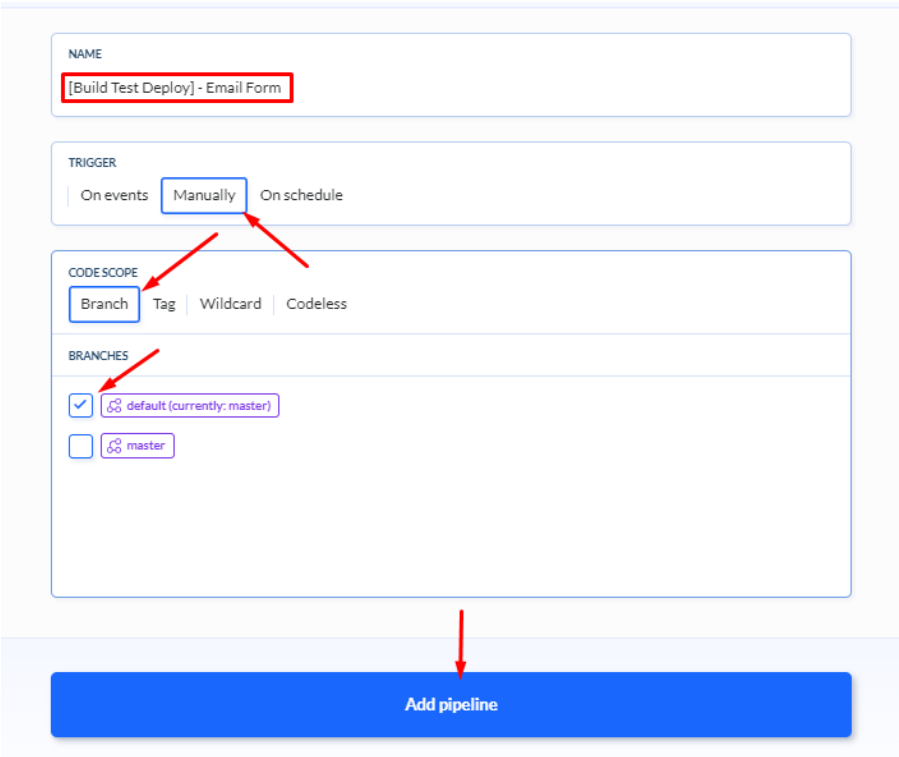
3.5 Δημιουργία Αγωγού & Διαδικασία Υλοποίησης

Για την δημιουργία του αγωγού το πρώτο βήμα ήταν εφόσον μπήκαμε μέσα στο project να πατήσουμε το κουμπί ”Add new pipeline”.



Εικόνα 3-13 Δημιουργία Αγωγού στο Buddy – Βήμα 1

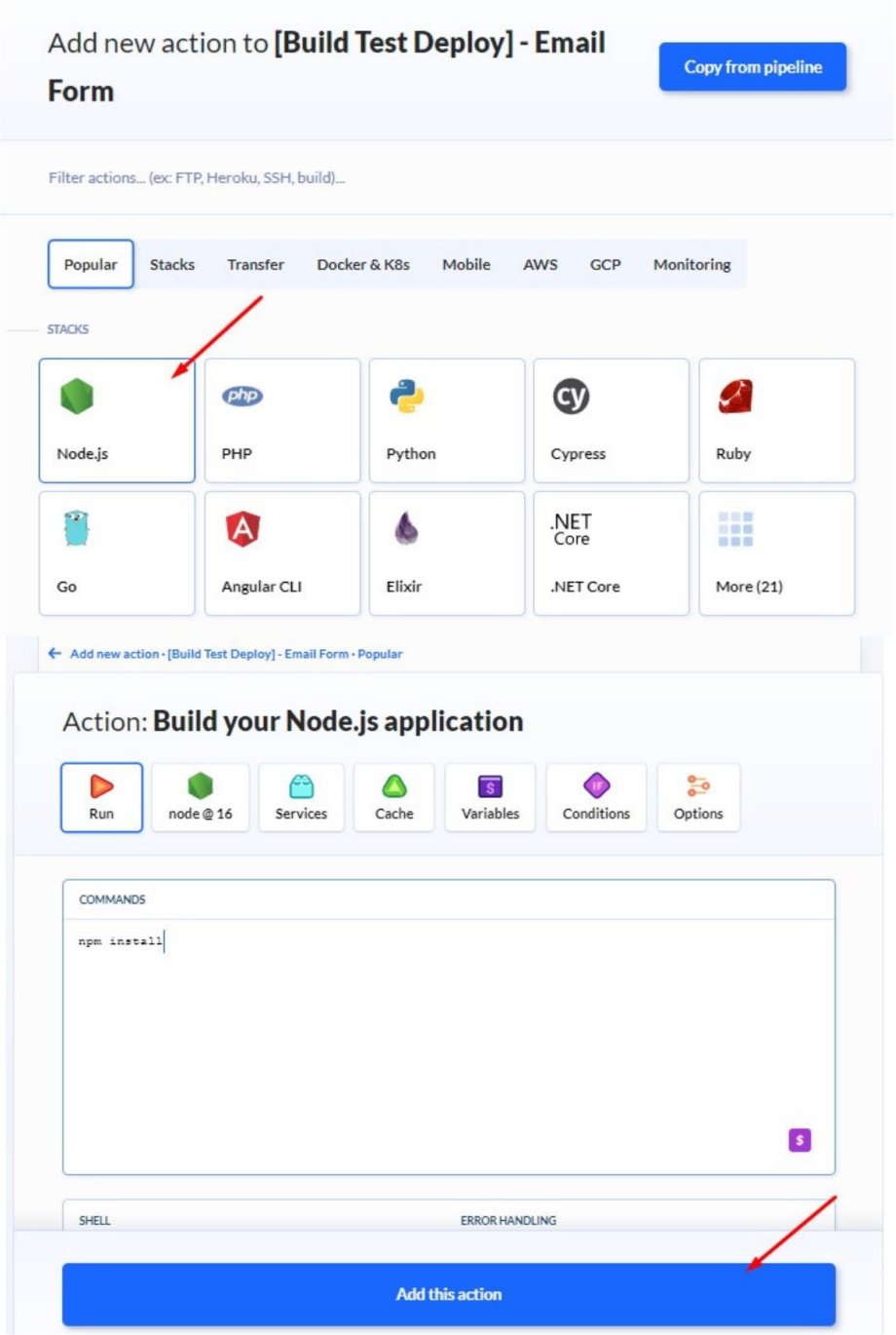
Έπειτα, γράφτηκε ένα όνομα για τον αγωγό, επιλέχθηκε ως Trigger η επιλογή Manually και ως Code Scope επιλέχθηκε το Branch και η επιλογή default. (Το Manually προτιμήθηκε καθώς για τους σκοπούς της εργασίας, οι ενέργειες του αγωγού δεν θα έπρεπε να πραγματοποιούνται αυτόματα την στιγμή που γίνεται οποιαδήποτε αλλαγή γίνεται μέσα στο Github, αλλά τη χρονική στιγμή που θα κρίνει ο developer ότι είναι η σωστή). Τέλος, πατώντας το κουμπί “Add pipeline” δημιουργήθηκε ο αγωγός του Project.



Εικόνα 3-14 Δημιουργία Αγωγού στο Buddy – Βήμα 2

3.5.1 Εγκατάσταση NPM

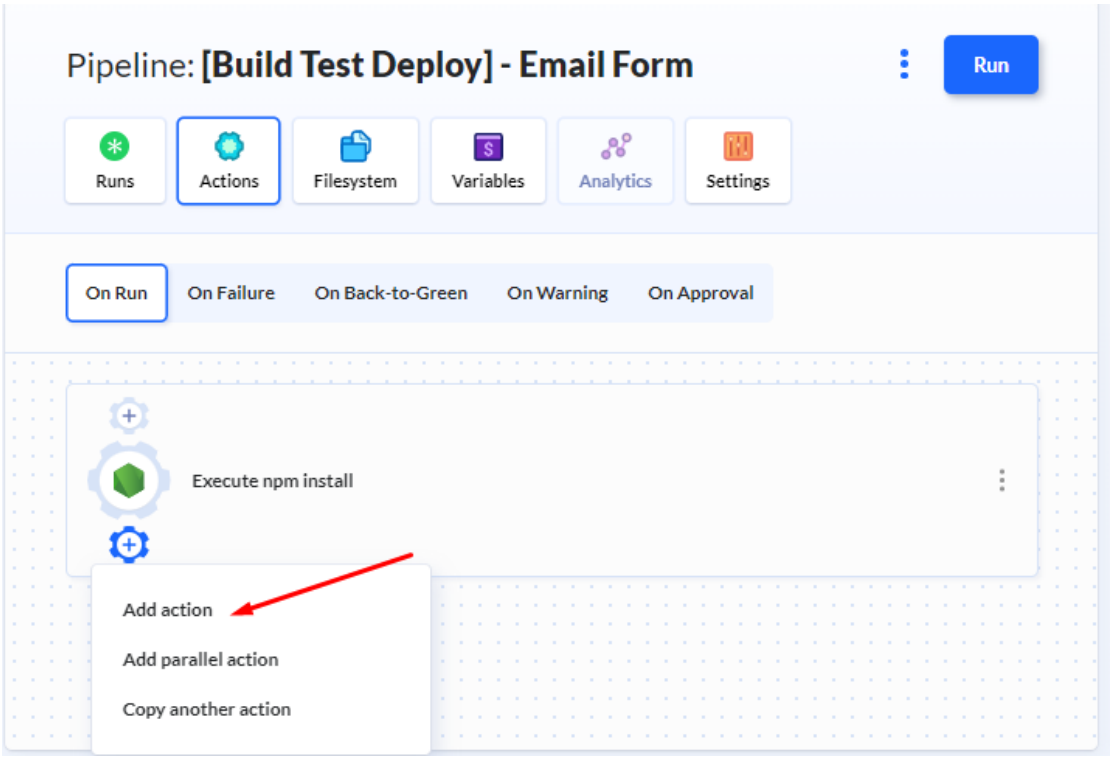
Αφού δημιουργήθηκε ο αγωγός, το περιβάλλον μας οδήγησε στην προσθήκη της πρώτης ενέργειας που θέλαμε να πραγματοποιηθεί. Η πρώτη ενέργεια λοιπόν ήταν η προσθήκη της ενέργειας Node.js μέσα από την οποία έγινε το πρώτο βήμα της διαδικασίας που ήταν η εγκατάσταση του npm το οποίο ήταν απαραίτητο καθώς μέσω αυτού εγκαταστάθηκαν όλες οι βιβλιοθήκες που θα χρησίμευαν στη συνέχεια για την ενέργεια του build.



Εικόνα 3-15 Εγκατάσταση NPM

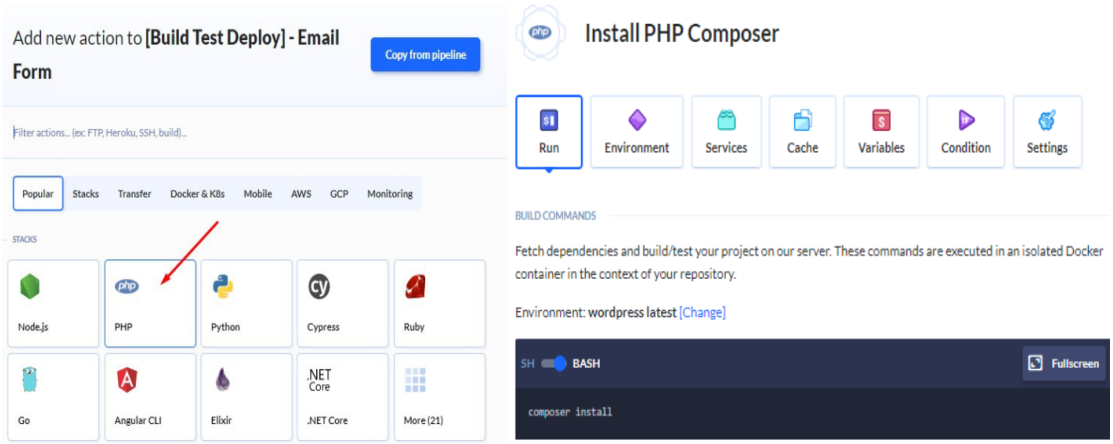
3.5.2 Εγκατάσταση Composer

Αφού δημιουργήθηκε η πρώτη ενέργεια και εμφανίστηκε στο περιβάλλον, για την προσθήκη δεύτερης ενέργειας έπρεπε να πατηθεί το γρανάζι με την ένδειξη + και να επιλεχθεί η επιλογή Add action.



Εικόνα 3-16 Προσθήκη Νέας Ενέργειας στον Αγωγό

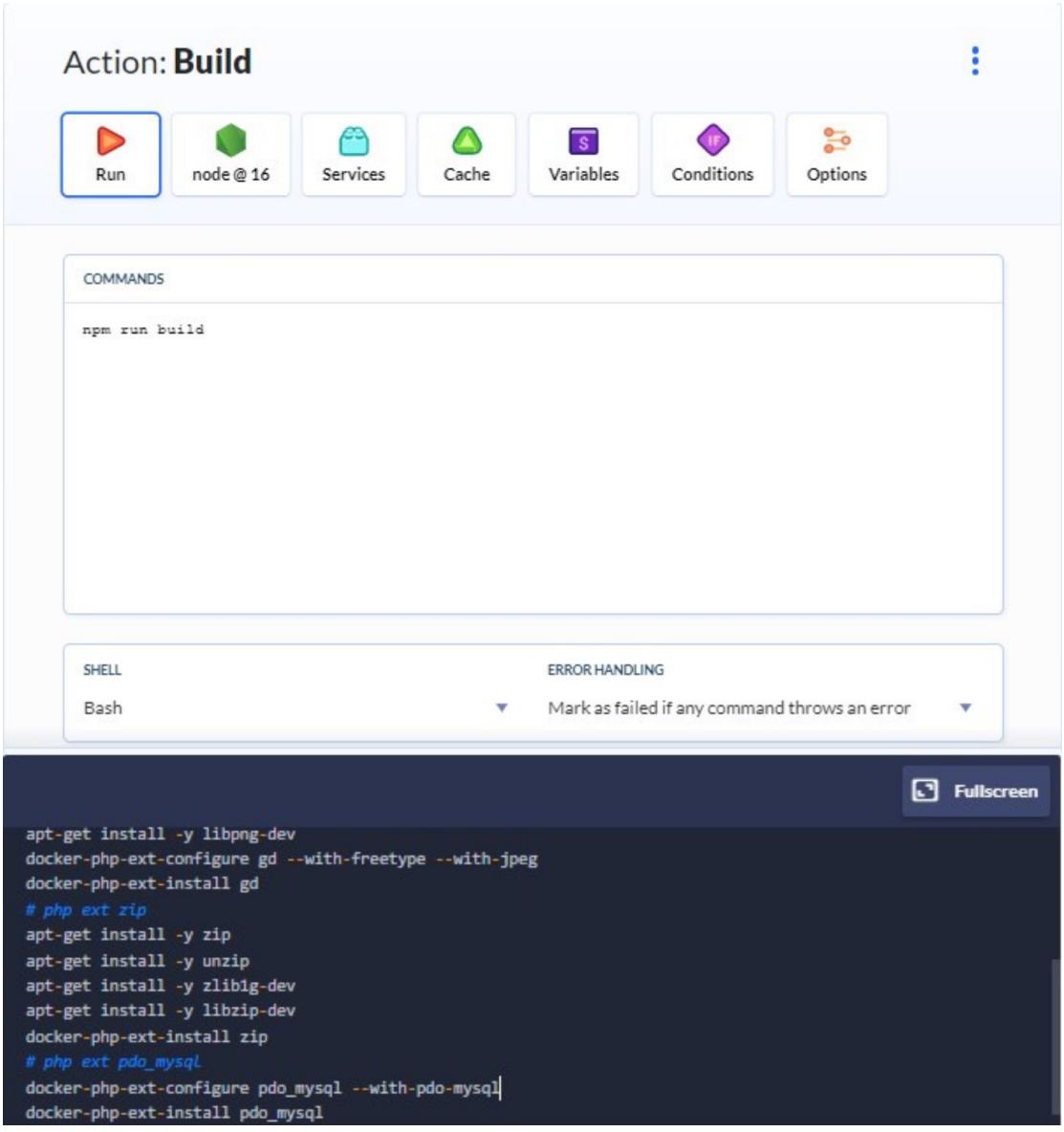
Ως δεύτερη ενέργεια λοιπόν, επιλέχθηκε η PHP, μέσω της οποίας έγινε η εγκατάσταση του composer μέσω του οποίου εγκαταστάθηκαν όλες οι βιβλιοθήκες που θα ήταν απαραίτητες για την πορεία του αγωγού και κυρίως για την πραγματοποίηση των test τα οποία ήταν βασισμένα στη γλώσσα PHP.



Εικόνα 3-17 Εγκατάσταση PHP Composer

3.5.3 Διαδικασία Continuous Building

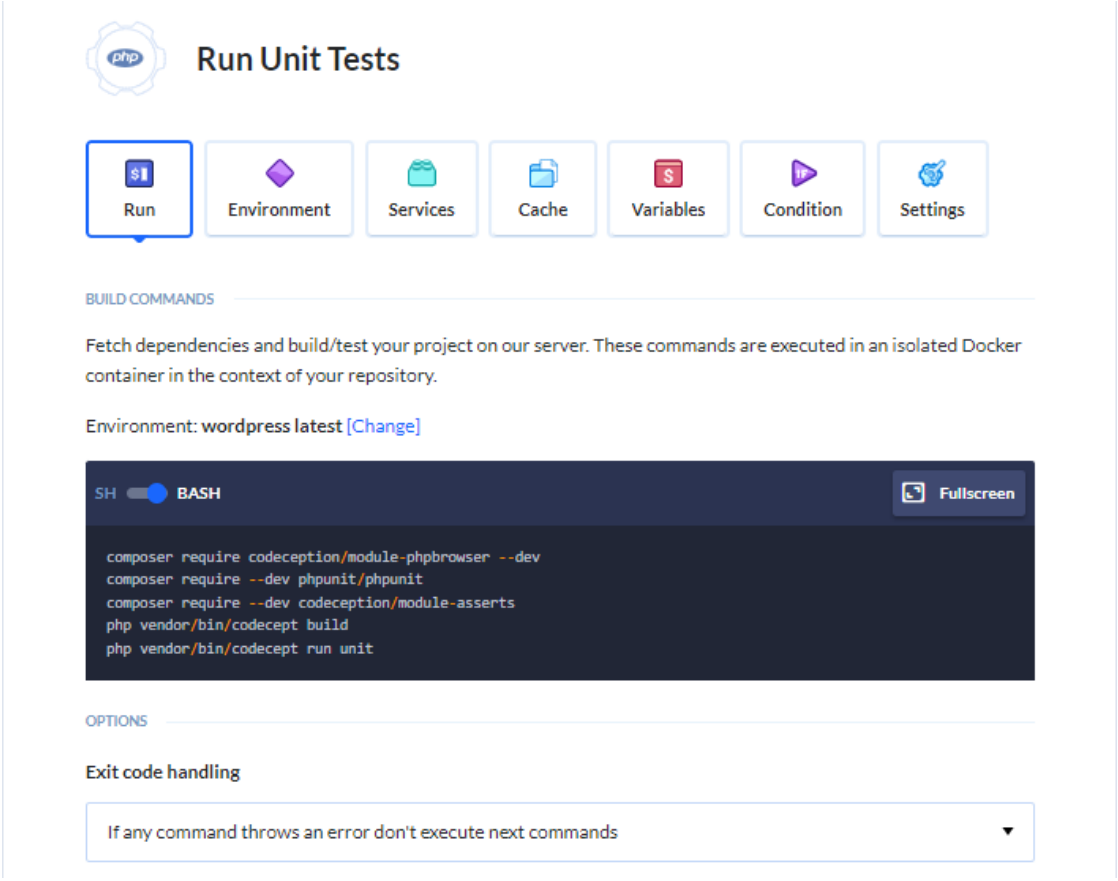
Στη συνέχεια προσθέτωντας άλλη μία ενέργεια Node.js ακολούθησε η διαδικασία του Build όπου ο κώδικας ελέγχθηκε για τυχόν σφάλματα και συμπιέστηκε μέσω docker χρησιμοποιώντας τις παρακάτω εντολές που υπήρχαν προεπιλεγμένες από το Buddy.



Εικόνα 3-18 Διαδικασία Build

3.5.4 Unit Tests

Επόμενο Βήμα είναι τα Unit Tests. Για την επίτευξη συμβατότητας των tests με τα αρχεία του Project χρησιμοποιήθηκαν εντολές Codeception (το Codeception είναι ένα πλαίσιο-framework δοκιμών PHP) οι οποίες έχουν δημιουργηθεί ειδικά για δοκιμές σε εφαρμογές PHP, στη γλώσσα δηλαδή όπου δημιουργήθηκαν και τα unit tests του Project.



Εικόνα 3-19 Unit Tests

Για τα unit tests αρχικά, δημιουργήθηκε αρχείο Email.php το οποίο περιείχε κώδικα PHP ο οποίος δήλωνε στο σύστημα πότε ένα email θεωρείται έγκυρο και πότε όχι.

```php
<?php declare(strict_types=1);
final class Email
{
    private $email;

    private function __construct(string $email)
    {
        $this->ensureIsValidEmail($email);

        $this->email = $email;
    }

    public static function fromString(string $email): self
    {
        return new self($email);
    }

    public function __toString(): string
    {
        return $this->email;
    }

    private function ensureIsValidEmail(string $email): void
    {
        if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
            throw new InvalidArgumentException(
                sprintf(
                    'Το email "%s" δεν είναι έγκυρο. Παρακαλώ, προσπαθήστε ξανά...',
                    $email
                )
            );
        }
    }
}
```

Εικόνα 3-20 Αρχείο EmailTest.php

Στη συνέχεια δημιουργήθηκε αρχείο EmailTest.php το οποίο περιείχε εντολές PHP οι οποίες χρησιμοποιήθηκαν με σκοπό να ελεγχθεί η λειτουργικότητα του αρχείου Email.php, δηλαδή αν όντως θα εγκριθεί ένα έγκυρο email ή θα εμφανίσει μήνυμα απόρριψης στην περίπτωση που συμπληρωθεί μη έγκυρο email (Εικόνα 3-20).

Έπειτα, για να τρέξουν τα συγκεκριμένα tests μέσω του αγωγού στο buddy χρειάστηκε να δημιουργηθεί αρχείο unit.suite.yml μέσα από το οποίο δόθηκε η εντολή στο framework codeception να τρέξει στη γραμμή εντολών τα συγκεκριμένα tests μέσω της εντολής «**php vendor/bin/codecept run tests/unit/EmailTest.php**».

```yaml
# Codeception Test Suite Configuration

# suite for unit (internal) tests.
# RUN `build` COMMAND AFTER ADDING/REMOVING MODULES.

actor: UnitTester
modules:
    enabled:
      - Asserts
```

Εικόνα 3-21 Αρχείο unit.suite.yml

Για να επιβεβαιωθεί λοιπόν η λειτουργικότητα των unit tests, η εντολή «**php vendor/bin/codecept run tests/unit/EmailTest.php**» έτρεξε 2 φορές, τη μία καταχωρώντας μη έγκυρο email (user.example.com) στο αρχείο EmailTest.php και την άλλη καταχωρώντας ένα έγκυρο email (user@example.com) αντίστοιχα.

```php
<?php declare(strict_types=1);
require_once __DIR__ . '/Email.php';

use PHPUnit\Framework\TestCase;

final class EmailTest extends TestCase
{
    public function testCanBeCreatedFromValidEmailAddress(): void
    {
        $this->assertInstanceOf(
            Email::class,
            Email::fromString('user.example.com')
        );
    }

    public function testCannotBeCreatedFromInvalidEmailAddress(): void
    {
        $this->expectException(InvalidArgumentException::class);

        Email::fromString('invalid');
    }

        public function testCanBeUsedAsString(): void
    {
        $this->assertEquals(
            'user.example.com',
            Email::fromString('user.example.com')
        );
    }
}
```

Εικόνα 3-22 Αρχείο EmailTest.php – Μη Έγκυρο Email

```
Loading assets
php vendor/bin/codecept run tests/unit/EmailTest.php
Codeception PHP Testing Framework v4.2.1 https://helpukrainewin.org
Powered by PHPUnit 8.5.28 #StandWithUkraine
Unit Tests (3) ----------------------------------------
E EmailTest: Can be created from valid email address (0.00s)
✓ EmailTest: Cannot be created from invalid email address (0.00s)
E EmailTest: Can be used as string (0.00s)
------------------------------------------------------------
Time: 187 ms, Memory: 10.00 MB
There were 2 errors:
---------
1) EmailTest: Can be created from valid email address
 Test  tests/unit/EmailTest.php:testCanBeCreatedFromValidEmailAddress

  [InvalidArgumentException] Το email "user.example.com" δεν είναι έγκυρο. Παρακαλώ, προσπαθήστε ξανά...

#1  /buddy/blockpress/tests/unit/Email.php:26
#2  /buddy/blockpress/tests/unit/Email.php:8
#3  /buddy/blockpress/tests/unit/Email.php:15
#4  /buddy/blockpress/tests/unit/EmailTest.php:12
#5  /buddy/blockpress/vendor/bin/codecept:115
---------
```

Εικόνα 3-23 Unit Test - Μη Έγκυρο Email

```php
<?php declare(strict_types=1);
require_once __DIR__ . '/Email.php';

use PHPUnit\Framework\TestCase;

final class EmailTest extends TestCase
{
    public function testCanBeCreatedFromValidEmailAddress(): void
    {
        $this->assertInstanceOf(
            Email::class,
            Email::fromString('user@example.com')
        );
    }

    public function testCannotBeCreatedFromInvalidEmailAddress(): void
    {
        $this->expectException(InvalidArgumentException::class);

        Email::fromString('invalid');
    }

        public function testCanBeUsedAsString(): void
    {
        $this->assertEquals(
            'user@example.com',
            Email::fromString('user@example.com')
        );
    }
}
```

Εικόνα 3-24 Αρχείο EmailTest.php – Έγκυρο Email

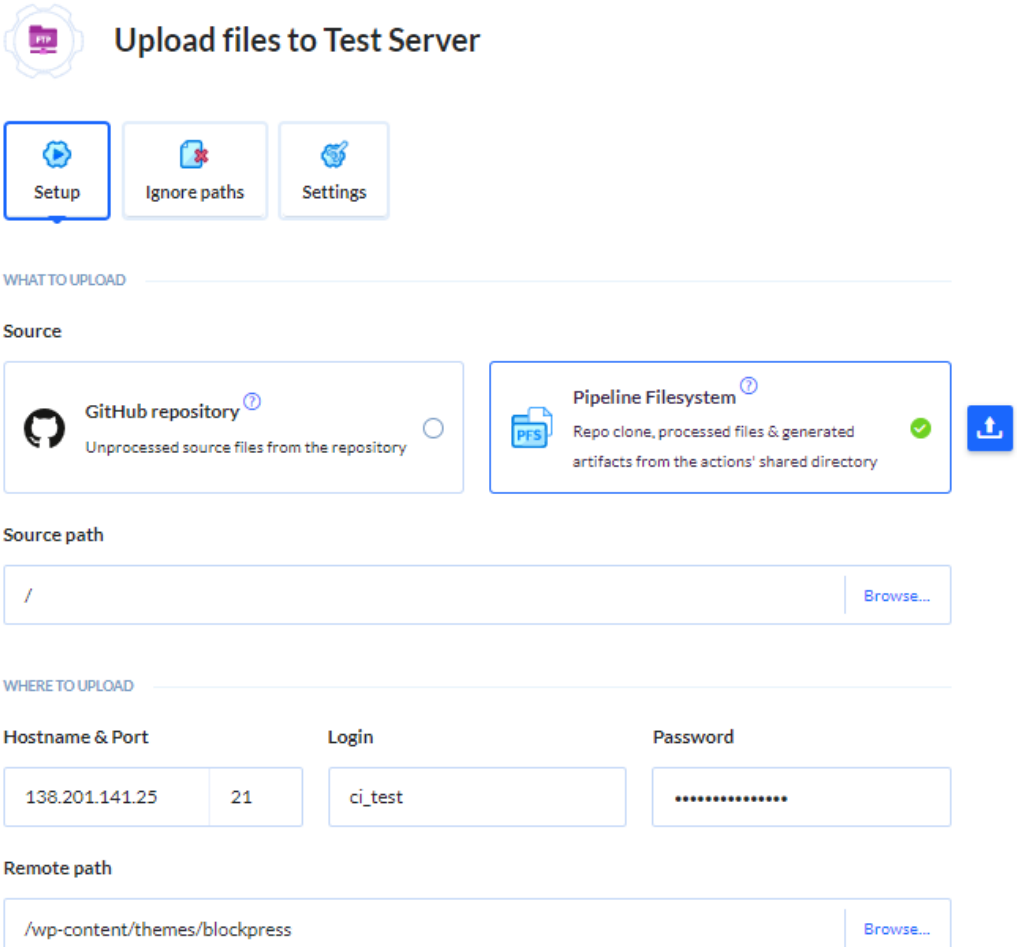
```
Loading assets
php vendor/bin/codecept run tests/unit/EmailTest.php
Codeception PHP Testing Framework v4.2.1 https://helpukrainewin.org
Powered by PHPUnit 8.5.28 #StandWithUkraine
Unit Tests (3) ----------------------------------------
✓ EmailTest: Can be created from valid email address (0.00s)
✓ EmailTest: Cannot be created from invalid email address (0.00s)
✓ EmailTest: Can be used as string (0.00s)
------------------------------------------------------------
Time: 180 ms, Memory: 10.00 MB
OK (3 tests, 3 assertions)
Build finished successfully!
```

Εικόνα 3-25 Unit Test - Έγκυρο Email

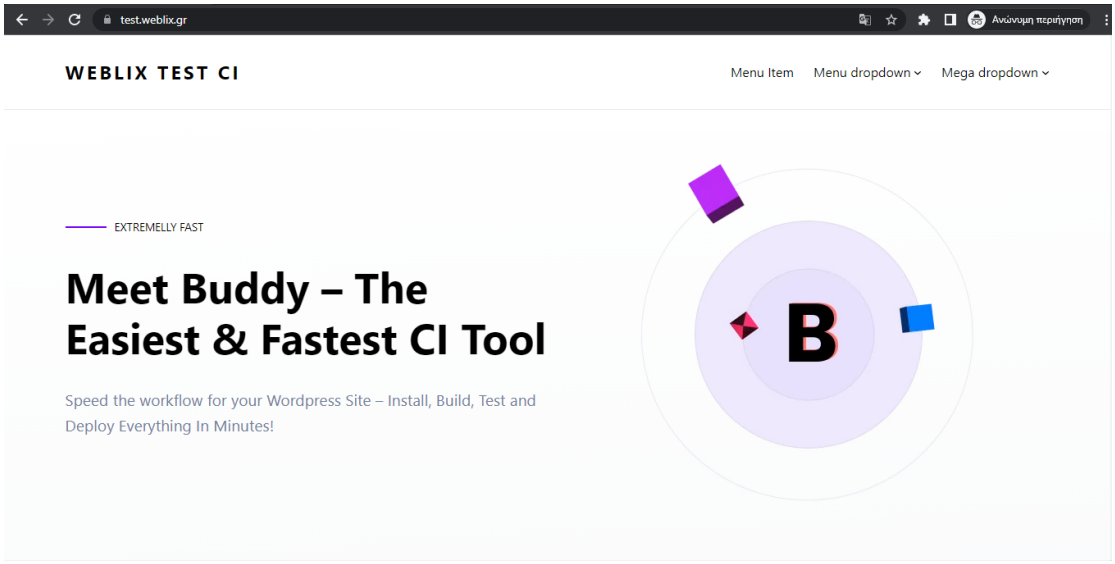
3.5.5 Ανέβασμα Αρχείων Σε Test Server

Εφόσον τα unit tests επαληθεύτηκαν, η επόμενη διαδικασία ήταν το ανέβασμα των αρχείων στον test server. Η διασύνδεση του Buddy με τον test server έγινε μέσω FTP,

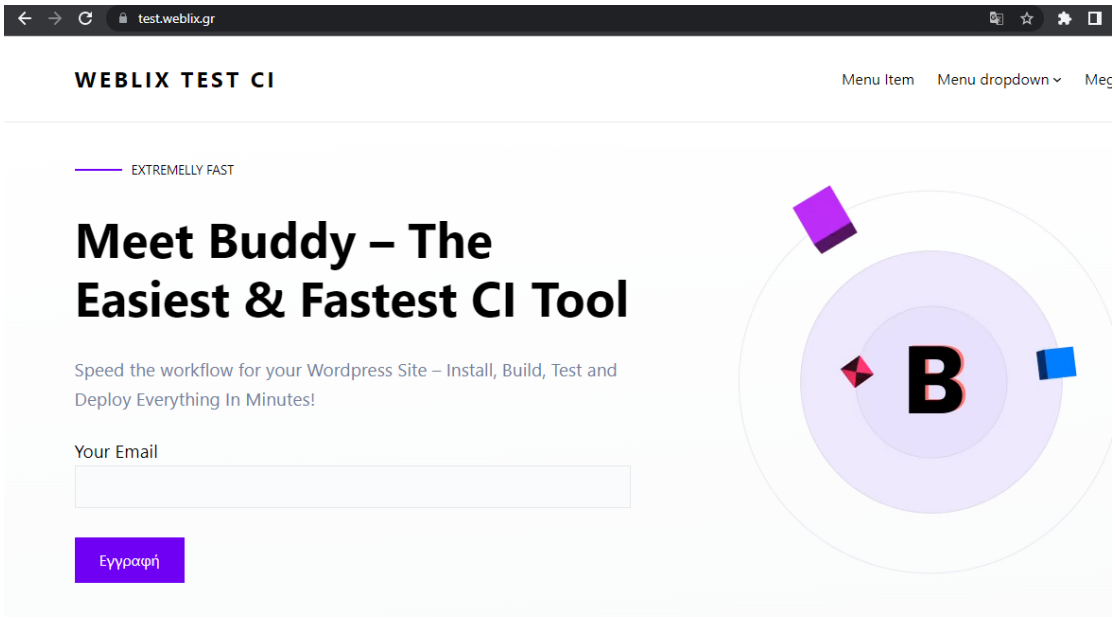
όπου για την επίτευξη της σύνδεσης αυτής χρειάστηκαν η IP που φιλοξενείται η δοκιμαστική ιστοσελίδα καθώς επίσης και η θύρα, το username και το password του λογαριασμού FTP που δημιουργήθηκε από το περιβάλλον του Δοκιμαστικού Server που φιλοξενείται η δοκιμαστική ιστοσελίδα. Τέλος, επιλέχθηκε το “Remote path”, δηλαδή ο φάκελος στον οποίο θα προστεθεί η φόρμα email όταν ολοκληρωθεί η διαδικασία και δημοσιευθεί στον server παραγωγής.



Εικόνα 3-26 Ανέβασμα Νέας Έκδοσης στον Test Server



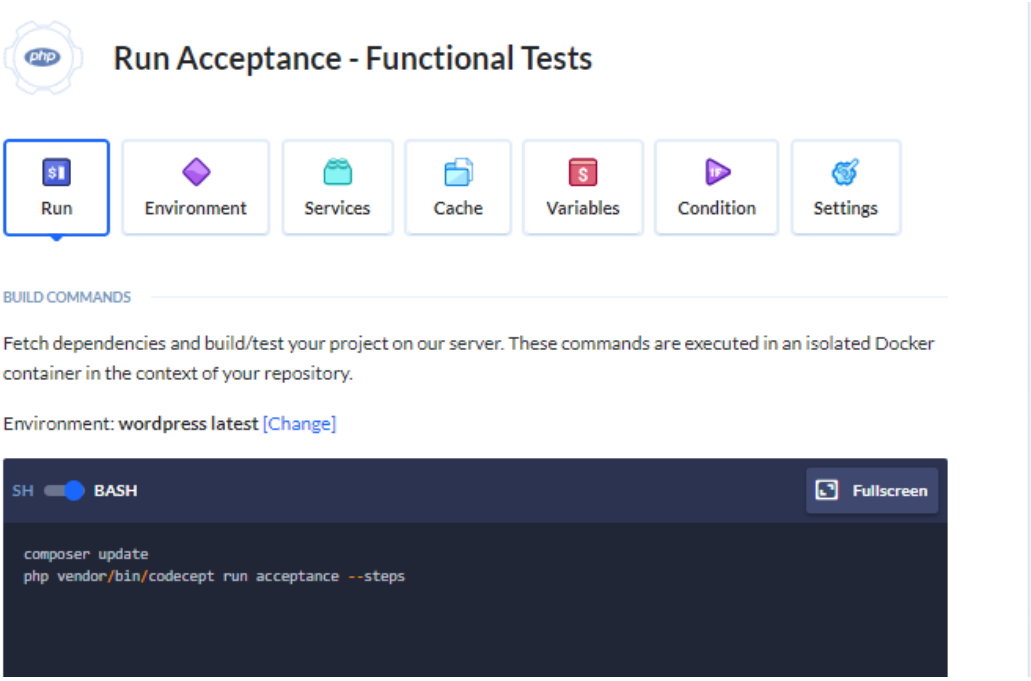
Εικόνα 3-27 Ιστοσελίδα Test - Πριν το Upload



Εικόνα 3-28 Ιστοσελίδα Test - Μετά το Upload

3.5.6 Acceptance & Funtional Tests

Ακολούθως, με την ολοκλήρωση του ανεβάσματος ο αγωγός ήταν έτοιμος να προχωρήσει στο στάδιο των acceptance και functional tests, τα οποία είχαν ως σκοπό να επαληθεύσουν ότι η νέα φόρμα email που προστέθηκε στην ιστοσελίδα είναι λειτουργική, δηλαδή εμφανίζει αρχικά τα στοιχεία της φόρμας που θέλαμε και στη συνέχεια το μήνυμα επιτυχής συμπλήρωσης όταν συμπληρώσουμε ένα έγκυρο email.



Εικόνα 3-29 Acceptance – Functional Tests

Για τα acceptance και functional tests αρχικά, δημιουργήθηκε αρχείο FirstCest.php το οποίο περιείχε κώδικα PHP ο οποίος έδινε εντολή στο σύστημα αρχικά, να επιβεβαιώσει ότι η βλέπει την αρχική σελίδα. Έπειτα, να ελέγχξει ότι υπάρχει η φράση «Your Email» ως Label του πεδίου και στη συνέχεια να καταχωρήσει ένα έγκυρο email στο πεδίο, να κάνει κλικ στο κουμπί «Εγγραφή» και τέλος, να δει το μήνυμα που θα επιβεβαιώσει την επιτυχημένη καταχώρηση του email. Όλα τα παραπάνω βήματα εάν ήταν επιτυχημένα και περνούσαν αυτά τα test, τότε θα είχαμε την επιβεβαίωση ότι η σελίδα υπάρχει και δεν εμφάνισε κάποιο πρόβλημα και η φόρμα email έχει εμφανιστεί επιτυχώς στη σελίδα και είναι λειτουργική καθώς θα μπορεί να καταχωρήσει emails.

```php
<?php
    /**
     * A single example test.
     */

class FirstCest
{
    public function frontpageWorks(AcceptanceTester $I)
    {
        $I->amOnPage('/');
        $I->see('Your Email');
    }

    public function submitform(AcceptanceTester $I)
    {
        $I->amOnPage('/');
        $I->fillField('email-840', 'example@weblix.gr');
        $I->click('Εγγραφή');
        $I->see('Η εγγραφή σας ολοκληρώθηκε επιτυχώς!');
    }

}
```

Εικόνα 3-30 Αρχείο FirstCest.php

Επιπρόσθετα, για να τρέξουν τα συγκεκριμένα tests μέσω του αγωγού στο buddy χρειάστηκε να δημιουργηθεί αρχείο **acceptance.suite.yml** μέσα από το οποίο γνωστοποιήθηκε η ιστοσελίδα στην οποία θα τρέξουν τα συγκεκριμένα tests (test.weblix.gr) ενώ παράλληλα δόθηκε και η εντολή στο στο framework codeception να τρέξει στη γραμμή εντολών τα συγκεκριμένα tests μέσω της εντολής «**php vendor/bin/codecept run acceptance --steps**».

```yaml
actor: AcceptanceTester
modules:
    enabled:
        - PhpBrowser:
            url: https://test.weblix.gr/
```

Εικόνα 3-31 Αρχείο acceptance.suite.yml

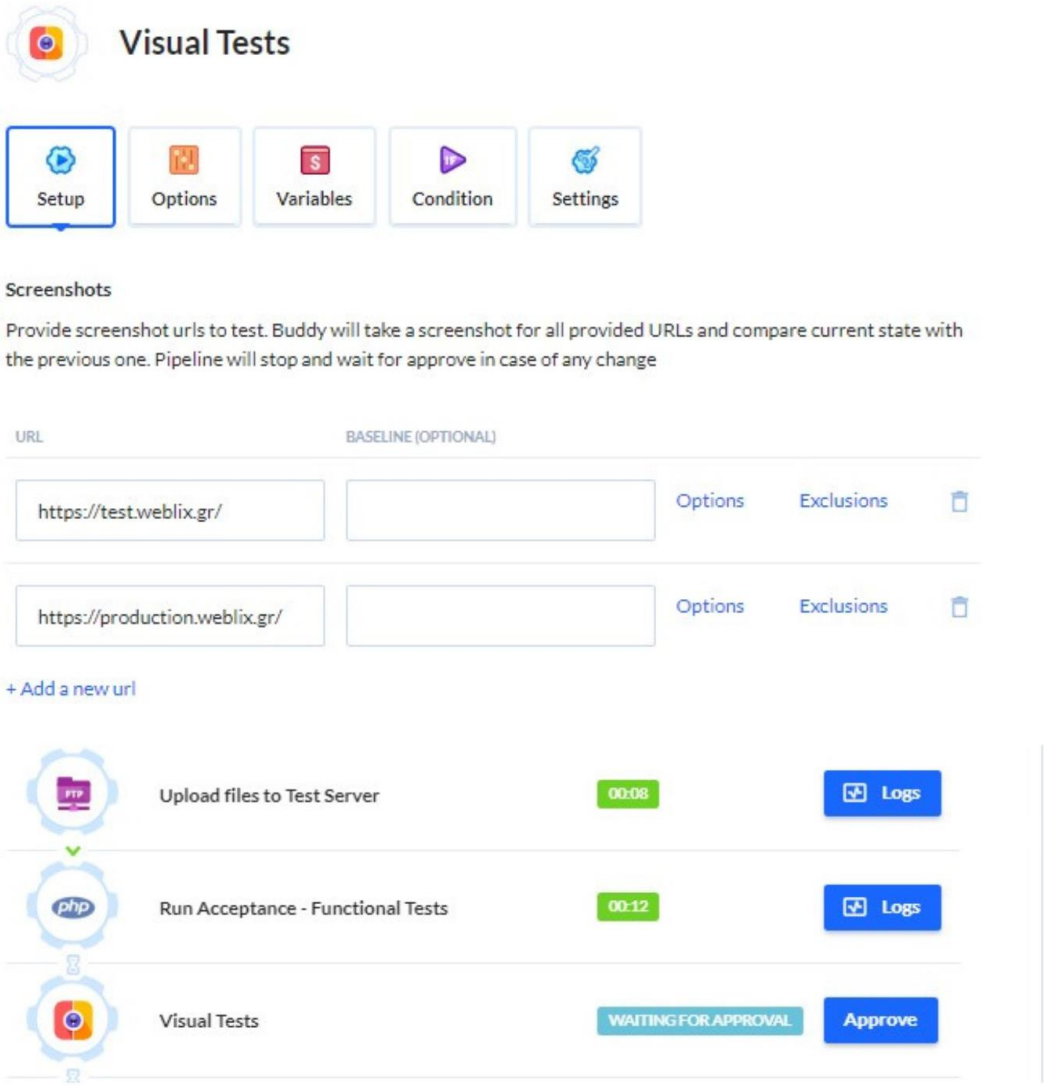
```
Loading assets
php vendor/bin/codecept run acceptance --steps
Codeception PHP Testing Framework v4.2.1 https://helpukrainewin.org
Powered by PHPUnit 8.5.28 #StandWithUkraine
Acceptance Tests (2) ---------------------------------------
FirstCest: Frontpage works
Signature: FirstCest:frontpageWorks
Test: tests/acceptance/FirstCest.php:frontpageWorks
Scenario --
 I am on page "/"
 I see "Your Email"
 PASSED
FirstCest: Submitform
Signature: FirstCest:submitform
Test: tests/acceptance/FirstCest.php:submitform
Scenario --
 I am on page "/"
 I fill field "email-840","example@weblix.gr"
 I click "Εγγραφή"
 I see "Η εγγραφή σας ολοκληρώθηκε επιτυχώς!"
 PASSED
------------------------------------------------------------
Time: 2.83 seconds, Memory: 12.00 MB
OK (2 tests, 2 assertions)
Build finished successfully!
```

Εικόνα 3-32 Acceptance & Functional Tests – Επιτυχία

Σε αυτό το σημείο διαπιστώθηκε ότι τα στάδια του αγωγού CI δεν ήταν αρκετά ώστε η νέα έκδοση να δημοσιευτεί στον server παραγωγής. Το Buddy λοιπόν, έδινε τη δυνατότητα επέκτασης του αγωγού από CI σε CI/CD όπου ως CD, επιλέχθηκε τελικά η πρακτική του Continuous Delivery έναντι του Continuous Deployment, για λόγους ασφαλείας και καλύτερου ελέγχου του συστήματος. Για την ενσωμάτωση του Continuous Delivery χρησιμοποιήθηκαν δύο ενέργειες, μία για το στάδιο Release (Visual Tests) και μία για το στάδιο Deploy (Upload via FTP).

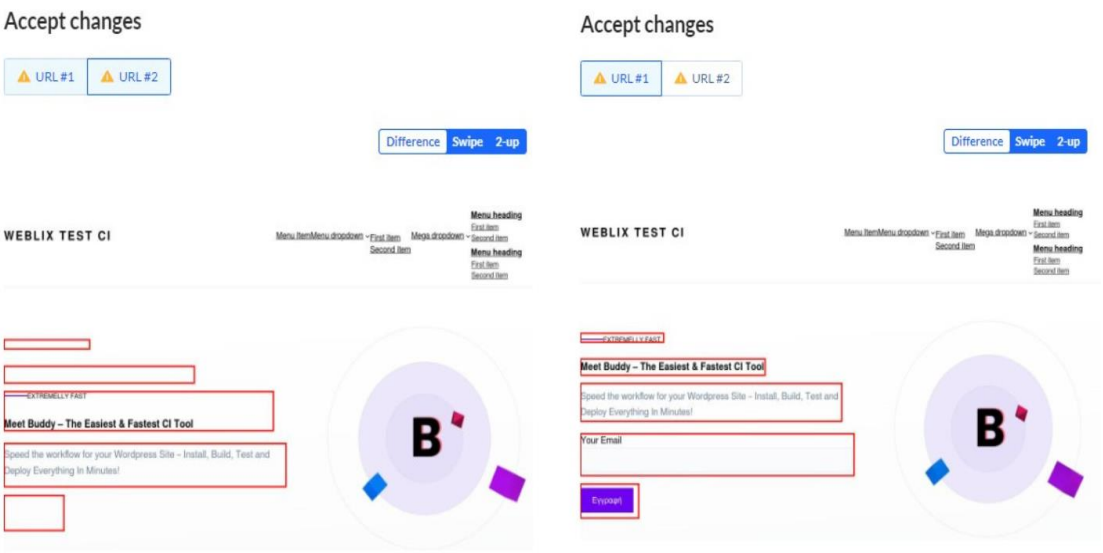
3.5.7 Visual Tests (Στάδιο Release)

Το εργαλείο Buddy διαθέτει τη λειτουργία οπτικής σύγκρισης του περιεχομένου που βρίσκεται στον δοκιμαστικό server με το περιεχόμενο που υπάρχει στον server παραγωγής. Η συγκεκριμένη λειτουργία λοιπόν, τράβηξε απομακρυσμένα την ίδια χρονική στιγμή σε μορφή screenshot, το περιεχόμενο που είχε η κάθε ιστοσελίδα που είχαμε καταχωρήσει, δίνοντας έτσι τη δυνατότητα ελέγχου της νέας αλλαγής και σε επίπεδο εμφάνισης, δηλαδή στο πως θα εμφανίζεται πρακτικά η συγκεκριμένη αλλαγή σε έναν χρήστη που θα επισκεφτεί τη σελίδα παραγωγής.



Εικόνα 3-33 Visual Tests

Η συγκεκριμένη λειτουργία αν εντοπίσει κάποια αλλαγή στο περιεχόμενο των 2 ιστοσελίδων τότε αυτόματα σημειώνει με κόκκινο χρώμα τα σημεία στα οποία υπάρχουν αλλαγές και έπειτα περιμένει την έγκριση του διαχειριστή του αγωγού για να δει και να εγκρίνει τις συγκεκριμένες αλλαγές ώστε να προχωρήσει η διαδικασία στο επόμενο βήμα (όπως φαίνεται και στην Εικόνα 3-32)

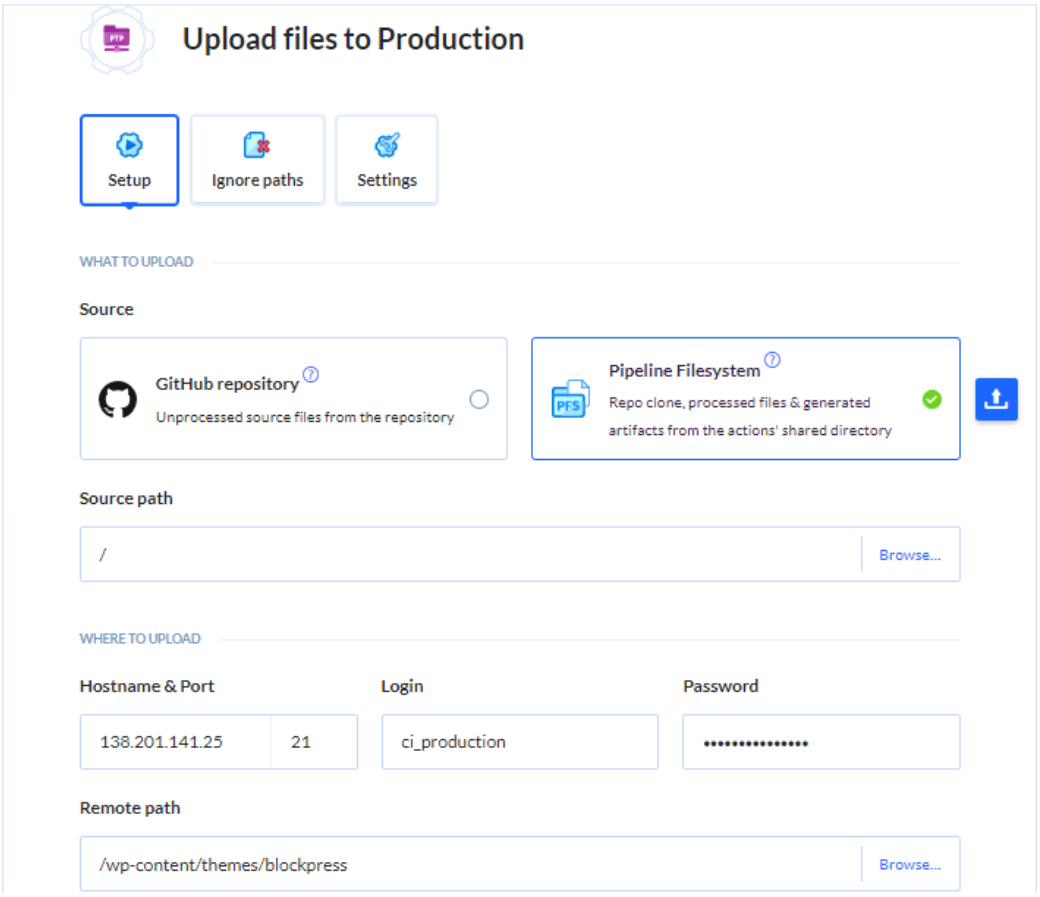


Εικόνα 3-34 Σύγκριση Test Server/Production Server μέσω Visual Testing

Τέλος, εφόσον οι αλλαγές ήταν αυτές που θέλαμε να δούμε, πατήσαμε το κουμπί "Approve" ολοκληρώνοντας έτσι τις οπτικές δοκιμές και το στάδιο του Release, οδηγώντας έτσι τη διαδικασία στο τελικό στάδιο, το οποίο ήταν αυτό του ανεβάσματος των αρχείων στον server παραγωγής μέσω FTP (Στάδιο Deploy)

3.5.8 Ανέβασμα Αρχείων στον Production Server (Στάδιο Deploy)

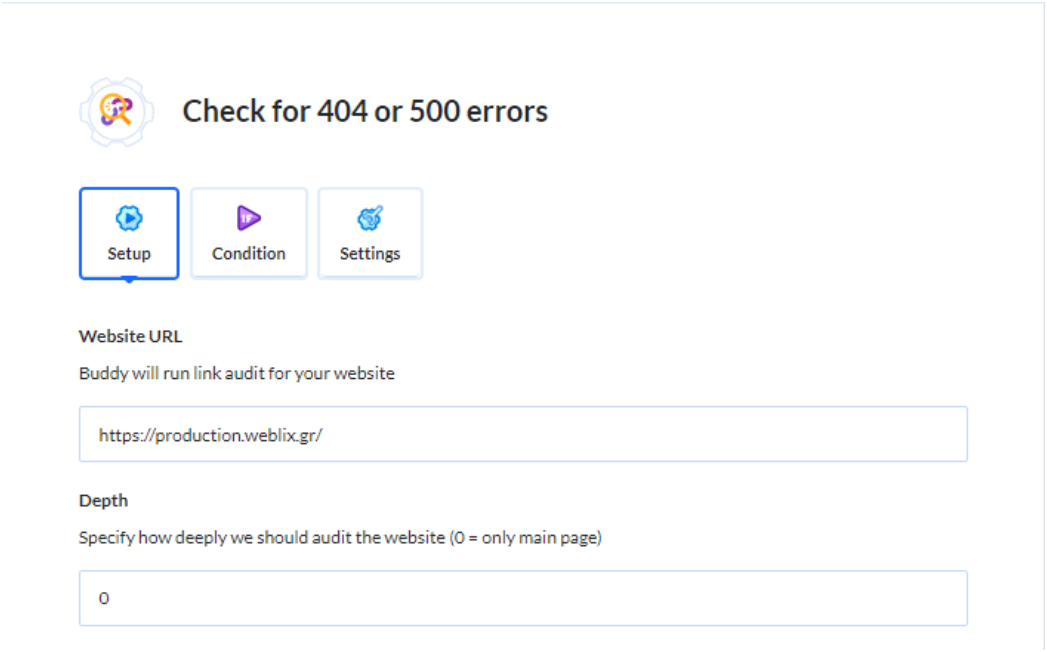
Η διασύνδεση του Buddy με τον server παραγωγής έγινε με τον ίδιο ακριβώς τρόπο που πραγματοποιήθηκε και η διασύνδεση με τον δοκιμαστικό server, δηλαδή μέσω FTP. Για την επίτευξη της σύνδεσης αυτής χρειάστηκαν η IP που φιλοξενείται η ιστοσελίδα παραγωγής καθώς επίσης και η θύρα, το username και το password του νέου λογαριασμού FTP που δημιουργήθηκε και αυτός από το περιβάλλον του server που φιλοξενείται η ιστοσελίδα παραγωγής. Τέλος, επιλέχθηκε το "Remote path", δηλαδή ο φάκελος στον οποίο τελικά θα προστεθεί η φόρμα ώστε να δημοσιευθεί στην παραγωγή ολοκληρώνοντας έτσι και το στάδιο του Deploy.



Εικόνα 3-35 Ανέβασμα Νέας Έκδοσης στον Production Server

3.5.9 Έλεγχος Σφαλμάτων 404/500

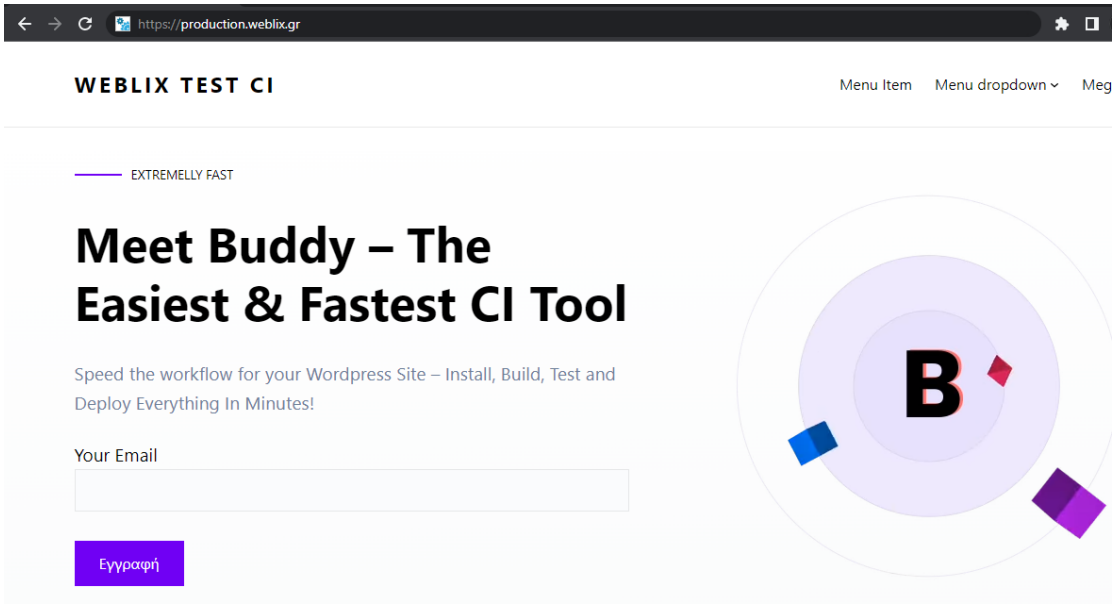
Ως δικλείδα ασφαλείας, προστέθηκε και η λειτουργία ελέγχου της ιστοσελίδας παραγωγής για τυχόν σφάλματα 404 ή 500, κυρίως για να διασφαλίσουμε ότι δεν υπήρξε κάποιο πρόβλημα στον server κατά τη διαδικασία ανεβάσματος των αρχείων.



Εικόνα 3-36 Έλεγχος για Σφάλματα 404 ή 500

3.5.10 Αποτέλεσμα Δοκιμαστικής Εφαρμογής Εργαλείου Buddy

Εφόσον λοιπόν ο αγωγός ολοκληρώθηκε περνώντας όλα τα στάδια και τις ενέργειες που χρειάζονταν, διαπιστώθηκε πως η φόρμα καταχωρήσης email είχε προστεθεί με επιτυχία ως νέα έκδοση στην ιστοσελίδα παραγωγής (**production.weblix.gr**) χωρίς σφάλματα.



Εικόνα 3-37 Αποτέλεσμα Δοκιμαστικής Εφαρμογής

4 ΣΥΜΠΕΡΑΣΜΑΤΑ

Η παρούσα Διπλωματική εργασία είχε ως σκοπό την ανάπτυξη μίας δοκιμαστικής εφαρμογής με χρήση ενός εργαλείου CI, ώστε να επιτευχθεί γρηγορότερη και ευκολότερη ανάπτυξη για κάθε νέα έκδοση μίας ιστοσελίδας σε πλατφόρμα Wordpress.

Για την επίτευξη λοιπόν του σκοπού αυτού, κρίθηκε απαραίτητη η σύγκριση 8 εργαλείων CI επιλέγοντας το εργαλείο που πληρούσε τις περισσότερες προϋποθέσεις για την συγκεκριμένη υλοποίηση. Ένα από τα πιο σημαντικά συμπεράσματα που προέκυψαν κατά τη διάρκεια της σύγκρισης αυτής, ήταν ότι δεν υπάρχει εργαλείο που με σιγουριά θα μπορούσε να χαρακτηριστεί ως το καλύτερο εργαλείο CI στην αγορά ανεξαρτήτως σκοπού και εφαρμογής. Το κάθε εργαλείο έχει τα πλεονεκτήματά του και τις διαφοροποιήσεις του σε σχέση με τα υπόλοιπα, επομένως αυτές οι διαφορές, μπορεί να είναι αναγκαίες και χρήσιμες για τους σκοπούς κάποιας άλλης υλοποίησης.

Κάθε περίπτωση λοιπόν, έχει τις δικές της συνθήκες και απαιτήσεις και για την εφαρμογή της παρούσας εργασίας, επιλέχθηκε το εργαλείο Buddy ως το πιο κατάλληλο, για το οποίο προέκυψαν και τα εξής συμπεράσματα:

- Το Buddy είναι ιδανικό για κάποιον που κάνει τα πρώτα του βήματα στον χώρο του CI καθώς διαθέτει ένα πολύ εύχρηστο και σύγχρονο περιβάλλον εργασίας που καθιστά πολύ εύκολη και ξεκάθαρη την διαδικασία δημιουργίας ενός αγωγού CI.

- Το 24/7 Live Chat, οι δωρεάν οδηγοί και τα videos που έχει η πλατφόρμα μπορούν να επιλύσουν οποιοδήποτε πρόβλημα ή απορία προκύψει, κάτι που επισπεύδει ακόμη περισσότερο την εκμάθηση του εργαλείου.

- Η έξυπνη ανίχνευση αλλαγών, η υπερσύγχρονη προσωρινή αποθήκευση δεδομένων και η δυνατότητα παράλληλων ενεργειών καθιστούν σίγουρα το Buddy, ένα από τα εργαλεία CI με την γρηγορότερη ανάπτυξη, χωρίς όμως να επιβεβαιωθεί ότι είναι το καλύτερο εργαλείο στην ταχύτητα ανάπτυξης.

- Η άριστη συμβατότητά του με την πλατφόρμα Wordpress, το καθιστούν σίγουρα ως το πιο κατάλληλο εργαλείο για εφαρμογές βασισμένες στην συγκεκριμένη πλατφόρμα ή άλλες παρόμοιες πλατφόρμες CMS.

- Οι δυνατότητες του δωρεάν πλάνου του Buddy είναι μεγαλύτερες ακόμη και από πλάνα επί πληρωμή άλλων εργαλείων CI. Ωστόσο, σε ενδεχόμενη ενσωμάτωση μίας μικρής και απλής εφαρμογής με ανάπτυξη, πολλές φορές μέσα στην ημέρα, το δωρεάν πλάνο δεν θα ήταν αρκετό. Έχοντας λοιπόν, μία εφαρμογή με απλή υλοποίηση και συχνή ανάπτυξη, θεωρητικά η επιλογή ενός δωρεάν εργαλείου CI θα ήταν η πιο σωστή και επικρατέστερη επιλογή σε σχέση με το επί πληρωμή πλάνο του εργαλείου Buddy.

Ένα ακόμη σημαντικό συμπέρασμα είναι πως το εργαλείο Buddy όπως και όλα τα εργαλεία δεν μπορούν να διεκπεραιώσουν όλα τα στάδια του CI. Κανένα δεν έχει τη

δυνατότητα δημιουργίας αποθετηρίου το οποίο είναι απαραίτητο για το στάδιο του Source Control. Για το συγκεκριμένο στάδιο τα εργαλεία CI χρειάζεται να συνεργαστούν και συνδεθούν με άλλες πλατφόρμες, με πιο διαδεδομένη και αξιόπιστη να είναι αυτή του Github, η οποία αναλύθηκε και χρησιμοποιήθηκε και στην υλοποίηση του κεφαλαίου 3.

Επιπλέον, κατά τη διάρκεια δημιουργίας του αγωγού CI με το Buddy, διαπιστώθηκε πως για να δημοσιευτούν οι αλλαγές στην ιστοσελίδα παραγωγής, τα στάδια της πρακτικής του CI δεν ήταν αρκετά, καθώς μετά το στάδιο των tests, ήταν απαραίτητο να προστεθούν ενέργειες που είτε θα ανεβάζουν απευθείας την νέα έκδοση στο περιβάλλον παραγωγής (continuous deployment), είτε θα περιμένουν έγκριση από ανθρώπινο παράγοντα (continuous delivery). Επομένως, συμπεραίνουμε πως για να αναπτυχθεί μία εφαρμογή σε περιβάλλον παραγωγής είναι απαραίτητη η χρήση είτε του Continuous Delivery είτε του Continuous Deployment.

Τέλος, όπως είδαμε και στο κεφάλαιο 2 μέσω των τεσσάρων μεγάλων εταιρειών που εφαρμόζουν CI, η αναγκαιότητα χρήσης της πρακτικής CI για μία σύγχρονη εταιρεία κρίνεται απαραίτητη καθώς μπορεί στην αρχή η ενσωμάτωση ενός νέου τμήματος να φαντάζει μία αρκετά δύσκολη και σύνθετη διαδικασία, κρίνοντας όμως πάντα σε ένα μακροπρόσθεσμο πλαίσιο, η βελτίωση ποιότητας και ταχύτητας παράδοσης λογισμικού, η βελτίωση της ομαδικότητας και της παραγωγικότητας των υπαλλήλων, καθώς και η σημαντική εξοικονόμηση πόρων είναι μόνο μερικά από τα πλεονεκτήματα του CI που θα μπορέσουν να αυξήσουν την βιωσιμότητα και την κερδοφορία μίας εταιρείας, οδηγώντας την σε ένα επίπεδο που θα την εξελίξει και θα την ξεχωρίσει σε σχέση με τον ανταγωνισμό.

Περιορισμοί

Κατά την έναρξη της διπλωματικής εργασίας, η υλοποίηση είχε προγραμματιστεί να εφαρμοστεί σε διαδικτυακές εφαρμογές εταιρείας η οποία δυστυχώς δεν πραγματοποιήθηκε λόγω έξαρσης της νόσου Covid-19. Επομένως, αυτό είχε ως συνέπεια την αδυναμία ποιοτικής και ποσοτικής αξιολόγησης του συστήματος σε πραγματικές συνθήκες, δηλαδή, σε μία εν λειτουργία ιστοσελίδα που έχει πραγματικούς χρήστες και εφαρμόζει καθημερινή ανάπτυξη κώδικα, ώστε να επιβεβαιωθεί ότι το σύστημα εξακολουθεί να λειτουργεί σωστά και χωρίς προβλήματα.

Επιπλέον, επειδή ο παραπάνω περιορισμός δεν επέτρεψε την δοκιμή του εργαλείου Buddy σε μία εν λειτουργία ιστοσελίδα με πολλές αναπτύξεις μέσα στη μέρα, δεν κρίθηκε αναγκαία και η αναβάθμιση του λογαριασμού Buddy σε πλάνο μηνιαίας συνδρομής. Σε μία τέτοια ιστοσελίδα λοιπόν, με περισσότερες απαιτήσεις, θα δινόταν η ευκαιρία στο να γίνει στην πράξη η σύγκριση του επί πληρωμή εργαλείου Buddy με ένα δωρεάν εργαλείο CI, ώστε να διαπιστωθεί εάν θα παρέμενε και σε αυτή την περίπτωση ως η πρώτη επιλογή.

Μελλοντικά Βήματα / Προεκτάσεις

Λόγω των περιορισμών που προαναφέρθηκαν, είναι σημαντικό να γίνει μία ποσοτική και ποιοτική αξιολόγηση του συστήματος σε πραγματικές συνθήκες, δηλαδή σε μία εν λειτουργία ιστοσελίδα η οποία βασίζεται αντίστοιχα σε πλατφόρμα Wordpress και αναπτύσσει κώδικα πολλές φορές μέσα στην ημέρα.

Έπειτα, έχοντας μία ιστοσελίδα με καθημερινή και συχνή ανάπτυξη θα κριθεί αναγκαία η αναβάθμιση του λογαριασμού Buddy σε πλάνο μηνιαίας συνδρομής. Στην περίπτωση αυτή λοιπόν, μία περαιτέρω προέκταση θα μπορούσε να είναι η δημιουργία της ίδιας ακριβώς υλοποίησης μέσω ενός δωρεάν εργαλείου CI (πχ Jenkins) όπου στη συνέχεια καταγράφοντας δεδομένα για το κάθε εργαλείο, θα προέκυπτε το ποσό που εξοικονομείται μέσα σε ένα μήνα από το Buddy και το ποσό που εξοικονομείται αντίστοιχα από το δωρεάν εργαλείο. Σκοπός θα είναι να αποδειχθεί αν το Buddy ακόμη και σε αυτή την περίπτωση θα μπορούσε να επιλεχθεί ως κατάλληλο εργαλείο ή όχι, μιας και στη θεωρία φαίνεται πως είναι ένα από τα αδύναμα σημεία του και είναι ένας από τους λόγους που δεν επιλέγεται, παρόλο που στην πράξη κάτι τέτοιο δεν έχει ακόμα αποδειχθεί.

Βιβλιογραφία – Αναφορές - Διαδικτυακές Πηγές

[1] A. Joshi, M. Khosravy, και N. Gupta, Επιμ., *Machine Learning for Predictive Analysis: Proceedings of ICTIS 2020*, τ. 141. Singapore: Springer Singapore, 2021. doi: 10.1007/978-981-15-7106-0.

[2] S.-T. Lai και F.-Y. Leu, 'Applying Continuous Integration for Reducing Web Applications Development Risks', στο *2015 10th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA)*, Krakow, Poland, Νοεμβρίου 2015, σσ. 386–391. doi: 10.1109/BWCCA.2015.54.

[3] K. Sureshchandra και J. Shrinivasavadhani, 'Moving from Waterfall to Agile', στο *Agile 2008 Conference*, Toronto, ON, Canada, 2008, σσ. 97–101. doi: 10.1109/Agile.2008.49.

[4] Akarsha AP και Dr Ambedkar institute of technology, 'Continuous Integration Research Based on Docker', *IJERT*, τ. V5, τχ. 05, σ. IJERTV5IS050443, Μαΐου 2016, doi: 10.17577/IJERTV5IS050443.

[5] L. Riungu-Kalliosaari, S. Mäkinen, L. E. Lwakatare, J. Tiihonen, και T. Männistö, 'DevOps Adoption Benefits and Challenges in Practice: A Case Study', στο *Product-Focused Software Process Improvement*, τ. 10027, P. Abrahamsson, A. Jedlitschka, A. Nguyen Duc, M. Felderer, S. Amasaki, και T. Mikkonen, Επιμ. Cham: Springer International Publishing, 2016, σσ. 590–597. doi: 10.1007/978-3-319-49094-6_44.

[6] A. Hemon, B. Lyonnet, F. Rowe, και B. Fitzgerald, 'From Agile to DevOps: Smart Skills and Collaborations', *Inf Syst Front*, τ. 22, τχ. 4, σσ. 927–945, Αυγούστου 2020, doi: 10.1007/s10796-019-09905-1.

[7] I. Karamitsos, S. Albarhami, και C. Apostolopoulos, 'Applying DevOps Practices of Continuous Automation for Machine Learning', *Information*, τ. 11, τχ. 7, σ. 363, Ιουλίου 2020, doi: 10.3390/info11070363.

[8] J. Rossberg, 'Agile Practices in Azure DevOps and TFS', στο *Agile Project Management with Azure DevOps*, Berkeley, CA: Apress, 2019, σσ. 197–228. doi: 10.1007/978-1-4842-4483-8_6.

[9] C. Ebert, G. Gallardo, J. Hernantes, και N. Serrano, 'DevOps', *IEEE Softw.*, τ. 33, τχ. 3, σσ. 94–100, Μαΐου 2016, doi: 10.1109/MS.2016.68.

[10] L. Chen, 'Microservices: Architecting for Continuous Delivery and DevOps', στο *2018 IEEE International Conference on Software Architecture (ICSA)*, Seattle, WA, Απριλίου 2018, σσ. 39–397. doi: 10.1109/ICSA.2018.00013.

[11] B. B. N. de França, P. S. M. dos Santos, και S. Matalonga, 'Towards a Theory on Architecting for Continuous Deployment'. arXiv, 21 Αύγουστος 2021. Ημερομηνία πρόσβασης: 2 Ιούνιος 2022. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://arxiv.org/abs/2108.09571

[12] B. S. Farroha και D. L. Farroha, 'A Framework for Managing Mission Needs, Compliance, and Trust in the DevOps Environment', στο *2014 IEEE Military Communications Conference*, Baltimore, MD, USA, Οκτωβρίου 2014, σσ. 288–293. doi: 10.1109/MILCOM.2014.54.

[13] S. Uzunbayir και K. Kurtel, 'A Review of Source Code Management Tools for Continuous Software Development', στο *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, Sarajevo, Σεπτεμβρίου 2018, σσ. 414–419. doi: 10.1109/UBMK.2018.8566644.

[14] M. Krief, *Learning DevOps: the complete guide to accelerate collaboration with Jenkins, Kubernetes, Terraform and Azure DevOps*. 2019. Ημερομηνία πρόσβασης: 17 Δεκέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=2284100

[15] A. Agarwal, S. Gupta, και T. Choudhury, 'Continuous and Integrated Software Development using DevOps', στο *2018 International Conference on Advances in Computing and Communication Engineering (ICACCE)*, Paris, Ιουνίου 2018, σσ. 290–293. doi: 10.1109/ICACCE.2018.8458052.

[16] A. Capizzi, S. Distefano, και M. Mazzara, 'From DevOps to DevDataOps: Data Management in DevOps processes'. arXiv, 7 Οκτώβριος 2019. Ημερομηνία πρόσβασης: 2 Ιούνιος 2022. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://arxiv.org/abs/1910.03066

[17] M. Rodriguez, L. J. P. de Araújo, και M. Mazzara, 'Good practices for the adoption of DataOps in the software industry', *J. Phys.: Conf. Ser.*, τ. 1694, τχ. 1, σ. 012032, Δεκεμβρίου 2020, doi: 10.1088/1742-6596/1694/1/012032.

[18] P. Rodríguez *κ.ά.*, 'Continuous deployment of software intensive products and services: A systematic mapping study', *Journal of Systems and Software*, τ. 123, σσ. 263–291, Ιανουαρίου 2017, doi: 10.1016/j.jss.2015.12.015.

[19] L. Chen, 'Continuous Delivery: Huge Benefits, but Challenges Too', *IEEE Softw.*, τ. 32, τχ. 2, σσ. 50–54, Μαρτίου 2015, doi: 10.1109/MS.2015.27.

[20] S. Kim, S. Park, J. Yun, και Y. Lee, 'Automated Continuous Integration of Component-Based Software: An Industrial Experience', στο *2008 23rd IEEE/ACM International Conference on Automated Software Engineering*, L'Aquila, Italy, Σεπτεμβρίου 2008, σσ. 423–426. doi: 10.1109/ASE.2008.64.

[21] F. Zampetti, S. Scalabrino, R. Oliveto, G. Canfora, και M. Di Penta, 'How Open Source Projects Use Static Code Analysis Tools in Continuous Integration Pipelines', στο *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, Buenos Aires, Argentina, Μαΐου 2017, σσ. 334–344. doi: 10.1109/MSR.2017.2.

[22] M. Grossmann και C. Ioannidis, 'Continuous Integration of Applications for ONOS', στο *2019 IEEE Conference on Network Softwarization (NetSoft)*, Paris, France, Ιουνίου 2019, σσ. 213–217. doi: 10.1109/NETSOFT.2019.8806696.

[23] K. Priyadarsini, E. F. I. Raj, A. Y. Begum, και V. Shanmugasundaram, 'Comparing DevOps procedures from the context of a systems engineer', *Materials Today: Proceedings*, σ. S2214785320373491, Οκτωβρίου 2020, doi: 10.1016/j.matpr.2020.09.624.

[24] J. Humble, 'Continuous delivery sounds great, but will it work here?', *Commun. ACM*, τ. 61, τχ. 4, σσ. 34–39, Μαρτίου 2018, doi: 10.1145/3173553.

[25] S. Mathew, 'Overview of Amazon Web Services', σ. 30, 2014.

[26] J. Bae, C. Kim, και J. Kim, 'Automated deployment of SmartX IoT-cloud services based on continuous integration', στο *2016 International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju, Οκτωβρίου 2016, σσ. 1076–1081. doi: 10.1109/ICTC.2016.7763372.

[27] T. Yu και T. Wang, 'A Study of Regression Test Selection in Continuous Integration Environments', στο *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*, Memphis, TN, Οκτωβρίου 2018, σσ. 135–143. doi: 10.1109/ISSRE.2018.00024.

[28] N. Marini-Wear, 'QUALITATIVE CASE STUDY: SOFTWARE SECURITY IN DEVOPS', σ. 24.

[29] A. Balalaie, A. Heydarnoori, και P. Jamshidi, 'Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture', *IEEE Softw.*, τ. 33, τχ. 3, σσ. 42–52, Μαΐου 2016, doi: 10.1109/MS.2016.64.

[30] T. Savor, M. Douglas, M. Gentili, L. Williams, K. Beck, και M. Stumm, 'Continuous deployment at Facebook and OANDA', στο *Proceedings of the 38th International Conference on Software Engineering Companion*, Austin Texas, Μαΐου 2016, σσ. 21–30. doi: 10.1145/2889160.2889223.

[31] D. G. Feitelson, E. Frachtenberg, και K. L. Beck, 'Development and Deployment at Facebook', *IEEE Internet Comput.*, τ. 17, τχ. 4, σσ. 8–17, Ιουλίου 2013, doi: 10.1109/MIC.2013.25.

[32] A. A. U. Rahman, E. Helms, L. Williams, και C. Parnin, 'Synthesizing Continuous Deployment Practices Used in Software Development', στο *2015 Agile Conference*, National Harbor, MD, USA, Αυγούστου 2015, σσ. 1–10. doi: 10.1109/Agile.2015.12.

[33] C. Rossi, E. Shibley, S. Su, K. Beck, T. Savor, και M. Stumm, 'Continuous deployment of mobile software at facebook (showcase)', στο *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, Seattle WA USA, Νοεμβρίου 2016, σσ. 12–23. doi: 10.1145/2950290.2994157.

[34] ‘Full-scale Software Engineering / Current Trends in Release Engineering’, σ. 72.

[35] S. Denning, ‘New lessons for leaders about continuous innovation’, *Strategy & Leadership*, τ. 43, τχ. 1, σσ. 11–15, Ιανουαρίου 2015, doi: 10.1108/SL-11-2014-0083.

[36] D. Polkhovskiy, ‘Comparison between Continuous Integration tools’, σ. 59.

[37] T. Ellis και M. Graphics, ‘Increased Efficiency with Questa® VRM and Jenkins Continuous Integration’, σ. 8.

[38] Z. Sampedro, A. Holt, και T. Hauser, ‘Continuous Integration and Delivery for HPC: Using Singularity and Jenkins’, στο *Proceedings of the Practice and Experience on Advanced Research Computing*, Pittsburgh PA USA, Ιουλίου 2018, σσ. 1–6. doi: 10.1145/3219104.3219147.

[39] *Jenkins: The Definitive Guide: Continuous Integration for the Masses*.

[40] K. Gallaba και S. McIntosh, ‘Use and Misuse of Continuous Integration Features: An Empirical Study of Projects That (Mis)Use Travis CI’, *IIEEE Trans. Software Eng.*, τ. 46, τχ. 1, σσ. 33–50, Ιανουαρίου 2020, doi: 10.1109/TSE.2018.2838131.

[41] S. M. Mohammad, ‘Continuous Integration and automation’, τ. 4, τχ. 3, σ. 8, 2016.

[42] M. Hüttermann, *DevOps for developers*. New York: Apress : Distributed to the book trade worldwide by Springer Science+Business Media New York, 2012.

[43] M. Meyer, ‘Continuous Integration and Its Tools’, *IEEE Softw.*, τ. 31, τχ. 3, σσ. 14–16, Μαΐου 2014, doi: 10.1109/MS.2014.58.

[44] M. Laaziri, K. Benmoussa, S. Khoulji, K. Mohamed Larbi, και A. E. Yamami, ‘A comparative study of laravel and symfony PHP frameworks’, *IJECE*, τ. 9, τχ. 1, σ. 704, Φεβρουαρίου 2019, doi: 10.11591/ijece.v9i1.pp704-712.

[45] G. B. Ghantous και A. Q. Gill, ‘DevOps Reference Architecture for Multi-cloud IOT Applications’, στο *2018 IEEE 20th Conference on Business Informatics (CBI)*, Vienna, Ιουλίου 2018, σσ. 158–167. doi: 10.1109/CBI.2018.00026.

[46] M. S. Mahalingam και T. Shi, *Learning continuous integration with teamcity: master the principles and practices behind continuous integration by setting it up for different technology stacks using teamcity*. Birmingham, England: Packt Publishing Ltd, 2014.

[47] H. M. Yuksel, E. Tuzun, E. Gelirli, E. Biyikli, και B. Baykal, ‘Using continuous integration and automated test techniques for a robust C4ISR system’, στο *2009 24th International Symposium on Computer and Information Sciences*, Guzelyurt, Cyprus, Σεπτεμβρίου 2009, σσ. 743–748. doi: 10.1109/ISCIS.2009.5291914.

[48] V. Sochat, ‘Containershare: Open Source Registry to build, test, deploy with CircleCI’, *JOSS*, τ. 3, τχ. 28, σ. 878, Αυγούστου 2018, doi: 10.21105/joss.00878.

[49] E. E. Romero *κ.ά.*, ‘Integration of DevOps Practices on a Noise Monitor System

with CircleCI and Terraform', *ACM Trans. Manage. Inf. Syst.*, τ. 13, τχ. 4, σσ. 1–24, Δεκεμβρίου 2022, doi: 10.1145/3505228.

[50] P. P. Than και M. P. Phyu, 'Continuous integration for Laravel applications with GitLab', στο *Proceedings of the International Conference on Advanced Information Science and System*, Singapore Singapore, Νοεμβρίου 2019, σσ. 1–6. doi: 10.1145/3373477.3373479.

[51] C. Singh, N. S. Gaba, M. Kaur, και B. Kaur, 'Comparison of Different CI/CD Tools Integrated with Cloud Platform', στο *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India, Ιανουαρίου 2019, σσ. 7–12. doi: 10.1109/CONFLUENCE.2019.8776985.

[52] S. T. Datko, S. Lueders, και H. Short, 'Static Code Analysis with Gitlab-CI', σ. 32.

[53] M. Asfour, 'Development and Maintenance of Continuous Integration pipelines for Mobile Applications', σ. 120.

[54] Hoang-Trieu Tong, 'Tong, H. T. Robust Development with Continuous Integration in TIDOWA Robuste Entwicklung mit Kontinuierlicher Integration in TIDOWA.' [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://www.martin-schreiber.info/data/student_projects/2021_BA_hoang_trieu_tong.pdf