ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη Λογισμικού και CI/CD υποδομές: Συνεχής Ενσωμάτωση/Συνεχής Παράδοση & Συνεχής Ανάπτυξη

Του φοιτητή:
Κωνσταντίνου Σαρακενίδη
Αρ. Μητρώου: it144253

Επιβλέπων
Ονοματεπώνυμο: Μιχάλης Σαλαμπάσης
Βαθμίδα: Καθηγητής

Ημερομηνία 14/4/2024

Τίτλος Δ.Ε.Ανάπτυξη Λογισμικού και CI/CD υποδομές: Συνεχής Ενσωμάτωση/Συνεχής Παράδοση & Συνεχής Ανάπτυξη
Κωδικός Δ.Ε.: 22223
Ονοματεπώνυμο φοιτητή: Κωνσταντίνος Σαρακενίδης
Ονοματεπώνυμο εισηγητή: Μιχάλης Σαλαμπάσης
Ημερομηνία ανάληψης Δ.Ε.: 14/04/2022
Ημερομηνία περάτωσης Δ.Ε.: 14/04/2024

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Πρόλογος

Σε αυτήν την εργασία επέλεξα να ερευνήσω τη συνεχή ενσωμάτωση και ανάπτυξη στο πλαίσιο της ανάπτυξης λογισμικού διότι εντοπίζω μεγάλη σημασία στην ανάπτυξη προηγμένων διαδικασιών ανάπτυξης. Καθώς η ταχεία παράδοση του λογισμικού και η διασφάλιση της ποιότητάς του είναι κρίσιμες για την επιτυχία των έργων, θεώρησα ότι η εμβάθυνση σε αυτό τον τομέα θα μου δώσει απαραίτητες γνώσεις και δεξιότητες για τη μελλοντική μου επαγγελματική ανάπτυξη. Επιπλέον, επιλέγοντας να αναπτύξω την υποδομή CI/CD σε ένα περιβάλλον Kubernetes στο Azure, επεδίωξα να εμβαθύνω σε επίκαιρες τεχνολογίες και να αποκτήσω πρακτική εμπειρία σ' ένα μεγάλο φάσμα εργαλείων και υπηρεσιών. Έτσι, η διπλωματική αυτή έρευνα αποτελεί ένα σημαντικό βήμα για την επαγγελματική μου ανάπτυξη και την κατανόηση των σύγχρονων πρακτικών ανάπτυξης λογισμικού.

Περίληψη

Η γρήγορη εξέλιξη των μεθοδολογιών ανάπτυξης λογισμικού απαιτεί την υιοθέτηση αποτελεσματικών διεργασιών και υποδομών για τη διασφάλιση της ευελιξίας, της αξιοπιστίας και της κλιμακωσιμότητας στην παράδοση λογισμικού. Η Συνεχής Ενσωμάτωση (CI), η Συνεχής Παράδοση (CD) και η Συνεχής Ανάπτυξη (CD) έχουν εμφανιστεί ως κρίσιμες πρακτικές στη σύγχρονη μηχανική λογισμικού, επιτρέποντας σε ομάδες να διαμορφώσουν τις ροές ανάπτυξης, να βελτιώσουν τη συνεργασία και να επιταχύνουν τον κύκλο κυκλοφορίας.

Η παρούσα εργασία εξετάζει τις θεμελιώδεις αρχές, τις στρατηγικές υλοποίησης και τις πρακτικές συνέπειες των σωληνώσεων (pipelines) CI/CD εντός των συστημάτων Kubernetes, μιας κορυφαίας πλατφόρμας ενορχήστρωσης. Μέσα από μια σφαιρική έρευνα της βιβλιογραφίας, περιστατικών μελέτης και εμπειρικής ανάλυσης, η εργασία εξετάζει την αμοιβαία σχέση μεταξύ του Kubernetes και των μεθοδολογιών CI/CD, ξεκαθαρίζοντας πώς το Kubernetes διευκολύνει τον αυτοματισμό, την κλιμακωσιμότητα και την ανθεκτικότητα που απαιτούνται για αποτελεσματικές ροές εργασιών CI/CD.

Τα κύρια θέματα που αναλύονται περιλαμβάνουν τον σχεδιασμό και την αρχιτεκτονική των σωληνώσεων (pipelines) CI/CD, την ενσωμάτωση με συστήματα ελέγχου εκδόσεων, τεχνικές διαχείρισης, ενορχήστρωση διαδικασιών δοκιμής και παραγωγής, καθώς και βέλτιστες πρακτικές για την παρακολούθηση και τη βελτιστοποίηση. Επιπλέον, η διατριβή ερευνά τον αντίκτυπο της υιοθέτησης CI/CD στην ποιότητα του λογισμικού, τη δυναμική της ομάδας και την οργανωτική κουλτούρα, παρέχοντας εισηγήσεις για το μετασχηματιστικό δυναμικό αυτών των μεθοδολογιών.

Η παρούσα εργασία προσφέρει πρακτικές συστάσεις για ομάδες ανάπτυξης λογισμικού που επιθυμούν να εφαρμόσουν ή να βελτιώσουν τις πρακτικές CI/CD. Συνδυάζοντας θεωρητικά πλαίσια, συμβάλλει στον αυξανόμενο όγκο γνώσης γύρω από το DevOps, τις μεθοδολογίες μηχανικής λογισμικού και τη διαχείριση υποδομών, ανοίγοντας το δρόμο για πιο αποτελεσματικές, αξιόπιστες και κλιμακώσιμες ροές παράδοσης λογισμικού στα σύγχρονα συστήματα ανάπτυξης λογισμικού.

Λέξεις κλειδιά: εξέλιξη, διασφάλιση ευελιξίας, αξιοπιστία, ανθεκτικότητα, Kubernetes, βελτιστοποίηση

«Software Development and CI/CD Infrastructure: Continuous Integration/Continuous Delivery & Continuous Deployment»

«Konstantinos Sarakenidis»

Abstract

The rapid evolution of software development methodologies has necessitated the adoption of efficient processes and infrastructure to ensure agility, reliability, and scalability in software delivery. Continuous Integration (CI), Continuous Delivery (CD), and Continuous Deployment (CD) have emerged as pivotal practices in modern software engineering, enabling teams to streamline development workflows, enhance collaboration, and accelerate the release cycle.

This thesis explores the fundamental principles, implementation strategies, and practical implications of CI/CD pipelines within Kubernetes clusters, a leading container orchestration platform. Through a comprehensive review of literature, case studies, and empirical analysis, the thesis examines the symbiotic relationship between Kubernetes and CI/CD methodologies, elucidating how Kubernetes facilitates the automation, scalability, and resilience required for efficient CI/CD workflows.

Key topics addressed include the design and architecture of CI/CD pipelines on Kubernetes, integration with version control systems, containerization techniques, orchestration of testing and deployment processes, and best practices for monitoring and optimization. Additionally, the thesis investigates the impact of CI/CD adoption on software quality, team dynamics, and organizational culture, providing insights into the transformative potential of these methodologies.

This thesis offers practical recommendations for software development teams seeking to implement or enhance CI/CD practices within Kubernetes environments. By synthesizing theoretical frameworks, it contributes to the growing body of knowledge surrounding DevOps, software engineering methodologies, and infrastructure management, paving the way for more efficient, dependable, and scalable software delivery pipelines in contemporary software development ecosystems.

Key words: agility, reliability, scalability, Kubernetes, automation, optimization

Ευχαριστίες

Θέλω να εκφράσω τις ευχαριστίες μου σε όλους εσάς που με στηρίξατε και με βοηθήσατε κατά τη διάρκεια της πορείας μου.

Πρώτα απ' όλα, θέλω να ευχαριστήσω την οικογένειά μου για την τεράστια στήριξη, την κατανόηση και την αγάπη που μου προσέφεραν καθ' όλη τη διάρκεια αυτής της πορείας.

Στους φίλους μου, που ήταν δίπλα μου, θέλω να πω ένα μεγάλο ευχαριστώ. Ήταν παραλαβή χαράς, ενθάρρυνσης και θετικής ενέργειας καθ' όλη τη διάρκεια αυτής της πορείας. Η υποστήριξή σας με βοήθησε να ξεπεράσω τις δυσκολίες και να επιτύχω τους στόχους μου.

Τέλος, αλλά εξίσου σημαντικό, θέλω να ευχαριστήσω τον διδάσκοντά μου για την εμπιστοσύνη, την καθοδήγηση του κατά τη διάρκεια της εκπόνησης αυτής της εργασίας.

Πίνακας Περιεχομένων

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
CI/CD	Continuous Integration & Continuous Deployment
DevOps	Developer & Operations
AKS	Azure Kubernetes Service
ΔΙ.ΠΑ.Ε.	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
Τ.Ν.	Τεχνητή Νοημοσύνη

Κεφάλαιο 1ο: Εισαγωγή

Η ανάπτυξη λογισμικού αποτελεί ένα κρίσιμο κομμάτι της σύγχρονης τεχνολογικής προόδου, διαμορφώνοντας τον τρόπο με τον οποίο αλληλοεπιδρούμε με ψηφιακά συστήματα και υπηρεσίες. Μέσα σε αυτό το πλαίσιο, η δημιουργία αξιόπιστων υποδομών Συνεχούς Ενσωμάτωσης/Συνεχούς Ανάπτυξης (CI/CD) έχει εμφανιστεί ως μια καθοριστική πρακτική. Αυτή η υποδομή επιταχύνει τη διαδικασία ανάπτυξης, εξασφαλίζοντας γρήγορη, αξιόπιστη και αποτελεσματική παράδοση των προϊόντων λογισμικού.

Η παρούσα εργασία εξετάζει την ενσωμάτωση της ανάπτυξης λογισμικού και των υποδομών CI/CD, επικεντρώνοντας στις πρακτικές Συνεχούς Ενσωμάτωσης/Συνεχούς Παράδοσης και Συνεχούς Ανάπτυξης. Σκοπός της είναι να ερευνήσει τη σημασία και την υλοποίηση του CI/CD εντός των διαδικασιών ανάπτυξης λογισμικού, ειδικότερα στο πλαίσιο των σύγχρονων περιβαλλόντων που χρησιμοποιούν υπηρεσίες στο νέφος (cloud).

1.1 Στόχοι

Οι στόχοι αυτής της εργασίας περιλαμβάνουν:

- Εξέταση των βασικών αρχών και των οφελών των πρακτικών CI/CD.
- Ανάπτυξη υποδομής CI/CD εντός ενός περιβάλλοντος Kubernetes στην υπηρεσία Azure Kubernetes Service (AKS).
- Αξιολόγηση της απόδοσης και της αποτελεσματικότητας της υποδομής CI/CD.

1.2 Δομή

Η ενότητα αυτή προσδιορίζει τα κεφάλαια που θα ακολουθηθούν και αναλυθούν στην συνέχεια της εργασίας. Παρακάτω, (κεφάλαιο 2) αναφερόμαστε στις σύγχρονες τεχνολογίες, αναλύουμε τις μικρουπηρεσίες (microservices), το Docker και το Kubernetes, προσφέροντας ένα θεωρητικό υπόβαθρο για την κατανόησή τους. Ακολουθεί (κεφάλαιο 3) εξετάζουμε τις αρχές της συνεχούς ενσωμάτωσης και παράδοσης (CI/CD), περιγράφοντας πώς αυτές οι αρχές εφαρμόζονται στο πλαίσιο της ανάπτυξης λογισμικού. Έπειτα, (κεφάλαιο 4) αναλύουμε λεπτομερώς τη ροή εργασιών που ακολουθείται σε ένα περιβάλλον CI/CD. Αναφέρουμε τα στάδια ενός pipeline ανάπτυξης εφαρμογής, από τον πηγαίο κώδικα έως την ανάπτυξη της εικόνας (image) και την τελική κυκλοφορία. Ακόμη, (κεφάλαιο 5) αξιολογούμε τα πλεονεκτήματα και τις προκλήσεις που προκύπτουν από την διαδικασία του CI/CD στο πλαίσιο του παρόντος project. Αναλύουμε επίσης τα πιθανά προβλήματα και προτείνουμε λύσεις για αυτά. Τέλος, (κεφάλαιο 6) συνοψίζουμε τα βασικά συμπεράσματα και προτείνουμε πιθανές βελτιώσεις ή μελλοντικές επεκτάσεις, ακολουθούμενα από τη βιβλιογραφία.

Κεφάλαιο 2ο: Θεωρητική Ανάλυση Σύγχρονων Τεχνολογιών

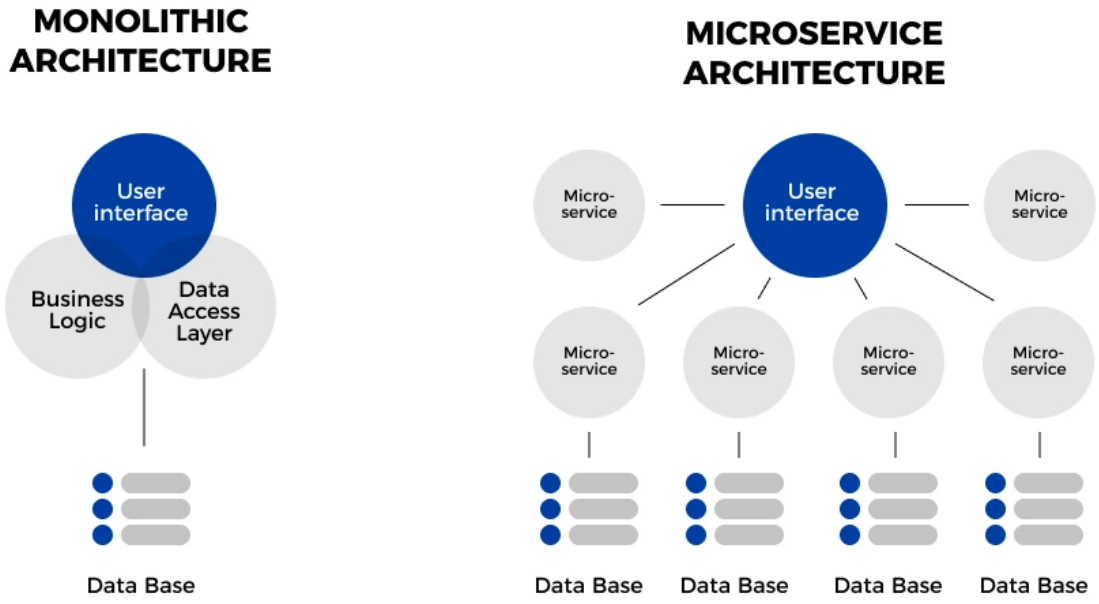
2.1 Εισαγωγή

Πριν ξεκινήσουμε την ανάλυση της συνεχούς ενσωμάτωσης και συνεχούς ανάπτυξης (CI/CD), θα αναφερθούμε στο Kubernetes, με το οποίο έχουν στηθεί και συντονιστεί τα συστήματα της εργασίας. Θα χρειαστεί αρχικά να κατανοήσουμε την τεχνολογία που χρησιμοποιείται σε αυτόν τον σύγχρονο τομέα. Στο κεφάλαιο αυτό θα παρουσιαστούν οι έννοιες των micro-services, των containers, και της οργάνωσης με το Kubernetes.

2.2 Microservices

Η αρχιτεκτονική των microservices είναι ένας τρόπος σχεδίασης και ανάπτυξης λογισμικού όπου οι εφαρμογές αναπτύσσονται ως ένα σύνολο ανεξάρτητων, απομονωμένων μικρών υπηρεσιών. Προσεγγίζεται κυρίως από μεγάλες πολύπλοκες εφαρμογές, όπου η εφαρμογή διασπάται σε μικρότερα τμήματα, γνωστά ως microservices (μικρουπηρεσίες). Κάθε microservice διαθέτει μια διεπαφή, η οποία είναι ο τρόπος με τον οποίο τα microservices επικοινωνούν μεταξύ τους [1].

Παρακάτω απεικονίζεται η διαφορά μεταξύ των αρχιτεκτονικών Monolithic και Microservice:



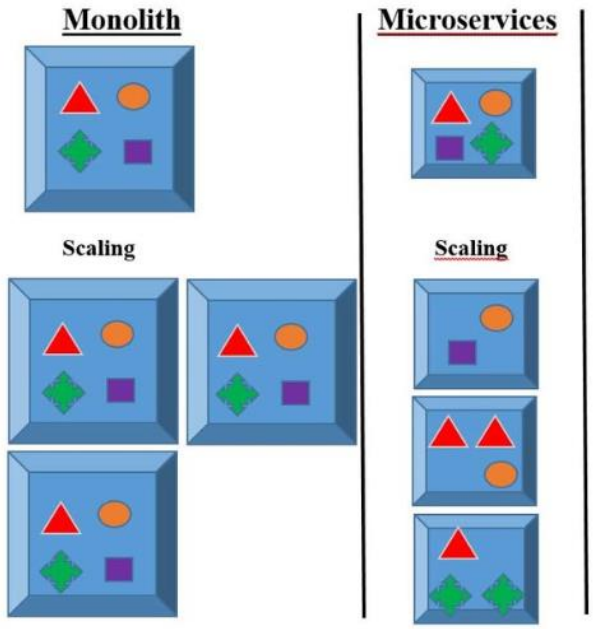
Σχήμα 2. 1: Monolithic vs Microservice αρχιτεκτονικής

Στη δεξιά πλευρά του παραπάνω σχήματος 2.1, στα microservices υπάρχει διαίρεση της εφαρμογής σε μικρότερα μέρη, το οποίο καθιστά ευκολότερη τη διαχείριση, τη συντήρηση και την επέκταση του λογισμικού. Ένα απ' τα βασικά πλεονεκτήματα του microservice είναι ότι μπορεί να αναπτυχθεί, να ενημερωθεί και να αντικατασταθεί ανεξάρτητα χωρίς να επηρεάζει τις υπόλοιπες υπηρεσίες.

Αντίθετα, στην παραδοσιακή μονολιθική αρχιτεκτονική, ολόκληρη η εφαρμογή αναπτύσσεται και αναβαθμίζεται ως μόνο ένα σύνολο, το οποίο προκαλεί δυσκολίες ανάπτυξης και συντήρησης της εφαρμογής καθώς μεγαλώνει.

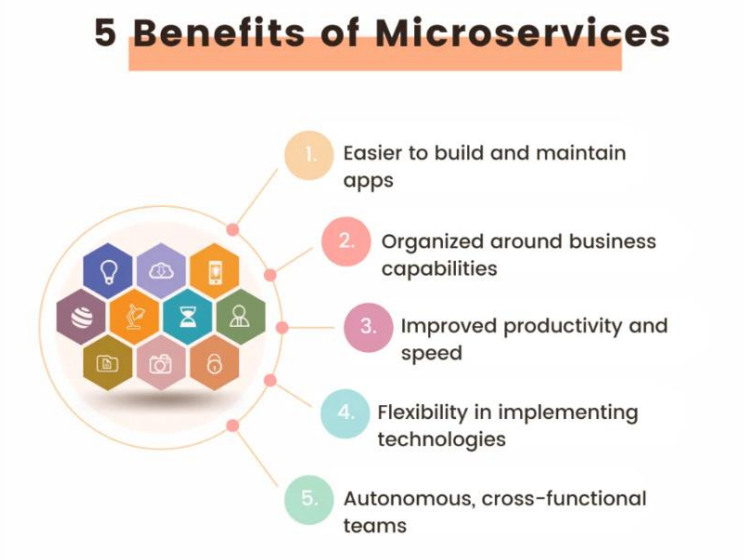
Στη περίπτωση της μονολιθικής αρχιτεκτονικής, η κλιμάκωση (scaling) γίνεται συνήθως κατακόρυφα (vertical), με αύξηση των μηχανών, το οποίο μπορεί να οδηγήσει σε αυξημένο κόστος και υπερβολική χρήση πόρων, καθώς κάθε αναβάθμιση αφορά ολόκληρη την εφαρμογή.

Αντίθετα, στην αρχιτεκτονική Microservices, η κλιμάκωση μπορεί να είναι πιο ευέλικτη και αποτελεσματική, καθώς κάθε microservice μπορεί να κλιμακωθεί ανεξάρτητα ανάλογα με τις ανάγκες του. Αυτό μπορεί να οδηγήσει σε καλύτερη χρήση των πόρων και πιο οικονομική λύση. Ωστόσο, η διαχείριση πολλαπλών microservices μπορεί να είναι πιο περίπλοκη και απαιτητική.



Σχήμα 2. 2: Κλιμάκωση μιας μονολιθικής εφαρμογής vs κλιμάκωση μιας microservice εφαρμογή.

Η αρχιτεκτονική Microservices παρέχει μεγαλύτερη ευελιξία και αποδοτικότητα στην κλιμάκωση [2], αλλά απαιτεί περισσότερη προσοχή και διαχείριση λόγω της πολυπλοκότητας της.



Σχήμα 2. 3: Οφέλη των Microservices

- **Ευκολότερη ανάπτυξη και συντήρηση εφαρμογών (Easier to build and maintain apps):** Τα microservices διασπάνε τις εφαρμογές σε μικρότερα, διαχειριστικά modules, κάνοντας τις ευκολότερες στη δημιουργία, ανάπτυξη και διαχείριση. Κάθε microservice μπορεί να χρησιμοποιεί διαφορετικές γλώσσες και περιβάλλοντα.

- **Οργανωμένες γύρω από τις επιχειρηματικές δυνατότητες (Organized around business)**: Τα microservices επικεντρώνονται στην ανάπτυξη λειτουργικότητας επιχείρησης αντί για τεχνολογία, επιτρέποντας στις ομάδες να είναι πιο δημιουργικές και παραγωγικές. Βοηθούν στη δημιουργία προϊόντων και μπορούν να επαναχρησιμοποιηθούν σε πολλαπλά περιβάλλοντα.

- **Βελτιωμένη παραγωγικότητα και ταχύτητα (Improved productivity and speed)**: Διαφορετικές ομάδες μπορούν να εργαστούν σε διαφορετικά μέρη ταυτόχρονα, επιταχύνοντας τις διαδικασίες ανάπτυξης και διασφάλισης ποιότητας. Κάθε microservice μπορεί να δοκιμαστεί ανεξάρτητα.

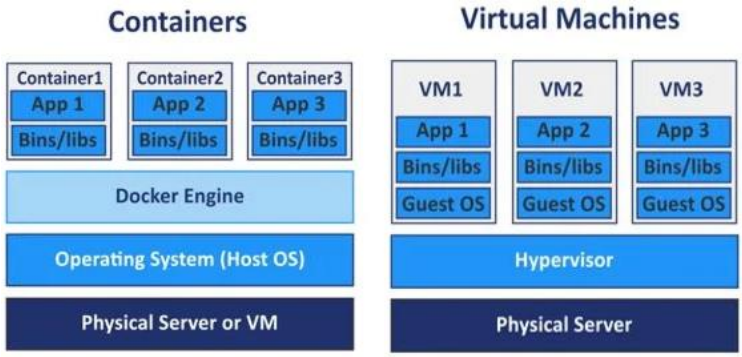
- **Ευελιξία στην υλοποίηση τεχνολογιών και στις δυνατότητες κλιμακούμενης ανάπτυξης (Flexibility in implementing technologies)**: Τα microservices επιτρέπουν τη χρήση διαφορετικών τεχνολογιών στην ίδια αρχιτεκτονική, απλοποιώντας την επιλογή τεχνολογικού σωρού και την κλιμάκωση. Επιπλέον, οι διαδικασίες λειτουργούν ανεξάρτητα από διαφορετικές γλώσσες προγραμματισμού.

- **Αυτόνομες, διασυνοριακές ομάδες (Autonomous, cross-functional)**: Τα microservices επιτρέπουν σε κατανεμημένες ομάδες να εργάζονται αυτόνομα και να προβαίνουν σε τεχνικές αποφάσεις γρήγορα και αποτελεσματικά. Αυτές οι ομάδες είναι ιδανικές για μεγάλης κλίμακας λύσεις και για ανάπτυξη μέσω διαδικτύου.

2.3 Docker

Το Docker είναι μια ανοικτού κώδικα (open source) πλατφόρμα [3] που παρέχει δυνατότητες στους developers να εκτελούν/διαχειρίζονται εφαρμογές σε διαφορετικά περιβάλλοντα με εύκολο και αποτελεσματικό τρόπο.

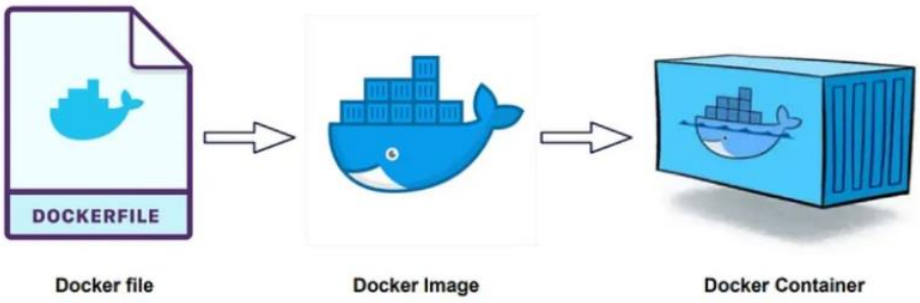
Ένα από τα μεγάλα πλεονεκτήματα του container είναι ότι απ' τη στιγμή που δημιουργηθούν μπορούν έπειτα να εκτελεστούν σε οποιοδήποτε περιβάλλον υποστηρίζει Docker [4], με αποτέλεσμα την εύκολη μετακίνηση της εφαρμογής στα περιβάλλοντα ανάπτυξης.



Σχήμα 2. 4: Containers vs Virtual Machines

Με τη χρήση των containers μπορεί να πραγματοποιηθεί η εξοικονόμηση χώρου πολύ καλύτερα σε σχέση με τα Virtual Machines. Κάθε εφαρμογή συσκευάζεται με τα αντίστοιχα υλικά στοιχεία (βλ. σχήμα 2.5). Η ανάπτυξη με χρήση των containers είναι επίσης πιο αποδοτική από την ανάπτυξη με χρήση των VMs επειδή το Docker Engine έχει πρόσβαση απευθείας στον πυρήνα του λειτουργικού συστήματος. Σε έναν υπολογιστή τα πολλαπλά containers εκτελούνται ως απομονωμένες (isolated) διεργασίες.

Το λογισμικό που διατηρεί τα containers και για το οποίο θα μιλήσουμε στην υποενότητα 2.3.1 ονομάζεται Docker Engine.

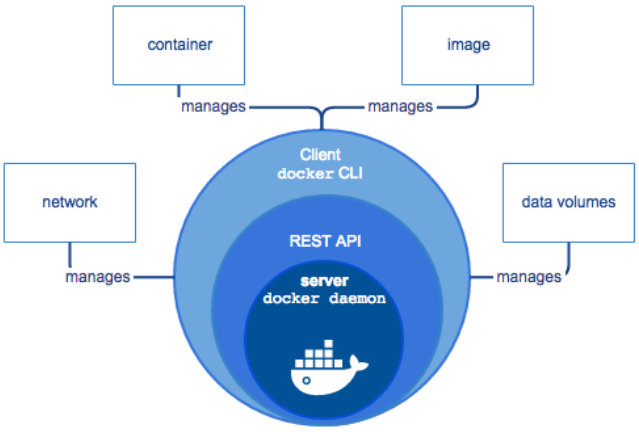


Σχήμα 2. 5: Απεικόνιση δημιουργίας του Docker Container

2.3.1 Docker engine

Το Docker engine είναι ένα software που χρησιμοποιείται για τη δημιουργία και τη διαχείριση των containers [5]. Μπορεί να χρησιμοποιηθεί από πολλές άλλες εφαρμογές και εργαλεία που έχουν ενοποιήσει τη δυνατότητα διαχείρισης των containers, όπως για παράδειγμα το Kubernetes, χρησιμοποιεί το Docker engine για τη δημιουργία και τη διαχείριση των containers.

Το Kubernetes ενορχηστρώνει τα containers. Ενημερώνει το ίδιο το Kubernetes τις ενέργειες που θα θέλαμε να κάνουμε (διαχείριση/εκτέλεση των containers) μέσω του Docker engine αντί να τα εκτελούμε εμείς οι ίδιοι.



Σχήμα 2. 6: Docker Engine

2.3.2 Container

Κατά την εκτέλεση ενός container, μπορούμε να χαρτογραφήσουμε και να αποκαλύψουμε τις θύρες του. Τα containers μπορούν να πολλαπλασιαστούν [6], να σταματήσουν και να ξεκινήσουν οποιαδήποτε στιγμή (μέσω του K8s). Αυτό σημαίνει ότι δεν υπάρχουν μόνιμα δεδομένα μέσα σε ένα container. Προσθέτοντας σε αυτό, κατά την εκτέλεση ενός container, μπορούμε να τοποθετήσουμε εξωτερικού δίσκο αρχείων μέσα στο ίδιο το container για να διατηρήσουμε τα δεδομένα του ακόμη και μετά την επανεκκίνηση ή την διαγραφή του container.

2.3.3 Dockerfile

Το Dockerfile [7] είναι ένα αρχείο που λέει στο docker πως να δημιουργήσει μια εικόνα (image). Αποτελείται από οδηγίες που περιγράφουν τη διαδικασία προετοιμασίας του περιβάλλοντος και δημιουργία της εφαρμογής. Κάθε microservice θα πρέπει να αντιστοιχεί σε ένα Dockerfile.

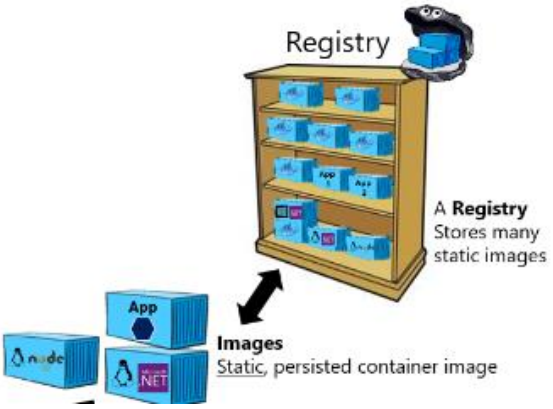
2.3.4 Εικόνα (Image)

Το image είναι αυτό που παράγεται συνήθως από ένα Dockerfile κατά τη διάρκεια μιας διαδικασίας. Κάθε Image που παράγεται μπορεί να τοποθετηθεί στο docker αποθετήριο (docker repository).

Σχήμα 2. 7: Docker Image

2.3.5 Κεντρικό Αποθετήριο Docker

Το Docker registry είναι μια πλατφόρμα που αποθηκεύει εκδόσεις και διαχειρίζεται Docker images. Τα Docker images [8] είναι τοποθετημένα και αριθμημένα με βάση το version στο Docker Registry. Το βασικό Docker registry είναι το docker hub.



Σχήμα 2. 8: Δυνητική Ερμηνεία του Docker Hub Registry

2.4 Kubernetes

Το Kubernetes, συχνά συντομογραφημένο ως "k8s", είναι μια πλατφόρμα ανοιχτού κώδικα για την ενορχήστρωση των containers που αυτοματοποιεί την ανάπτυξη, την κλιμάκωση και τη διαχείριση εφαρμογών που είναι συσκευασμένες σε containers [9]. Στη συντομογράφηση του Kubernetes (k8s), ο αριθμός 8 στο "k8s" αναφέρεται στα οκτώ γράμματα ανάμεσα στο πρώτο γράμμα "k" και το τελευταίο γράμμα "s" στη λέξη "Kubernetes".

Σκοπός όλης αυτής της ενορχήστρωσης είναι η ανάπτυξη των containers με αυτόματο τρόπο. Ο συνδυασμός των microservices και της ενορχήστρωσης βοηθάει στο να αντιμετωπίσουμε τα ακόλουθα ζητήματα σε περιβάλλοντα παραγωγής και όχι μόνο: ανοχή σε σφάλματα, καλύτερη διαχείριση κόστους, υψηλή διαθεσιμότητα (HA, High Availability) μεταξύ των συστημάτων και οριζόντια κλιμάκωση.

Ένα cluster του Kubernetes αποτελείται από ένα σύνολο μηχανών, που αποκαλούνται κόμβοι (nodes), οι οποίοι είναι υπεύθυνοι για την εκτέλεση εφαρμογών που έχουν συσκευαστεί σε containers. Στο εσωτερικό ενός cluster του Kubernetes υπάρχουν δύο βασικά συστατικά:

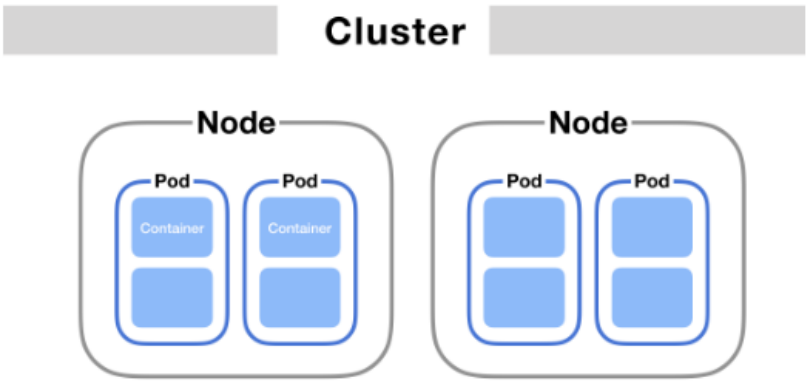
1. **Control Plane**: Το control plane διαχειρίζεται την κατάσταση του cluster. Σε περιβάλλοντα παραγωγής, συνήθως λειτουργεί σε πολλούς κόμβους που επεκτείνονται σε διάφορες ζώνες κέντρων δεδομένων (data center). Το control plane αποτελείται από διάφορα βασικά συστατικά [9], συμπεριλαμβανομένων των:

- **API Server**: Κύρια διεπαφή για την αλληλεπίδραση με το control plane.
- **etcd**: Κατανεμημένο σύστημα αποθήκευσης κλειδιών-τιμών που χρησιμοποιείται για την αποθήκευση της μόνιμης κατάστασης του cluster.
- **Scheduler**: Υπεύθυνος για τον προγραμματισμό των pods με τα worker nodes βάσει της διαθεσιμότητας των πόρων.
- **Controller Manager**: Παρακολουθεί την τρέχουσα κατάσταση του cluster μέσω του API Server και πραγματοποιεί τις κατάλληλες αλλαγές για να διατηρήσει τη λειτουργία της, εξασφαλίζοντας ότι υπάρχουν Pods σε υγιή κατάσταση.

2. **Worker Nodes**: Εκτελούν τα workloads των εφαρμογών που έχουν συσκευαστεί σε containers. Φιλοξενούν τα pods, τα οποία είναι οι μικρότερες μονάδες που μπορούν να αναπτυχθούν στο Kubernetes. Τα pods μπορούν να περιέχουν ένα ή περισσότερα containers και παρέχουν κοινόχρηστη αποθήκευση και δικτύωση για αυτά τα containers. Το control plane του Kubernetes δημιουργεί και διαχειρίζεται τα pods, καθιστώντας τα βασικά οικοδομικά στοιχεία των εφαρμογών στο Kubernetes.

2.4.1 Kubernetes Κόμβοι (Nodes)

Το Kubernetes συνήθως έχει εγκατασταθεί σε πολλούς υπολογιστές, οι οποίοι προσθέτουν το κάθε ένα τα δικά του resources και λειτουργούν ως cluster. Κάθε υπολογιστής στο cluster ονομάζεται κόμβος. Το Kubernetes είναι το κεντρικό σύστημα [9] το οποίο περικλείεται από όλους αυτούς τους κόμβους. Οι τύποι των κόμβων μπορεί να είναι δύο: master nodes που χρησιμοποιούνται για τη διαχείριση του cluster, την πρόσβαση στο API και τη διαχείριση των πόρων εντός του cluster καθώς και των worker nodes για τους οποίους όπως έχουμε ήδη αναφέρει φιλοξενούν τα containers.



Σχήμα 2. 9: Αναπαράσταση Δομής των Kubernetes Nodes

2.4.2 Στοιχεία Διαχείρισης Εφαρμογών

Υπάρχουν θεμελιώδη στοιχεία στο Kubernetes [10] και αποτελούν τα βασικά εργαλεία για τη δημιουργία, ανάπτυξη και τη διαχείριση εφαρμογών.

Ξεκινώντας από:

- Τα **pods** που είναι η μικρότερη μονάδα στο Kubernetes, μπορεί να περιέχει έναν ή και περισσότερα containers και μοιράζεται πόρους και δίκτυο.

- Αμέσως επόμενο είναι τα **deployments** όπου διαχειρίζονται την διαδικασία ανάπτυξης των pods, επιτρέποντας τη διαχείριση τους, τον έλεγχο ενημέρωσης τους και την επαναδιαμόρφωση σε περίπτωση κάποιου σφάλματος.

- Τέλος, τα **services** τα οποία είναι υπεύθυνα για την επικοινωνία μεταξύ των Pods, καθώς και για την εξισορρόπηση του workload τους για εξυπηρέτηση αιτημάτων.

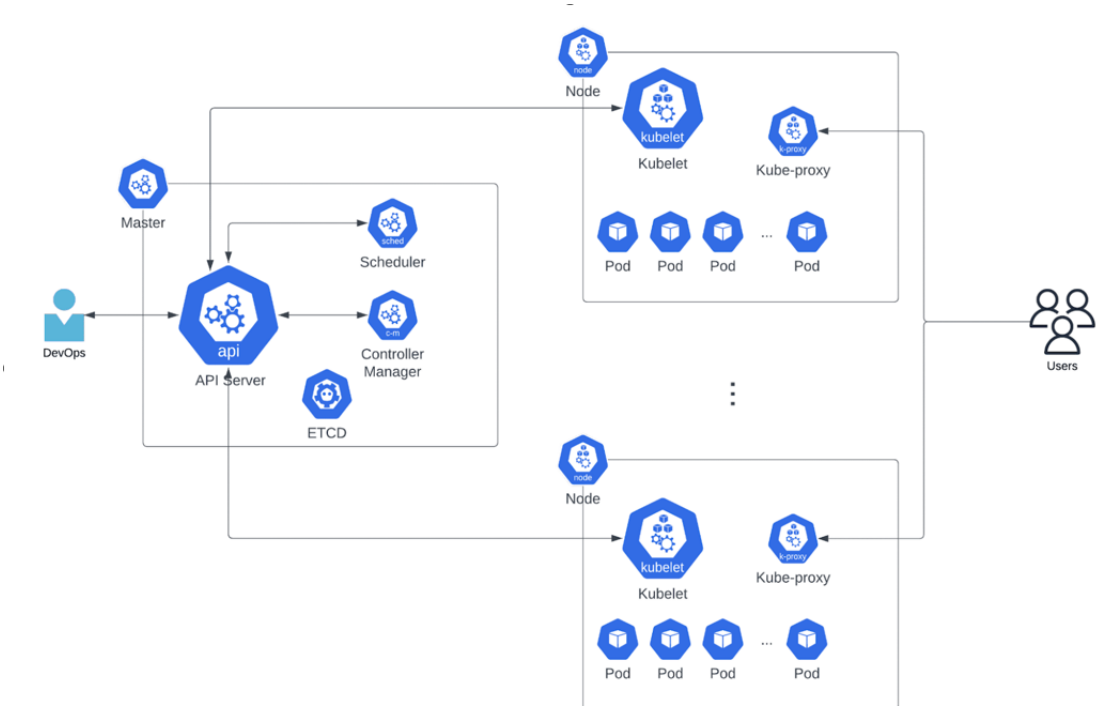
2.4.3 Στοιχεία Διαχείρισης Δικτύωσης και Συντήρησης Συστήματος

Το Kubernetes ακόμη αποτελείται από διάφορα κρίσιμα στοιχεία που συνεργάζονται για την ομαλή λειτουργία του συστήματος. Δύο από αυτά τα βασικά στοιχεία [9] [11] είναι το Kubelet και το Kubeproxy.

- Το **Kubelet** είναι ένας βασικός εκτελεστής στον κόμβο (node) του Kubernetes. Η κύρια του λειτουργία είναι να επικοινωνεί με τον API server του Kubernetes και να διαχειρίζεται τη λειτουργία των Pods που εκτελούνται στον κόμβο. Το Kubelet είναι υπεύθυνο για τη δημιουργία, την εποπτεία και τη διαχείριση των Pods σύμφωνα με τις προδιαγραφές τους, εξασφαλίζοντας ότι οι εφαρμογές λειτουργούν σωστά και αποτελεσματικά στο περιβάλλον του Kubernetes.

- Το **Kubeproxy** είναι υπεύθυνο για τη διαχείριση του δικτύου στον κόμβο του Kubernetes. Λειτουργεί ως μια διαμεσολαβητική υπηρεσία μεταξύ των διαφορετικών pods και services, επιτρέποντας την αυτόματη εύρεση και δρομολόγηση των αιτημάτων προς τις κατάλληλες εφαρμογές. Επίσης, διαχειρίζεται τους κανόνες δικτύου και την επικοινωνία μεταξύ των Pods εντός του cluster, εξασφαλίζοντας τη σωστή λειτουργία των εφαρμογών σε διάφορες δικτυακές συνθήκες.

Αφού ολοκληρώθηκε η ενημέρωση όλων τα βασικών στοιχείων ενός Kubernetes Cluster, για τη καλύτερη κατανόηση αυτών και των λειτουργιών τους, απεικονίζεται στο σχήμα 2.10 η αρχιτεκτονική ενός Kubernetes Cluster με τα αντίστοιχα στοιχεία που αναφέρθηκαν στις τελευταίες υποενότητες.



Σχήμα 2. 10: Kubernetes Αρχιτεκτονική

2.4.4 Κατάλληλη στιγμή για τη χρήση του Kubernetes

Όπως και με κάθε τεχνολογία, υπάρχουν συμβιβασμοί [9] που πρέπει να ληφθούν υπόψη κατά τη λήψη απόφασης για τη χρήση του Kubernetes. Ας εξετάσουμε τα πλεονεκτήματα και τα μειονεκτήματα:

Πλεονεκτήματα:

- **Κλιμακωσιμότητα** (**scalability**) **και Υψηλή Διαθεσιμότητα** (**High-availability**,**HA**): Το Kubernetes προσφέρει χαρακτηριστικά όπως αυτο-επούλωση, αυτόματη επαναφορά και οριζόντια κλιμάκωση, καθιστώντας το ιδανικό για τη διαχείριση εφαρμογών με μεταβαλλόμενες απαιτήσεις.

- **Φορητότητα**: Το Kubernetes επιτρέπει τη συνεπή και αξιόπιστη ανάπτυξη και διαχείριση εφαρμογών σε διαφορετικά περιβάλλοντα υποδομής, συμπεριλαμβανομένων των εγκαταστάσεων εντός εγκατάστασης, δημόσιου νέφους (cloud) ή υβριδικών συνδυασμών.

- **Τυποποίηση**: Το Kubernetes παρέχει ένα ομοιόμορφο τρόπο συσκευασίας, ανάπτυξης και διαχείρισης εφαρμογών, απλοποιώντας τις διαδικασίες ανάπτυξης και λειτουργίας.

Μειονεκτήματα:

- **Πολυπλοκότητα**: Η δημιουργία και η λειτουργία του Kubernetes μπορεί να είναι πολύπλοκη, απαιτώντας τεχνογνωσία και πόρους. Μπορεί να έχει υψηλό κόστος εκ των προτέρων, ιδίως για οργανισμούς που είναι νέοι στη διαχείριση της ορχηστρώσεως container.

- **Απαιτήσεις Πόρων**: Το Kubernetes απαιτεί έναν ελάχιστο επίπεδο πόρων για να υποστηρίξει πλήρως τα χαρακτηριστικά του. Μικρότεροι οργανισμοί μπορεί να το θεωρήσουν υπερβολικό και να αντιμετωπίσουν προκλήσεις για την εκπλήρωση αυτών των απαιτήσεων.

Μια λογική προσέγγιση για οργανισμούς που εξετάζουν το Kubernetes είναι να επιλέξουν υπηρεσίες διαχείρισης Kubernetes που παρέχονται από παρόχους νέφους, όπως το Amazon EKS, το Google Kubernetes Engine (GKE) και ο Azure Kubernetes Service (AKS). Αυτές οι υπηρεσίες εξοφλούν τη διαχείριση του ελεγκτικού πίνακα (control plane), συμπεριλαμβανομένων της κλιμάκωσης, της διαμόρφωσης και των εργασιών συντήρησης, επιτρέποντας στους οργανισμούς να επικεντρωθούν στην εκτέλεση των εφαρμογών τους.

Για μικρότερους οργανισμούς, είναι σημαντικό να αξιολογήσουν εάν το Kubernetes συμβαδίζει με τις άμεσες ανάγκες τους.

2.5 Helm Charts

Τα Helm charts είναι ένας τρόπος για τη διαχείριση των εφαρμογών στο Kubernetes [12]. Είναι πακέτα που περιέχουν όλες τις απαραίτητες πληροφορίες για να εγκατασταθεί και να λειτουργήσει μια εφαρμογή σε ένα Kubernetes cluster.

Ένα Helm chart αποτελείται από δύο βασικά μέρη:

1. Αρχείο Chart.yaml: Περιέχει πληροφορίες σχετικά με το chart, όπως την έκδοση, τον περιγραφέα, τον δημιουργό κ.λπ.

2. Αρχεία Πρότυπα (templates): Αυτά τα αρχεία περιέχουν τις διαμορφώσεις YAML των αντικειμένων Kubernetes που απαιτούνται για να εκτελεστεί η εφαρμογή, όπως Pods, Services, Deployments κ.λπ. Οι τιμές στα αρχεία τεμπλέιτς μπορούν να παραμετροποιηθούν, επιτρέποντας την ευελιξία στη διαμόρφωση της εγκατάστασης.

Ένας χρήστης μπορεί να εγκαταστήσει ένα Helm chart στο Kubernetes cluster του ακολουθώντας μια σειρά από εντολές Helm, όπως helm install ή helm upgrade, που δημιουργούν τα αντικείμενα Kubernetes που περιγράφονται στα templates. Το Helm αναλαμβάνει τη διαχείριση των εξαρτήσεων και των εκδόσεων των εφαρμογών, παρέχοντας έναν τρόπο οργανωμένο και επαναχρησιμοποιήσιμο για την ανάπτυξη και τη διαχείριση εφαρμογών σε ένα Kubernetes περιβάλλον.

2.6 Επίλογος

Το Kubernetes έχει επαναστατήσει τον τρόπο ενορχήστρωσης των containers, παρέχοντας ισχυρά εργαλεία για την ανάπτυξη, την αναπροσαρμογή και τη διαχείριση εφαρμογών που χρησιμοποιούν containers. Η ευελιξία, η κλιμακούμενη δυνατότητα και η φορητότητά του καθιστούν το Kubernetes μια δημοφιλή επιλογή για τη σύγχρονη ανάπτυξη και λειτουργία εφαρμογών. Αν και μπορεί να προκαλέσει πολυπλοκότητα και να απαιτήσει περισσότερους πόρους, οι υπηρεσίες διαχειριζόμενου Kubernetes προσφέρουν μια βιώσιμη εναλλακτική λύση για εταιρείες διαφορετικού μεγέθους, επιτρέποντάς τους να εκμεταλλευτούν τα πλεονεκτήματά του χωρίς το φορτίο της διαχείρισης της υποδομής.

Κεφάλαιο 3ο: Αρχές της Συνεχούς Ενσωμάτωσης / Συνεχούς Παράδοσης (CI/CD)

3.1 Εισαγωγή

Έπειτα, μετά την εξέταση του Kubernetes, θα αναλύσουμε λεπτομερώς τις βασικές αρχές του Continuous Integration (CI) και Continuous Deployment (CD). Αυτές οι αρχές αποτελούν το βασικό κομμάτι της διαδικασίας ανάπτυξης λογισμικού και έχουν ως στόχο την επίτευξη συνεχούς ενσωμάτωσης και ανάπτυξης υψηλής ποιότητας λογισμικού.

Η σύγχρονη ανάπτυξη λογισμικού έχει εξελιχθεί δραματικά με την πάροδο του χρόνου και μια από τις βασικές τάσεις που έχουν επιφέρει σημαντική αλλαγή στον τρόπο που αντιμετωπίζουμε τη διαδικασία ανάπτυξης είναι το Continuous Integration και το Continuous Deployment, γνωστά ως CI/CD. Αυτές οι διαδικασίες αποτελούν τη βάση μιας αυτόματης διαδικασίας ανάπτυξης λογισμικού [1] [13], η οποία επιδιώκει τη συνεχή ανάπτυξη λειτουργικού και ποιοτικού λογισμικού στον πελάτη. Επισκόπηση των εργαλείων και τεχνολογιών που χρησιμοποιούνται (CI/CD Jenkins, Git (Bitbucket), Artifact Repository (nexus), Kubernetes, Application Server).

Το Continuous Integration (CI) αναφέρεται στη ενσωμάτωση του κώδικα από διάφορους προγραμματιστές σε ένα κοινό αποθετήριο (repository), όπου ο κώδικας αυτός υπόκειται σε διάφορα τεστ και ελέγχους αυτόματα, εξασφαλίζοντας έτσι τη συνεχή ορθή λειτουργία του. Από την άλλη πλευρά, το Continuous Deployment (CD) αφορά την αυτόματη διαδικασία ανάπτυξης στα κατάλληλα περιβάλλοντα όπως π.χ. δοκιμαστικών (test) και παραγωγικών (prod) περιβαλλόντων, με στόχο την γρήγορη και ασφαλή παράδοση του λογισμικού σε παραγωγική χρήση.

Η υιοθέτηση του CI/CD έχει επιφέρει πολλά οφέλη στη διαδικασία ανάπτυξης λογισμικού. Πέραν της μείωσης του χρόνου παράδοσης λογισμικού και της βελτίωσης της ποιότητας του, το CI/CD ενισχύει τη συνεργασία μεταξύ των μελών της ομάδας ανάπτυξης, ενθαρρύνει τη συνεχή ενσωμάτωση των αλλαγών και την άμεση ανταπόκριση σε προβλήματα ή σφάλματα και ενισχύει την αξιοπιστία του λογισμικού.

Παράλληλα, η τάση αυτή έχει προκαλέσει την εμφάνιση νέων τεχνολογιών και μεθοδολογιών που ενισχύουν ακόμη περισσότερο τη διαδικασία ανάπτυξης λογισμικού.

3.2 Από τη Χειροκίνητη στην Αυτόματη Ανάπτυξη

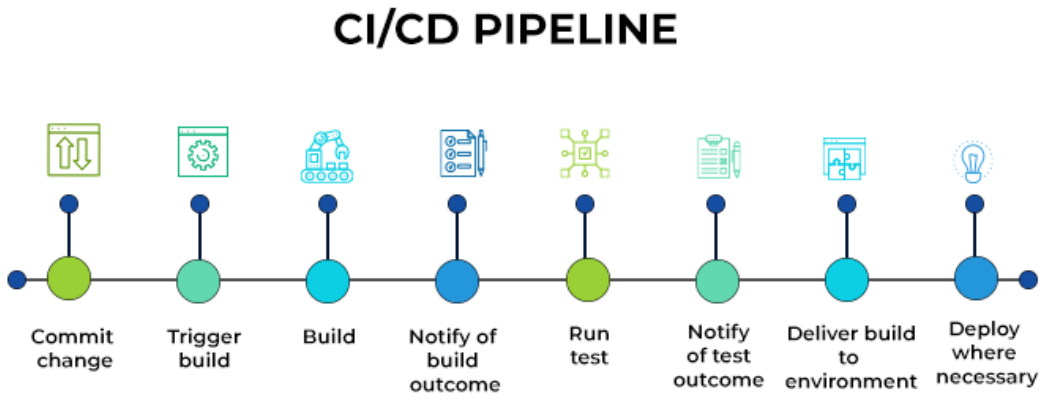
Η συνεχής Ενσωμάτωση και Συνεχής Ανάπτυξη (CI/CD) αποτελεί ένα αυτόματο ροής εργασιών που συνδυάζει τον κώδικα που αναπτύσσουν οι προγραμματιστές λογισμικού και τον αναπτύσσει σε ένα περιβάλλον με ελάχιστη ανθρώπινη παρέμβαση.

Πριν από την ύπαρξη των CI/CD pipelines, οι προγραμματιστές ανέλαβαν χειροκίνητα τον κώδικα, όπως για παράδειγμα τη συγγραφή του κώδικα, εκτελούσαν δοκιμές τοπικά στα συστήματά τους, κάνανε build λαμβάνοντας υπόψη και κάποιου είδους version, και στη συνέχεια κάνανε χειροκίνητη μεταφορά του εκτελέσιμου σε τεστ διακομιστές (servers). Αφού ολοκληρώνανε τα επιτυχώς τεστς, οι προγραμματιστές θα ενημέρωναν την ομάδα του operations να αναπτύξει τον κώδικα χειροκίνητα πάλι

στην παραγωγή. Αυτή η ιδέα λειτουργούσε καλά με τις μονολιθικές εφαρμογές όταν η συχνότητα ανάπτυξης ήταν μία φορά κάθε λίγους μήνες.

Με την εμφάνιση των μικρουπηρεσιών, όμως, οι προγραμματιστές άρχισαν να δημιουργούν μικρότερα σενάρια χρήσης πιο γρήγορα και ήθελαν να τα αναπτύσσουν αρκετά συχνά αναλόγως με τις ανάγκες του κάθε έργου. Η διαδικασία χειρισμού της εφαρμογής χειροκίνητα μετά την καταχώρηση του κώδικα ήταν επαναλαμβανόμενη, εκνευριστική και επιρρεπής σε σφάλματα.

Εδώ έρχονται στο προσκήνιο οι ευέλικτες μέθοδοι και οι αρχές του DevOps με το CI/CD στο επίκεντρό τους. Η ιδέα είναι να κατασκευάζονται και να αποστέλλονται αρκετές αλλαγές στο παραγωγικό περιβάλλον πιο συχνά αλλά και πολύ πιο γρήγορα χωρίς ανθρώπινες παρεμβάσεις που συνεπάγεται αρκετές φορές με σφάλματα. Ένα CI/CD pipeline έκανε ολόκληρη τη διαδικασία αυτόματη και υψηλής ποιότητας κώδικα αποστέλλονταν σε παραγωγικό περιβάλλον πολύ πιο γρήγορα και αποτελεσματικά.



Σχήμα 3. 1: CI/CD Pipeline

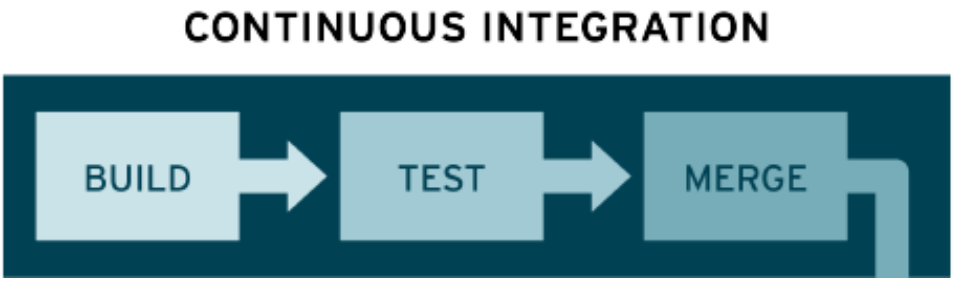
3.3 Βασικές αρχές CI/CD

- **Συνεχής Ενσωμάτωση (Continuous Integration):** Η Συνεχής Ενσωμάτωση (Continuous Integration - CI) αποτελεί σημαντικό κομμάτι της διαδικασίας ανάπτυξης λογισμικού [14]. Σε αυτό το στάδιο, η βασική ιδέα είναι να υπάρχει αυτόματη δημιουργία του λογισμικού κάθε φορά που οι προγραμματιστές καταχωρούν νέο κώδικα στο repository. Αν οι αλλαγές στον κώδικα πραγματοποιούνται συχνά – καθημερινά [15], τότε χρειάζεται να υπάρχει ένας αυτόματος τρόπος για να δημιουργείται και να ελέγχεται το λογισμικό κάθε μέρα. Αυτή η διαδικασία ονομάζεται μερικές φορές "αγωγός δημιουργίας" (build pipeline). Το τελικό πρόγραμμα ή αρχείο πηγαίου κώδικα (artifact) στη συνέχεια αποθηκεύεται σε ένα αποθετήριο (repository) μετά από διάφορες δοκιμές και ελέγχους.

- Το Continuous Integration (CI) αναφέρεται στη συνεχή ενσωμάτωση των αλλαγών του κώδικα από διάφορους προγραμματιστές σε ένα κοινό αποθετήριο (repository).

- Κάθε φορά που γίνεται μια αλλαγή, ο κώδικας ελέγχεται αυτόματα για πιθανά σφάλματα και συγκρούσεις με τον υπάρχοντα κώδικα.

- Ο στόχος είναι η άμεση ανίχνευση και διόρθωση των προβλημάτων, εξασφαλίζοντας έτσι ότι ο κώδικας είναι πάντα σε μια σταθερή και λειτουργική κατάσταση.



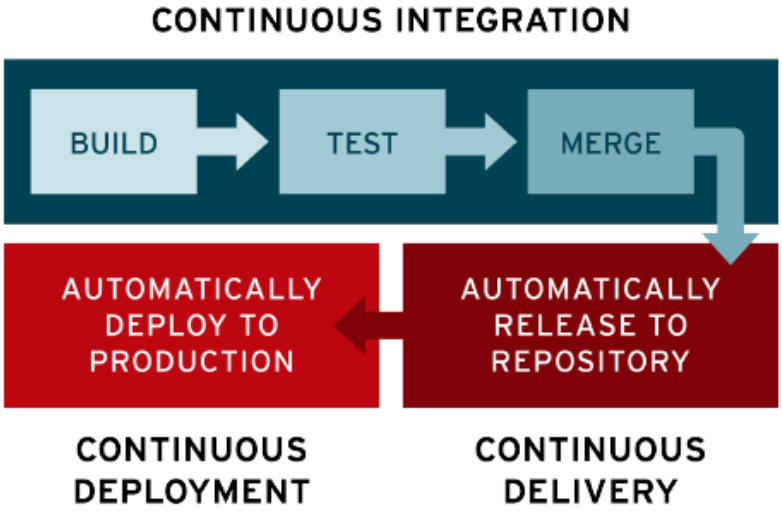
Σχήμα 3. 2: Διαδικασία συνεχούς ενσωμάτωσης

- **Συνεχής Ανάπτυξη (Continuous Deployment):** Το στάδιο της Συνεχούς Ανάπτυξης (Continuous Deployment), πολύ πιθανό να το ακούσετε και Συνεχής Παράδοση (Continuous Delivery), αν και οι δύο όροι έχουν διαφορετικές σημασίες όπως θα δείτε στο σχήμα 3.3, αναφέρεται στη διαδικασία ανάκτησης του αρχείου πηγαίου κώδικα από το αποθετήριο (git repository) και τη συχνή ασφαλή ανάπτυξή του [13]. Ένας αγωγός (pipeline) συνεχούς ανάπτυξης χρησιμοποιείται για την ανάπτυξη εφαρμογών σε τεστ περιβάλλοντα και παραγωγής με λιγότερη ανθρώπινη παρέμβαση.

  - Το Continuous Deployment (CD) επεκτείνει την έννοια του CI με την διαδικασία παράδοσης του λογισμικού σε περιβάλλοντα δοκιμών και παραγωγής.

  - Κάθε φορά που ολοκληρώνεται μια νέα ενσωμάτωση, το σύστημα διασφαλίζει ότι το λογισμικό είναι έτοιμο για παράδοση σε περιβάλλοντα παραγωγής.

  - Τέλος, αφού όλα τα παραπάνω ολοκληρωθούν επιτυχώς ξεκινάει αυτόματα η ανάπτυξη (deployment) στο κατάλληλο περιβάλλον.



Σχήμα 3. 3: Διαδικασία συνεχούς ενσωμάτωσης – Συνεχούς ανάπτυξης

- **Αυτόματη Διαδικασία:**

  - Η αυτόματη διαδικασία είναι κεντρικής σημασίας στο CI/CD, με τη χρήση εργαλείων και σεναρίων που εκτελούν αυτόματα τις διαδικασίες ελέγχου, δοκιμής, παράδοσης κι ανάπτυξης του λογισμικού.

  - Εξασφαλίζει τη συνέπεια και την επανάληψη των διαδικασιών, μειώνοντας τον ανθρώπινο παράγοντα και τον κίνδυνο σφαλμάτων.

- **Συνεργασία Ομάδας:**

  - Το CI/CD ενθαρρύνει τη συνεργασία μεταξύ των μελών της ομάδας ανάπτυξης και των διαχειριστών συστήματος.

  - Η συνεχή επικοινωνία βελτιώνει την ποιότητα του κώδικα και την αποτελεσματικότητα της ανάπτυξης.

Στο παρακάτω πίνακα δίνονται κάποιες από τις διαφορές μεταξύ του συνεχούς ενσωμάτωσης και συνεχούς ανάπτυξης [16]:

Συνεχούς Ενσωμάτωσης (CI)	**Συνεχούς Ανάπτυξη (CD)**
Το CI περιλαμβάνει την ενσωμάτωση του κώδικα στο κύριο κώδικα.	Η Συνεχής Παράδοση (CD) περιλαμβάνει τη δοκιμή, τη σταθερότητα και την ανάπτυξη κώδικα για την παροχή ενημερώσεων εφαρμογών στους χρήστες.
Ενσωματώνει αλλαγές κώδικα σε ένα κοινόχρηστο αποθετήριο σε τακτικά χρονικά διαστήματα.	Ο κώδικας μπορεί να αναπτυχθεί στο περιβάλλον παραγωγής.
Χρησιμοποιεί αυτοματισμό για να ανακαλύψει – παρακολουθήσει προβλήματα και να επικυρώσει γρήγορα τον νέο κώδικα πριν από την ενσωμάτωση.	Χρησιμοποιεί αυτοματισμό για ταχύτερη απελευθέρωση νέου κώδικα.
Επιτρέπει τη διαφάνεια και την προνοητικότητα στην ανάπτυξη και παράδοση λογισμικού.	Διευκολύνει αποτελεσματικές και επαναλαμβανόμενες διαδικασίες απελευθέρωσης.
Μειώνει τα έξοδα, εμπνέει εμπιστοσύνη, εξασφαλίζει συνεπή διαδικασία κατασκευής, μετριάζει τους κινδύνους, βελτιώνει την επικοινωνία της ομάδας.	Μειώνει τον κίνδυνο, παρέχει λογισμικό με λιγότερα προβλήματα, ανταποκρίνεται γρήγορα στις συνθήκες της αγοράς και κυκλοφορεί νέα προϊόντα πιο τακτικά, μεταξύ άλλων πλεονεκτημάτων.
Επιτρέπει την αυτόματη δοκιμή και τη συνεχή ενοποίηση με κώδικα από άλλους προγραμματιστές.	Κώδικας που διατίθεται στους επαγγελματίες χρήστες μετά την επιτυχή αποδοχή στο στάδιο CI.
Λιγότερο περίπλοκο και λιγότερο ακριβό.	Πιο περίπλοκο και πιο ακριβό από το CI.

Πίνακας 3. 1: Διαφορές μεταξύ Συνεχούς Ενσωμάτωσης (CI) & Συνεχούς Ανάπτυξη (CD)

3.4 Ανάλυση CI/CD με Kubernetes

Σε αυτή την ενότητα θα αναλύσουμε την ενσωμάτωση των δύο μαζί CI/CD και Kubernetes, τα εργαλεία τα οποία χρησιμοποιούνται, τα προγράμματα που πρέπει να αλληλοεπιδράσουν μεταξύ τους ώστε να δημιουργηθεί το Pipeline.

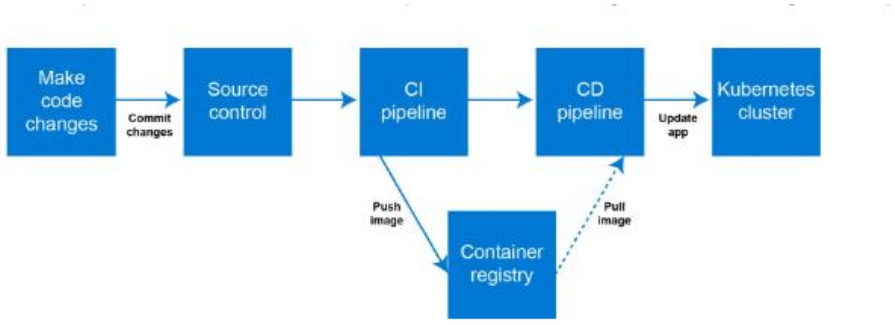
- **Συνεχής Ενσωμάτων (CI)**

  Όταν ο κώδικας του κάθε προγραμματιστή γίνει commit στο αποθετήριο (repository) [16], είναι καλή πρακτική να επιτραπεί σε άλλους προγραμματιστές να τον ελέγξουν. Όταν οι αρμόδιοι προγραμματιστές ολοκληρώσουν με τον έλεγχο και δώσουν την έγκριση, τότε ξεκινάει η συνεχής ενσωμάτωση pipeline. Ο κώδικας εντός του pipeline θα περάσει από ανάλυση για να εντοπίσει πιθανόν σφάλματα. Το κρίσιμο στάδιο της συνεχής ενσωμάτωσης (CI/CD) είναι η δημιουργία του microservice και το επόμενο η δοκιμή (το λεγόμενο testing). Για να μπορέσει να δημιουργηθεί αυτό θα πρέπει να έχει τοποθετηθεί ένα αρχείο με την ονομασία Dockerfile

στο git αποθετήριο (repository) με την κατάλληλη σύνταξη που εξηγεί πως δημιουργείται ένα image. Το pipeline δημιουργεί το container image και το καταχωρεί στη docker registry.

- **Συνεχής Ανάπτυξη (CD)**

  Αφού δημιουργηθεί το container image και αποθηκευτεί στο docker repository, το CD αναλαμβάνει να αναπτύξει την εικόνα (image) [16] στο περιβάλλον παραγωγής μέσα σε container, αυτό επιτυγχάνεται μέσω της ρύθμισης του Kubernetes. Το container σηκώνεται/αναπτύσσεται στο Kubernetes Cluster. Ωστόσο, οι διαδικασίες ανάπτυξης μπορούν να διαφέρουν ανάλογα με τις ανάγκες των έργων.



Σχήμα 3. 4: Kubernetes CI/CD Διαδικασία

3.5 Επίλογος

Κατά τη διάρκεια αυτού του κεφαλαίου, εξετάσαμε τις βασικές αρχές της CI/CD, περιγράφοντας επιπλέον τη συνεργασία της συνεχούς ενσωμάτωσης και ανάπτυξης μεταξύ ενός Kubernetes περιβάλλον, καθώς και τις πρακτικές που απαιτούνται για την επιτυχή εφαρμογή τους. Αναδείξαμε τη σημασία της διασφάλισης της ποιότητας του λογισμικού και της διαδικασίας παράδοσης, προκειμένου να επιτευχθεί μεγαλύτερη ευελιξία, αξιοπιστία και ταχύτητα στην ανάπτυξη λογισμικού.

Κεφάλαιο 4ο: Ανάλυση της ροής εργασιών

4.1 Εισαγωγή

Κατά την διάρκεια αυτής της ανάλυσης, θα εξετάσουμε λεπτομερώς κάθε στάδιο της ροής, περιγράφοντας τις διαδικασίες και τα εργαλεία που χρησιμοποιούνται, καθώς και τους στόχους και τα οφέλη που επιδιώκονται με κάθε βήμα. Ανάλογα με τις ανάγκες και τις απαιτήσεις του συγκεκριμένου έργου, το workflow μπορεί να προσαρμοστεί και να επεκταθεί για να καλύψει οποιαδήποτε ειδική ανάγκη ή απαίτηση ανάπτυξης λογισμικού.

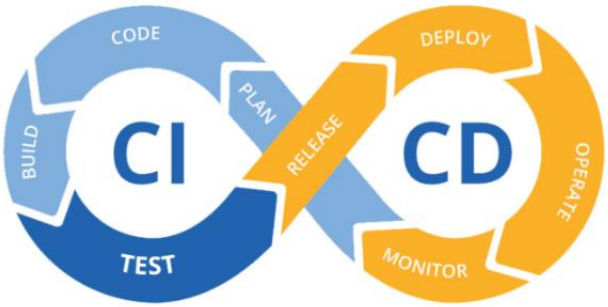
4.2 Περιγραφή διαδικασίας ενσωμάτωσης (CI) και Ανάπτυξης (CD)

Η διαδικασία συνεχούς ενσωμάτωσης (Continuous Integration - CI) και συνεχούς ανάπτυξης (Continuous Deployment - CD) αποτελεί ένα κρίσιμο στάδιο στον κύκλο ζωής ανάπτυξης λογισμικού [13]. Το CI επικεντρώνεται στον συχνό έλεγχο του κώδικα και την ολοκλήρωσή του σε ένα κοινό αποθετήριο, ενώ το CD στοχεύει στην παράδοση του λογισμικού σε παραγωγικό περιβάλλον.

Η διαδικασία του CI ξεκινά με τον προγραμματιστή να κάνει push τον κώδικά του στο κοινό αποθετήριο. Στη συνέχεια, ένα CI εργαλείο όπως το Jenkins ελέγχει το αποθετήριο για αλλαγές και εκτελεί αυτόματα μια σειρά διαδικασιών, όπως μεταγλώττιση, μοναδικές δοκιμές, και ολοκλήρωση. Σε περίπτωση που υπάρχουν σφάλματα ή ασυμβατότητες, η διαδικασία CI ειδοποιεί τους προγραμματιστές για να διορθώσουν τα προβλήματα. Αυτή η συνεχής ενσωμάτωση εξασφαλίζει την ποιότητα του κώδικα [17] και τη σταθερότητα της εφαρμογής.

Το CD επεκτείνει τη διαδικασία του CI στο στάδιο της ανάπτυξης. Μετά την επιτυχή ολοκλήρωση της διαδικασίας CI, το CD εκτελεί αυτόματα την ανάπτυξη του λογισμικού σε ένα παραγωγικό περιβάλλον. Αυτό περιλαμβάνει τη δημιουργία και την ενημέρωση images, την αναβάθμιση των υποδομών και την αυτόματη εκτέλεση δοκιμών. Με το CD, οι αλλαγές στον κώδικα παραδίδονται στον παραγωγικό περιβάλλον με ασφάλεια και αξιοπιστία.

Αυτές οι διαδικασίες επιτρέπουν στις ομάδες ανάπτυξης λογισμικού να επιτυγχάνουν συνεχή ενσωμάτωση και ανάπτυξη, βελτιώνοντας την ποιότητα, την απόδοση και την ανταπόκριση των εφαρμογών τους.



Σχήμα 4. 1: Κύκλος ζωή CI/CD

4.3 Στάδια του pipeline για την ανάπτυξη της εφαρμογής

Η ανάλυση που γίνεται σε αυτή την ενότητα βασίζεται στα στάδια ενός pipeline χρησιμοποιώντας CI/CD σε Kubernetes υποδομή καθώς σε μια τέτοια υποδομή επιλέχτηκε να στηθεί η εργασία.

4.3.1 Πηγαίος Κώδικας

Ο πηγαίος κώδικας είναι αυτός που οι προγραμματιστές [18] [19] γράφουν κώδικα για εφαρμογές. Μόλις γραφεί νέος πηγαίος κώδικας, αυτός ανεβαίνει σε κεντρική αποθήκη (central repository) σε ένα απομακρυσμένο αποθετήριο (συνήθως κάποια έκδοση του git), όπου αποθηκεύονται ο κώδικας της εφαρμογής και οι ρυθμίσεις. Είναι ένα κοινό αποθετήριο μεταξύ των προγραμματιστών, όπου ενσωματώνουν συνεχώς αλλαγές κώδικα στο αποθετήριο, συνήθως καθημερινά. Αυτές οι αλλαγές στο αποθετήριο κώδικα ενεργοποιούν το CI pipeline.

4.3.2 Έλεγχος/Τεστ Κώδικα

Ο έλεγχος/τεστ του κώδικα αφού έχει ενημερωθεί με τις νέες αλλαγές [18], το τεστ καθορίζει για το αν ο κώδικας είναι σωστός και μπορεί να προχωρήσει στο build του image. Στη περίπτωση της εργασίας, το εργαλείο που χρησιμοποιείται λόγω της Java εφαρμογής είναι το Maven. Κατά τη διάρκεια του σταδίου τεστ, το maven test εκτελεί unit tests που έχουν οριστεί στο κώδικα του έργου. Τα unit tests έχουν δηλωθεί από τον προγραμματιστή για να ελέγξει τη ορθή λειτουργία των τμημάτων του κώδικα. Στη περίπτωση σφαλμάτων, το pipeline σταματάει και προειδοποιεί για το σφάλμα που εμφανίστηκε. Ο σκοπός του σταδίου είναι να εντοπισθούν πιθανά σφάλματα και προβλήματα στον κώδικα και να διασφαλίσει την ποιότητα του λογισμικού πριν από την ανάπτυξη σε περιβάλλοντα παραγωγής.

4.3.3 Δημιουργία Εκτελέσιμων Αρχείων (Packaging)

Το packaging βοηθάει στη δημιουργία/παραγωγή των εκτελέσιμων αρχείων [20] που μπορούν να εκτελεστούν ανεξάρτητα σε διάφορα περιβάλλοντα. Επιπλέον, επιτρέπει την εύκολη μεταφορά και αναβάθμιση του λογισμικού σε διάφορα συστήματα χωρίς την ανάγκη για επανασυσκευασία ή επαναδημιουργία του κώδικα.

Συνολικά, η διαδικασία Packaging είναι σημαντική στην διαχείριση και παραγωγή του λογισμικού, καθώς βοηθά στη διασφάλιση της συνοχής και της σταθερής παράδοσης-ανάπτυξης.

4.3.4 Κατασκευή Εικόνας (Build Image)

Κατά τη διαδικασία αυτή, ο κώδικας της εφαρμογής συσκευάζεται σε ένα μοναδικό εκτελέσιμο αρχείο [21], με σκοπό να διευκολυνθεί η μεταφορά του μέχρι τη διαδικασία της ανάπτυξης. Αυτή η διαδικασία γνωστοποιείται ως "Build". Στη συνέχεια, το αντικείμενο που κατασκευάστηκε μετατρέπεται σε μια εικόνα (image) που μπορεί να αναπτυχθεί σε ένα περιβάλλον που χρησιμοποιεί τεχνολογίες όπως το Kubernetes.

4.3.5 Ανάπτυξη εικόνας (image) ως Container

Σ' αυτό το στάδιο της ανάπτυξης του image [18], η εφαρμογή που έχει κατασκευαστεί σε εικόνα (image) εξάγεται από το repository και αναπτύσσεται ως container σ' ένα περιβάλλον Kubernetes. Αυτό το περιβάλλον μπορεί να είναι δοκιμαστικό ή παραγωγικό. Το βήμα αυτό είναι αρκετά σημαντικό καθώς αντιπροσωπεύει τη μετάβαση της εφαρμογής από το στάδιο της ανάπτυξης στο στάδιο της παραγωγής, στο οποίο θα γίνουν αντιληπτές οι αλλαγές στον τελικό χρήστη. Το στάδιο αυτό αντιμετωπίζει μικρή δυσκολία λόγω της διαχείρισης των πόρων και των πολλών μεθόδων ανάπτυξης που μπορούν να χρησιμοποιηθούν.

4.4 Αναλυτική Περιγραφή της Ροής Εργασιών

Η ροή εργασιών που ακολουθείται στο πλαίσιο αυτής της εργασίας είναι μια σειρά από αυτόματες διαδικασίες που εφαρμόζονται για τη διαχείριση και την επεξεργασία του κώδικα, των ελέγχων, των δοκιμών, καθώς και την παράδοση - ανάπτυξη του λογισμικού σε ένα Kubernetes περιβάλλον.

Για τη δημιουργία του cluster έχει χρησιμοποιηθεί το Kubernetes AKS που προσφέρει η Microsoft στο Azure (cloud) το Kubernetes ως υπηρεσία. Πάνω στο cluster όπως θα δείτε (βλ. Σχήμα 4.2) ανήκουν αρκετά containers που βοηθούν στη δημιουργία και τη σωστή ροή του cluster. Τρία (3) είναι τα βασικά containers που μας ενδιαφέρουν παρακάτω (σε πορτοκαλί πλαίσιο): Calculator-app, Jenkins, Nexus repository**,** τα οποία στην ουσία είναι οι πρωταγωνιστές αυτής της εργασίας.

```
PS C:\thesis> kubectl get pods -A
NAMESPACE          NAME                                                READY   STATUS    RESTARTS   AGE
helm-deployment    calculator-app-appchart-64dc49c4db-tfghl            1/1     Running   0          49m
ingress-nginx      ingress-nginx-controller-7bf6cbf688-5h75n           1/1     Running   0          4d7h
jenkins            jenkins-0                                           2/2     Running   0          4d7h
kube-system        azure-ip-masq-agent-b5n7b                           1/1     Running   0          4d7h
kube-system        cloud-node-manager-5pcp8                            1/1     Running   0          4d7h
kube-system        coredns-789789675-md8qq                             1/1     Running   0          4d7h
kube-system        coredns-789789675-rmsv4                             1/1     Running   0          4d7h
kube-system        coredns-autoscaler-649b947bbd-7mczs                 1/1     Running   0          4d7h
kube-system        csi-azuredisk-node-7rdst                            3/3     Running   0          4d7h
kube-system        csi-azurefile-node-g2xmh                            3/3     Running   0          4d7h
kube-system        konnectivity-agent-65f4f6f47-dfdsj                  1/1     Running   0          4d7h
kube-system        konnectivity-agent-65f4f6f47-z97nh                  1/1     Running   0          4d7h
kube-system        kube-proxy-9zpr9                                    1/1     Running   0          4d7h
kube-system        metrics-server-5fffcb8954-gnfhk                     2/2     Running   0          4d7h
kube-system        metrics-server-5fffcb8954-v5z4h                     2/2     Running   0          4d7h
nexus              nexus-nexus-repository-manager-7c6d4dc589-pqc22     1/1     Running   0          4d7h
```

Σχήμα 4. 2: Kubernetes Περιβάλλον

Περιγραφή των Βασικών Συστημάτων:

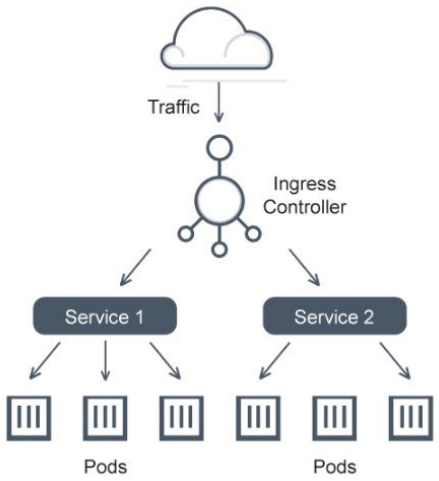
- Το **Calculator-App** αντικατοπτρίζει την ίδια την εφαρμογή η οποία έχει αναπτυχθεί στο Kubernetes Cluster σε μορφή container χρησιμοποιώντας helm chart. Για το web interface της εφαρμογής υπάρχει προσβασιμότητα μέσω της διεύθυνσης app.sarakenidis.gr.

- Το **Jenkins** είναι το CI/CD εργαλείο όπου έχει δημιουργηθεί το pipeline και εκτελεί τις κατάλληλες διαδικασίες ώστε να λάβει το κώδικα από το αποθετήριο έως και την παράδοση - ανάπτυξη αναβάθμισης της εφαρμογής (Calculator-App). Για τη διαχείριση του **Jenkins** υπάρχει το web portal του οποίου η διεύθυνση είναι jenkins.sarakenidis.gr.

- Το **Nexus** είναι το Artifact Repository το οποίο επίσης έχει στηθεί σε μορφή container και αποθηκεύει τα maven packages σε μορφή .jar. Ακολούθως, για τη διαχείριση του **Nexus** η διεύθυνση του web portal είναι η nexus.sarakenidis.gr.

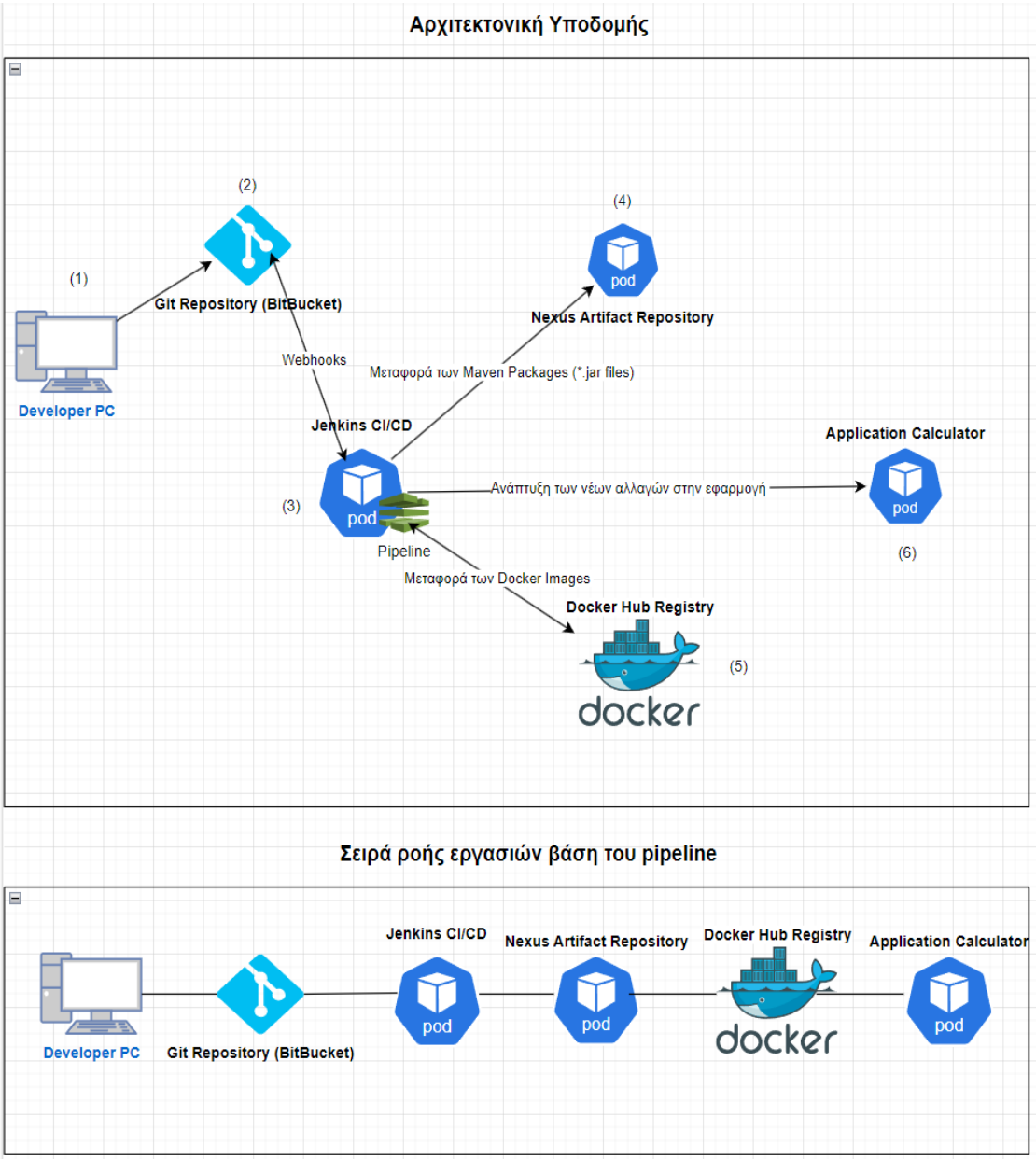
Η πρόσβαση των παραπάνω (Calculate-App, Jenkins, Nexus) στις αντίστοιχες διευθύνσεις είναι δημόσια, καθώς έχει αγοραστεί το domain sarakenidis.gr από πολύ γνωστό DNS registrar όπου έχουν δημιουργηθεί σ' αυτό τα απαραίτητα A records.

Επιπλέον, ένα από τα πολύ σημαντικά containers στο Kubernetes cluster είναι το **ingress-controller**, το οποίο επιτρέπει την επαφή με τον έξω κόσμο. Διαχειρίζεται τον δρομολογητή του δικτύου και τις ρυθμίσεις του διακομιστή προώθησης για την εισαγωγή των εισερχόμενων συνδέσεων στο cluster. Επιτρέπει τη δρομολόγηση της κίνησης που προορίζεται για την εφαρμογή σε διαφορετικούς προορισμούς μέσω μιας εισόδου, προσφέροντας ταυτόχρονα ευελιξία και ασφάλεια.



Σχήμα 4. 3: Παρουσίαση των incoming requests

Παρακάτω βρίσκεται ο σχεδιασμός υποδομής της εργασίας:



Σχήμα 4. 4: Σχεδίαση της Εφαρμογής

Ακολουθεί **λεπτομερής περιγραφή** και **εξήγηση** της ροής εργασιών εντός του Pipeline. Επιπλέον, θα δοθούν κάποια στιγμιότυπα αντιπροσωπεύοντας την υπάρχουσα υποδομή/εφαρμογή:

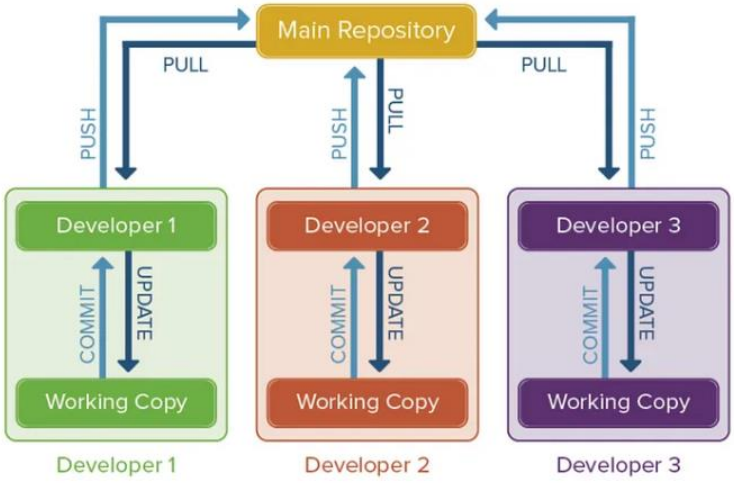
Τα παρακάτω βήματα επαναλαμβάνονται αυτόματα κάθε φορά που γίνεται commit μια αλλαγή στο κεντρικό αποθετήριο (Bitbucket).

- Ο προγραμματιστής για να μπορέσει να αναπτύσσει τον κώδικα στο κεντρικό αποθετήριο, θα πρέπει πρώτα στον υπολογιστή του χρησιμοποιώντας τη δική του πλατφόρμα ανάπτυξης κώδικα και με συγκεκριμένη εντολή στο CLI να τραβήξει το κώδικα από το κεντρικό αποθετήριο της εφαρμογής (στη συγκεκριμένη περίπτωση απ' το bitbucket) κάνοντας χρήση του εργαλείου git στον υπολογιστή του.

  Συγκεκριμένα χρησιμοποιώντας το terminal με την εντολή «git clone https://sarakenidis@bitbucket.org/infra-s/calculator.git» θα καταφέρει να κατεβάσει τοπικά το project από το αποθετήριο και να ξεκινήσει να δουλεύει τοπικά πάνω σ' αυτό.

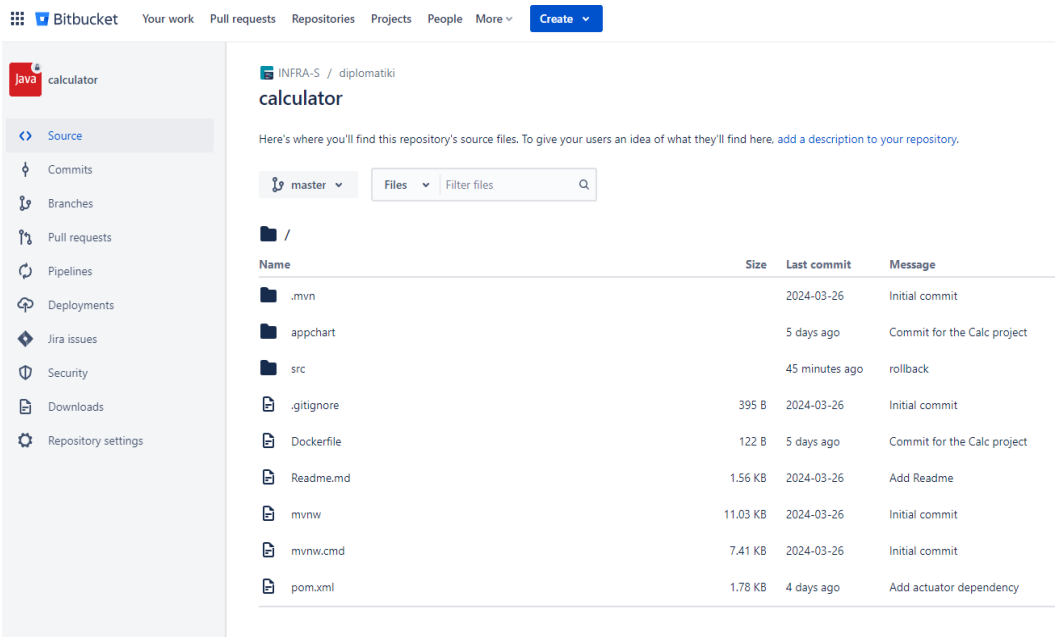
  Αφού πραγματοποιήσει νέες αλλαγές στο κώδικα τοπικά στο μηχάνημά του, έρχεται η ώρα να προσθέσει τον κώδικα στο κεντρικό αποθετήριο (bitbucket). Με τις παρακάτω εντολές προχωράει στην ενημέρωση/ανέβασμα των αλλαγών:

  a. Git status (για να ελέγξει τα αρχεία στα οποία έγιναν αλλαγές και πρόκειται να ανεβάσει).
  b. Git add . (σε περίπτωση που θέλει να προσθέσει στο repository του όλες τις αλλαγές του folder στον οποίο βρίσκεται).
  c. Git commit -m "2nd Modification on App" (Επιβεβαίωση/Εκτέλεση των αλλαγών και εσωτερικό message μεταξύ των εισαγωγικών χρησιμοποιώντας την "-m" παράμετρο).
  d. Git push (τελική εντολή για ανέβασμα των αλλαγών στο κεντρικό αποθετήριο της εφαρμογής).



Σχήμα 4. 5: Κατανεμημένο Σύστημα Ελέγχου Εκδόσεων.

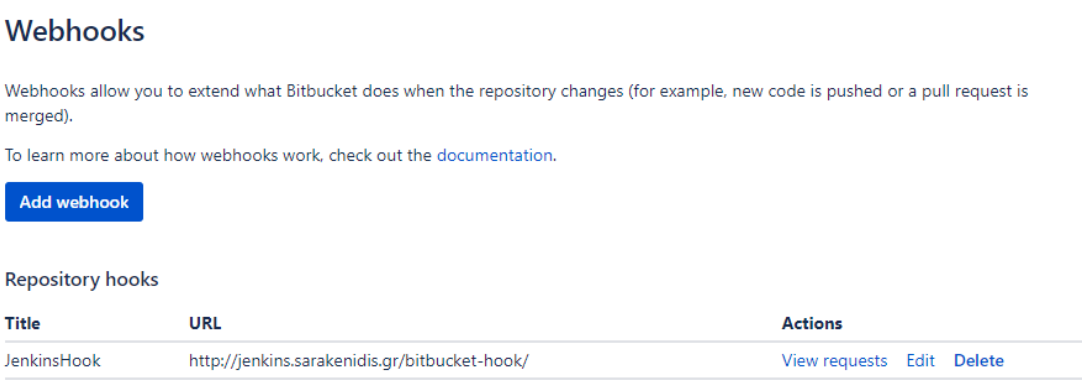
- Αφού ολοκληρωθεί το push από το developer, στη μεριά του bitbucket, στη στήλη “Last commit” του αρχείου που πραγματοποιήθηκε η αλλαγή, επιβεβαιώνει ότι η αλλαγή όντως εκτελέστηκε κάποια χρονική στιγμή. Για παράδειγμα (βλ. Σχήμα 4.6) στο φάκελο “src”, στη στήλη Message, το «rollback» έγινε πριν από 45 λεπτά, δηλαδή την ώρα που ο developer εκτέλεσε την εντολή “git push”.



Σχήμα 4. 6: BitBucket Κεντρικό Αποθετήριο Εφαρμογής

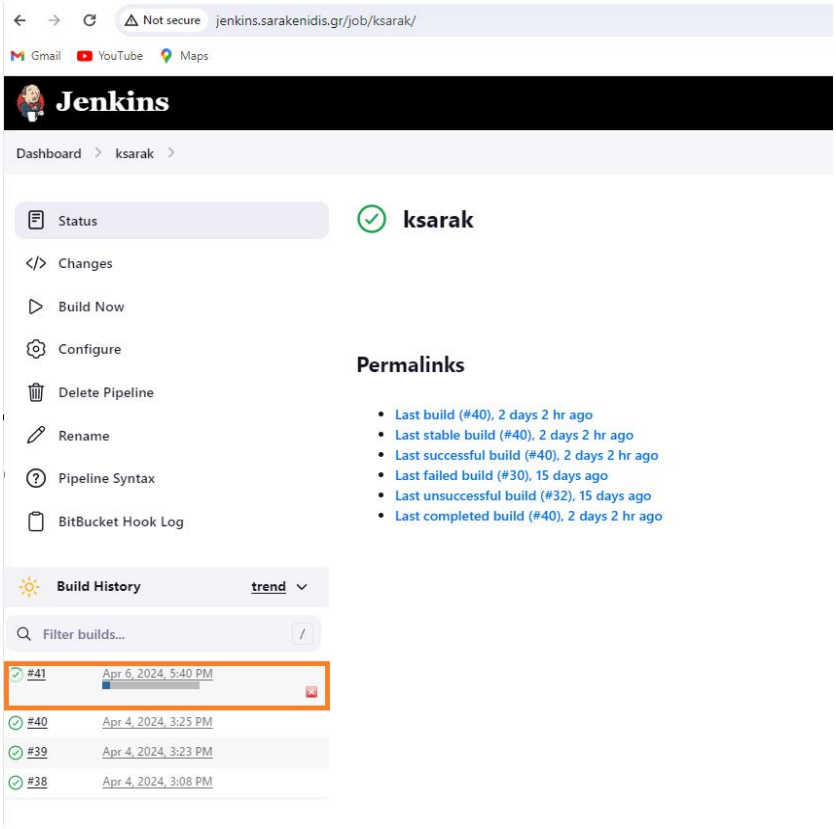
- Με την υποβολή των αλλαγών στο Bitbucket αποθετήριο, ξεκινά η διαδικασία του CI. Το Bitbucket δέχεται τις νέες αλλαγές και ενημερώνεται το σύστημα CI/CD, στην δική μας περίπτωση το Jenkins με βάση το προκαθορισμένο workflow.

  Το Jenkins ενημερώνεται για τις νέες αλλαγές στον κώδικα χρησιμοποιώντας webhooks (βλ. Σχήμα 4.7).



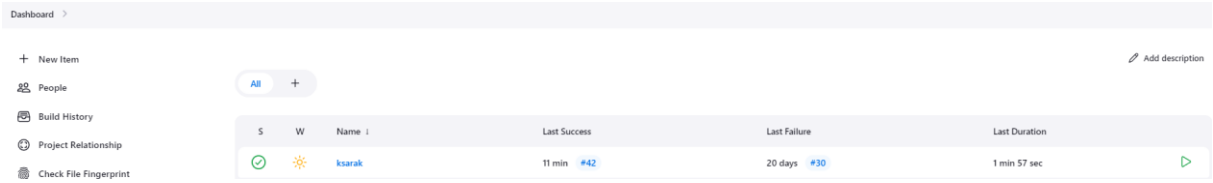
Σχήμα 4. 7: Webhook by Jenkins

- Και στη συνέχεια, εκτελείται αυτόματα η διαδικασία του pipeline



Σχήμα 4. 8: Pipeline in action

Τα επόμενα βήματα βρίσκονται εντός του pipeline (ksarak) και είναι τα εξής:



Σχήμα 4. 9: Pipeline in Jenkins

- **Pull** (τράβηγμα) του project από το BitBucket κεντρικό αποθετήριο, ώστε το jenkins να διαθέτει πρόσβαση σ' αυτό καθ' όλη τη διάρκεια του pipeline.

```
stage('Git Checkout') {
    steps {
        script {
            git branch: 'master',
                credentialsId: 'bitbucket',
                url: 'https://sarakenidis@bitbucket.org/infra-s/calculator.git'
        }
    }
}
```

Σχήμα 4. 10: Pipeline Git Checkout

Όπως φαίνεται στο παραπάνω σχήμα 4.10, γίνεται με λίγα λόγια το pull του αποθετηρίου σε επίπεδο Jenkins, ώστε να μπορέσει να εκτελέσει τα επόμενα stages χρησιμοποιώντας το συγκεκριμένο ενημερωμένο κεντρικό αποθετήριο.

Βασικά στοιχεία σε αυτό το stage είναι το **git branch**, που αναφέρεται στο branch του αποθετηρίου (repository) από το οποίο θα αποκτηθεί ο κώδικας. Επίσης, το **credentialsId**, τα οποία είναι στοιχεία σύνδεσης που θα χρειαστούν για την απόκτηση πρόσβασης του Jenkins στο BitBucket αποθετήριο. Επιπλέον, με το **url** ενημερώνουμε τη διεύθυνση της τοποθεσίας του αποθετηρίου απ' όπου θα ανακτηθεί.

- **Maven packaging**, αναλαμβάνει την αναβάθμιση και τη μεταγλώττιση του κώδικα Java. Για να λειτουργήσει αυτό θα χρειαστεί να εγκατασταθεί νωρίτερα το maven plugin. Χρησιμοποιείται για την εξαγωγή του jar εκτελέσιμου αρχείου.

```
stage('Maven Package') {
    steps {
        script {
            sh "mvn clean package"
        }
    }
}
```

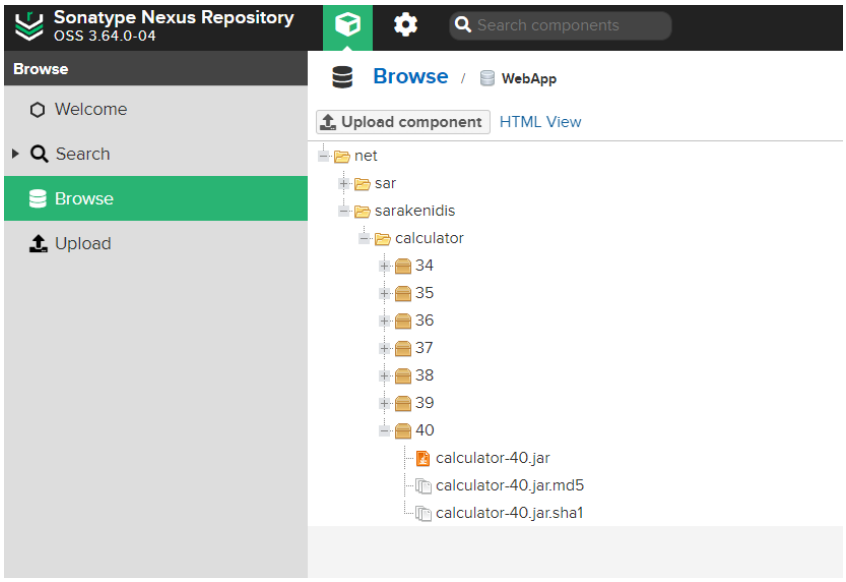
Σχήμα 4. 11: Maven packaging

- Ακολούθως εκτελείται το **Maven test**, ελέγχει το νέο κώδικα και αφού ολοκληρωθεί επιτυχώς συνεχίζει στα επόμενα βήματα.

```
stage('Maven Test') {
    steps {
        script {
            sh "mvn test"
        }
    }
}
```

Σχήμα 4. 12: Maven testing

- **Αποστολή** του **.jar αρχείου** που εξήγαγε η εντολή "mvn clean package" **στο Artifact Repository (nexus)**, ώστε να αποθηκεύουμε κάθε φορά το αρχείο σε περίπτωση που χρειαστεί να τρέξουμε απευθείας το jar και όχι το image ολόκληρο ακόμα και για ιστορικούς λόγους.



Σχήμα 4. 13: Artifact Repository for Maven Packages

Στη παραπάνω εικόνα εμφανίζονται τα jar packages που βρίσκονται στη πλευρά του Nexus (maven repository). Τα folders κάτω από το folder Calculator όπως {39}-{40}, συμβολίζουν τα builds που έχουν εκτελεστεί από το εργαλείο CI/CD.

Για την υλοποίηση του σταδίου (μεταφοράς του jar αρχείου) μέσα στο pipeline χρειάστηκαν τα παρακάτω:

```
environment {
    // This can be nexus3 or nexus2
    NEXUS_VERSION = "nexus3"
    // This can be http or https
    NEXUS_PROTOCOL = "http"
    // Where your Nexus is running
    NEXUS_URL = "nexus.sarakenidis.gr"
    // Repository where we will upload the artifact
    NEXUS_REPOSITORY = "WebApp"
    // Jenkins credential id to authenticate to Nexus OSS
    NEXUS_CREDENTIAL_ID = "nexusCred"
    ARTIFACT_VERSION = "${BUILD_NUMBER}"
}
stage("publish to nexus") {
        steps {
            script {
                // Read POM xml file using 'readMavenPom' step , this step 'readMavenPom' is included in: https://plugins.jenkins.io/pipeline-utility-steps
                pom = readMavenPom file: "pom.xml";
                // Find built artifact under target folder
                filesByGlob = findFiles(glob: "target/*.${pom.packaging}");
                // Print some info from the artifact found
                echo "${filesByGlob[0].name} ${filesByGlob[0].path} ${filesByGlob[0].directory} ${filesByGlob[0].length} ${filesByGlob[0].lastModified}"
                // Extract the path from the File found
                artifactPath = filesByGlob[0].path;
                // Assign to a boolean response verifying If the artifact name exists
                artifactExists = fileExists artifactPath;

                if(artifactExists) {
                    echo "*** File: ${artifactPath}, group: ${pom.groupId}, packaging: ${pom.packaging}, version ${pom.version}";

                    nexusArtifactUploader(
                        nexusVersion: NEXUS_VERSION,
                        protocol: NEXUS_PROTOCOL,
                        nexusUrl: NEXUS_URL,
                        groupId: pom.groupId,
                        version: ARTIFACT_VERSION,
                        repository: NEXUS_REPOSITORY,
                        credentialsId: NEXUS_CREDENTIAL_ID,
                        artifacts: [
                            // Artifact generated such as .jar, .ear and .war files.
                            [artifactId: pom.artifactId,
                            classifier: '',
                            file: artifactPath,
                            type: pom.packaging]
                        ]
                    );

                } else {
                    error "*** File: ${artifactPath}, could not be found";
                }
            }
```

Σχήμα 4. 14: Στάδιο μεταφοράς του .jar package στο Nexus repository

Όπως βλέπετε στο Σχήμα 4.14 στην αρχή του pipeline βρίσκονται οι τιμές μέσα σε environment variables για το καλώς έχει του pipeline. Σε συνέχεια, με βάση το plugin του nexus, πραγματοποιείται η μεταφορά στο αποθετήριο του Nexus.

- Build το image σύμφωνα με τις νέες αλλαγές του maven packaging και αποστολή του σε κάποιο docker repository, στη περίπτωση του συγκεκριμένου project το docker hub registry. Για να μπορέσει να κατασκευαστεί το image από τη μεριά του CI/CD που βρίσκεται ήδη μέσα στο Kubernetes cluster ως container, έγινε η χρήση του kaniko εργαλείου. Το kaniko χρησιμοποιείται για τη κατασκευή των images [22] μέσα σε περιβάλλοντα που δεν έχουν πρόσβαση στο Docker daemon, για παράδειγμα σε περιβάλλοντα CI/CD όπως το Jenkins, καθώς βρίσκεται ήδη και στο ίδιο cluster σε μορφή container.

Εφόσον ολοκληρωθεί η αποστολή του image στο docker hub registry, το image πλέον είναι διαθέσιμο για χρήση στο Kubernetes cluster ως container.

```
stage('Build with Kaniko') {
  environment {
    PATH = "/busybox:/kaniko:$PATH"
  }
  steps {
    container(name: 'kaniko', shell: '/busybox/sh') {
      sh '''#!/busybox/sh
        /kaniko/executor --dockerfile /Dockerfile --context . --destination treloki/hello-kaniko:$BUILD_NUMBER
      '''
    }
  }
}
```

Σχήμα 4. 15: Στάδιο Κατασκευής Image με τη χρήση του Kaniko

Η διαδικασία της αναπαραγωγής του image γίνεται χρησιμοποιώντας το Dockerfile που έχει δημιουργηθεί στο Git Repository. Το Dockerfile προσδιορίζει τα βήματα που απαιτούνται για τη δημιουργία του image. Αυτά τα βήματα μπορεί να περιλαμβάνουν την εγκατάσταση απαιτούμενων – απαραίτητων πακέτων, τη δημιουργία του περιβάλλοντος εκτέλεσης της εφαρμογής, την αντιγραφή του κώδικα της εφαρμογής κλπ.

Στα πλαίσια του CI/CD, το kaniko λαμβάνει το Dockerfile από το Bitbucket αποθετήριο και εκτελεί τα βήματα που έχουν οριστεί μέσα σε αυτό, χωρίς την ανάγκη να χρησιμοποιήσει το docker daemon.

Παρακάτω δίνεται το Dockerfile της εφαρμογής:

INFRA-S / diplomatiki / demo-app

Dockerfile

Here's where you'll find this repository's source files. To give your users an idea of wha

Source | master | ad1d32a | Full commit

demo-app / Dockerfile

```
1  FROM eclipse-temurin:17-jre-alpine
2  WORKDIR /opt
3  COPY target/*.jar /opt/app.jar
4  EXPOSE 8080
5  CMD ["java", "-jar", "app.jar"]
```

Σχήμα 4. 16: Dockerfile

Το παραπάνω Dockerfile, δέχεται εντολές – παραμέτρους που καθορίζει τη δημιουργία του image της εφαρμογής Calculator. Αναλύοντας τις εντολές μέσα σ' αυτό:

- **FROM**: Η εφαρμογή χρησιμοποιεί υπάρχον image με λειτουργικό Alpine Linux και JRE έκδοση 17, παρέχει ένα ελαφρύ περιβάλλον εκτέλεσης για εφαρμογές java.

- **WORKDIR**: Είναι το working directory και ορίζεται ως /opt, απ' όπου θα εκτελούνται οι εντολές της εφαρμογής μέσα στο container.

- **COPY**: Το Dockerfile αντιγράφει το αρχείο JAR που παράγεται από το στάδιο του Jenkins, στον κατάλογο /opt εντός του container και το ονομάζει app.jar.

- **EXPOSE**: Δηλώνει ότι η εφαρμογή πρέπει να είναι πρόσβασιμη μέσω της θύρας (port) 8080 του container.

- **CMD**: Ορίζει την εντολή που θα εκτελείται όταν ξεκινάει το container, δηλαδή να εκτελεί το .jar αρχείο με την εντολή «java -jar app.jar».

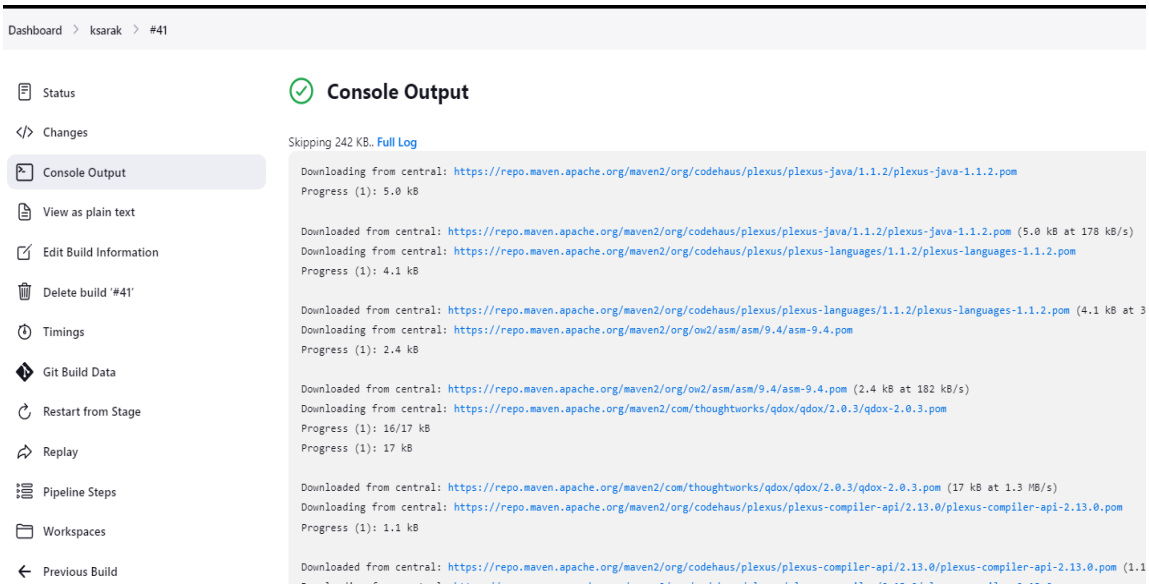
Έτσι, το dockerfile δημιουργεί ένα νέο image που μπορεί να εκτελεστεί για την εφαρμογή Calculator σε ένα ελαφρύ περιβάλλον σε μορφή του container.

- **Ανάπτυξη** (**deployment**) του image στο Kubernetes cluster ως container. Τέλος, έρχεται η ανάπτυξη του συγκεκριμένου κατασκευασμένου image μέσα στο Kubernetes cluster. Η ανάπτυξη (deployment) εκτελείται στο τελευταίο στάδιο του pipeline χρησιμοποιώντας την εντολή Helm. Η εντολή Helm είναι ένας package manager και χρησιμοποιείται για εγκατάσταση εφαρμογών σε ένα Kubernetes Cluster (βλ. Σχήμα 4.17).

```
stage('Helm Deploy') {
    steps {
      container(name: 'helm') {
        script {
          sh "helm upgrade --install calculator-app ./appchart --namespace helm-deployment --set image.tag=$BUILD_NUMBER"
        }
      }
    }
}
```

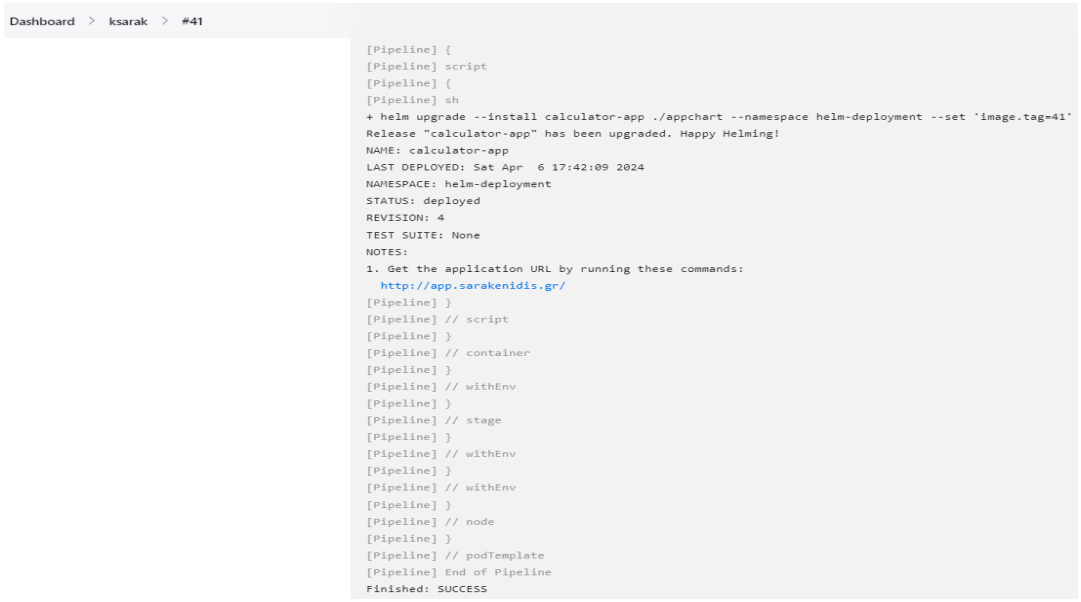
Σχήμα 4. 17: Ανάπτυξη εφαρμογής με Helm

- Αφού ολοκληρωθούν τα παραπάνω στάδια (stages) του pipeline, η εφαρμογή έχει παραδοθεί/αναπτυχθεί. Η καταγραφή των stages εμφανίζεται στο “Console Output” του Pipeline (βλ. Σχήμα 4.18).



Σχήμα 4. 18: Console Output

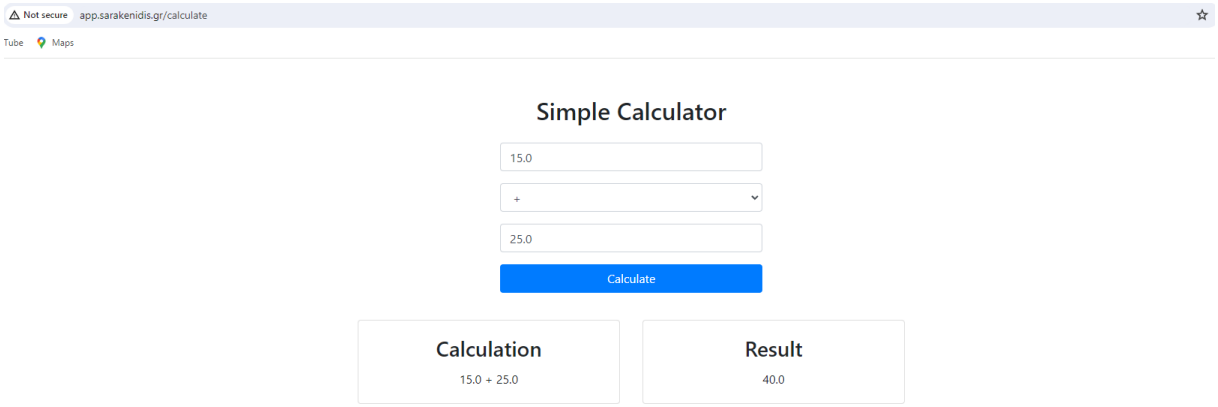
Στο κάτω μέρος του “Console Output” χρησιμοποιώντας τη μπάρα κύλισης στα δεξιά, εντοπίζεται η τελική κατάσταση του pipeline, αν ολοκληρώθηκε επιτυχώς ή αν εμφανίστηκε σφάλμα κατά τη διάρκεια εκτέλεσής του (βλ. Σχήμα 4. 19).



Σχήμα 4. 19: Κατάσταση της Console Output

Τη πρώτη φορά που θα εκτελεστεί το pipeline αν δεν υπάρχει ήδη η εφαρμογή θα δημιουργηθεί, σε διαφορετική περίπτωση θα ενημερώσει την υπάρχον - ίδια εφαρμογή. Αυτό υλοποιείται με τη βοήθεια της εντολής “helm upgrade –install” που βρίσκεται στο τελικό στάδιο του pipeline. Παρακάτω θα βρείτε το τελικό στάδιο της εφαρμογής αφού ολοκληρωθεί το pipeline επιτυχώς.

Πληκτρολογώντας στο browser app.sarakenidis.gr/calculator λαμβάνουμε το web interface της εφαρμογής:



Σχήμα 4. 20: Εφαρμογή Εργασίας

Ο παραπάνω σχεδιασμός επιτρέπει την αυτόματη και την επανάληψη της διαδικασίας ανάπτυξης, μειώνοντας τον χρόνο παράδοσης και αυξάνοντας την ποιότητα του λογισμικού. Επιπλέον, η ολοκληρωμένη συνδεσμολογία μεταξύ των εργαλείων εξασφαλίζει τη συνεχή παράδοση και την αξιοπιστία της διαδικασίας ανάπτυξης λογισμικού.

4.5 Παρουσίαση των βασικών Τεχνολογιών – Εργαλείων

Η επιτυχημένη υλοποίηση της διαδικασίας CI/CD απαιτεί τη χρήση ενός ευρέως φάσμα εργαλείων και τεχνολογιών που συνδυάζονται για την διαδικασία ανάπτυξης και παράδοσης λογισμικού. Ακολουθεί μια περιγραφή των βασικών εργαλείων και τεχνολογιών που χρησιμοποιήθηκαν και είναι κεντρικής σημασίας:

Jenkins:

- Ο Jenkins είναι ένα δωρεάν και ανοικτού κώδικα εξυπηρετητής αυτοματισμού.
- Βοηθά στον αυτοματισμό τμημάτων της ανάπτυξης λογισμικού [23] που σχετίζονται με την κατασκευή, τον έλεγχο και την ανάπτυξη, διευκολύνοντας τη συνεχή ενσωμάτωση και παράδοση.
- Πρόκειται για ένα σύστημα που βασίζεται σε εξυπηρετητές και λειτουργεί σε διακομιστές servlet όπως το Apache Tomcat.

Σχήμα 4. 21: Jenkins

Git:

- Αποθηκεύει τον κώδικα σε αποθετήρια (repositories) [24] για παρακολούθηση και διαχείριση.
- Επιτρέπει στους προγραμματιστές να δημιουργούν και να αναρτούν αλλαγές (commits) στον κώδικα .
- Παρέχει ευελιξία στο συντονισμό της εργασίας μεταξύ πολλαπλών μελών ομάδας.
- Επιτρέπει στους προγραμματιστές να αντιμετωπίζουν αντιφάσεις και να επαναφέρουν τον κώδικα σε προηγούμενα σημεία (rollbacks) με ασφάλεια.
- Παρέχει δυνατότητα συνεργασίας με άλλους προγραμματιστές μέσω κεντρικών αποθετηρίων (central repositories).

Σχήμα 4. 22: git

Bitbucket:

- Το Bitbucket είναι μια πλατφόρμα ανάπτυξης λογισμικού που χρησιμοποιείται για τη διαχείριση των αποθετηρίων κώδικα.
- Επιτρέπει τη συνεργασία μεταξύ προγραμματιστών [25] και τον έλεγχο των αλλαγών στον κώδικα.

Σχήμα 4. 23: Bitbucket

Maven:

- Το Maven είναι ένα εργαλείο διαχείρισης έργων λογισμικού [26] για την αυτόματη διαδικασία δημιουργίας και δοκιμής του κώδικα.
- Βασίζεται σε συνθήκες έργων (project object model - POM), τα οποία ορίζουν τη δομή του έργου και τις απαραίτητες διαμορφώσεις.
- Επιτρέπει την εύκολη διαχείριση της ομαδικής ανάπτυξης, επιτρέποντας στα μέλη της ομάδας να συνεργαστούν στο ίδιο έργο με ενιαίο τρόπο.
- Ενσωματώνεται εύκολα με διάφορα εργαλεία ανάπτυξης λογισμικού και συστήματα ελέγχου εκδόσεων.

Σχήμα 4. 24: Maven

Kaniko:

- Το Kaniko [22] είναι ένα εργαλείο κατασκευής εικόνων Docker που λειτουργεί από το εσωτερικό ενός container.
- Χρησιμοποιείται για τη δημιουργία εικόνων Docker χωρίς την ανάγκη πρόσβασης σε έναν Docker daemon.
- Μπορεί να ενσωματωθεί εύκολα σε περιβάλλοντα CI/CD.

Σχήμα 4. 25: Kaniko

Helm:

- Το Helm είναι ένα εργαλείο διαχείρισης πακέτων για την την εγκατάσταση και τη διαχείριση εφαρμογών σε Kubernetes [12].
- Παρέχει ένα μηχανισμό πακετοποίησης για τις εφαρμογές, γνωστό ως "charts", που περιέχουν τα απαραίτητα αρχεία διαμόρφωσης.
- Τα charts είναι ανεξάρτητα από το περιβάλλον και τον τρόπο εγκατάστασης, επιτρέποντας την εύκολη αναπαραγωγή και κατανομή των εφαρμογών.
- Με τη χρήση του Helm, η ανάπτυξη, η ενημέρωση και η διαχείριση των εφαρμογών στο Kubernetes γίνονται πιο αποτελεσματικές και ευέλικτες.

Σχήμα 4. 26: Helm

Kubernetes:

- Το Kubernetes είναι ένα ανοιχτού κώδικα σύστημα [27] διαχείρισης εφαρμογών σε ενός ή περισσότερων container clusters. Το Kubernetes είναι ένα ανοικτού κώδικα σύστημα διαχείρισης container που αναπτύχθηκε από την Google.
- Χρησιμοποιείται για τη διαχείριση και την κλιμάκωση εφαρμογών που λειτουργούν μέσα σε containers.
- Παρέχει ένα περιβάλλον που επιτρέπει την απομακρυσμένη διαχείριση και ελέγχου των εφαρμογών, από την εκκίνηση μέχρι τη διακοπή τους.
- Παρέχει λειτουργικότητες όπως τον αυτόματο εκκινούμενο επαναληπτικό έλεγχο, την ανάκτηση από αποτυχία, τη διαχείριση πόρων και την εποπτεία της κατάστασης των εφαρμογών.

Σχήμα 4. 27: Kubernetes

Nexus:

- Το Nexus Repository είναι ένα διαχειριστικό σύστημα αποθετηρίων [28] που χρησιμοποιείται για τη διαχείριση και την αποθήκευση διαφόρων τύπων αρχείων και πόρων.
- Χρησιμοποιείται κυρίως για τη διαχείριση αποθετηρίων Maven, npm, Docker και άλλων τύπων πόρων σε ένα κεντρικό μέρος.
- Παρέχει λειτουργίες όπως η ασφαλής αποθήκευση, η διαχείριση της πρόσβασης, η δυνατότητα αναζήτησης και ανακτήσεως πόρων και η υποστήριξη διαφόρων μορφών αποθετηρίων.
- Χρησιμοποιείται ευρέως στον κόσμο της ανάπτυξης λογισμικού για τη διατήρηση και την ανταλλαγή πόρων και εξαρτήσεων των εφαρμογών.

Σχήμα 4. 28: Nexus

Docker Hub Registry:

- Το Docker Hub Registry είναι ένας κεντρικός διακομιστής αποθετηρίου [29] που παρέχει αποθήκευση και διανομή εικόνων Docker.
- Χρησιμοποιείται για την αποθήκευση και την κοινοποίηση εικόνων Docker που δημιουργούνται από τους χρήστες.
- Παρέχει δυνατότητες όπως η διαχείριση των δικαιωμάτων πρόσβασης, η οργάνωση των εικόνων σε διάφορα αποθετήρια και η δυνατότητα αναζήτησης και λήψης εικόνων.

Σχήμα 4. 29: Docker Hub

- Είναι ένα από τα πιο δημοφιλή αποθετήρια για την κοινότητα των προγραμματιστών Docker και χρησιμοποιείται ευρέως για την ανάπτυξη και την κοινοποίηση εικόνων Docker.

Αυτά τα εργαλεία και οι τεχνολογίες αποτελούν τον πυρήνα της υλοποίησης μιας αυτόματης διαδικασίας CI/CD, επιτρέποντας την αποτελεσματική διαχείριση, έλεγχο και παράδοση λογισμικού.

4.6 Επίλογος

Κατά τη διάρκεια αυτού του κεφαλαίου, εξετάσαμε λεπτομερώς τη ροή εργασίας της εφαρμογής, από την αρχή της ανάπτυξης έως την τελική της κατάσταση. Αναλύσαμε τις διάφορες διαδικασίες και στάδια που απαιτούνται για την ανάπτυξη, δοκιμή, πακετάρισμα και αναπτυξιακή παράδοση της εφαρμογής.

Τέλος, αναγνωρίσαμε τη σημασία της διαδικασία και της εφαρμογής των αρχών του Continuous Integration (CI) και Continuous Delivery (CD) για την επίτευξη ενός αποτελεσματικού και αξιόπιστου περιβάλλοντος ανάπτυξης λογισμικού.

Κεφάλαιο 5ο: Αξιολόγηση

5.1 Εισαγωγή

Το κεφάλαιο αυτό εστιάζεται στην αξιολόγηση των πλεονεκτημάτων και των προκλήσεων που προκύπτουν από την εφαρμογή της συνεχούς ενσωμάτωσης και ανάπτυξης (CI/CD) στο πλαίσιο του project. Καθώς το λογισμικό αποτελεί ένα βασικό στοιχείο της σύγχρονης τεχνολογικής ανάπτυξης, η ανάγκη για αποτελεσματικές μεθόδους ανάπτυξης και παράδοσης είναι ουσιώδης. Εξετάζοντας την εφαρμογή στο πλαίσιο του project, μπορούμε να αναδείξουμε τα οφέλη που προσφέρει το CI/CD αλλά και τις προκλήσεις που πρέπει να αντιμετωπίσουμε.

5.2 Αξιολόγηση των πλεονεκτημάτων και των προκλήσεων

Η εφαρμογή CI/CD φέρνει μεγάλα πλεονεκτήματα καθώς και ορισμένες προκλήσεις που πρέπει να αντιμετωπιστούν. Ας εξετάσουμε πρώτα τα πλεονεκτήματα.

Πλεονεκτήματα εφαρμογής CI/CD:

Ένα από τα κύρια πλεονεκτήματα είναι η αύξηση της ταχύτητας και της συχνότητας των αναπτύξεων. Η διαδικασία ενσωμάτωσης και παράδοσης/ανάπτυξης επιτρέπει στην ομάδα ανάπτυξης να δημιουργεί, να ελέγχει και να κυκλοφορεί νέες εκδόσεις του λογισμικού με συνεχή ρυθμό. Αυτό μειώνει τον χρόνο που απαιτείται για την εκτέλεση των διαδικασιών χειρωνακτικά και αυξάνει την αποτελεσματικότητα της ανάπτυξης λογισμικού.

Ένα άλλο σημαντικό πλεονέκτημα είναι η βελτίωση της ποιότητας του λογισμικού. Με την εφαρμογή του CI/CD, οι έλεγχοι μπορούν να εκτελούνται αυτόματα σε κάθε αλλαγή κώδικα. Αυτό εξασφαλίζει ότι το λογισμικό είναι πάντα σε μια σταθερή κατάσταση, μειώνοντας τον κίνδυνο εμφάνισης σφαλμάτων στην παραγωγή.

Σχήμα 5. 1: Πλεονεκτήματα CI/CD

Επιπλέον, παρέχει μεγαλύτερη διαφάνεια και αξιοπιστία στη διαδικασία ανάπτυξης. Η διαδικασία ενσωμάτωσης και ανάπτυξης δημιουργεί ένα αξιόπιστο ιστορικό καταγραφής όλων των αλλαγών και των αποτελεσμάτων, ενώ επίσης διευκολύνει την παρακολούθηση της προόδου του έργου από τα μέλη της ομάδας και τους ενδιαφερόμενους.

Πιθανές προκλήσεις εφαρμογής CI/CD:

Παράλληλα, όμως, η εφαρμογή CI/CD μπορεί να αντιμετωπίσει και ορισμένες προκλήσεις. Μία από αυτές τις προκλήσεις είναι η ανάγκη για μεγάλες αλλαγές στις διαδικασίες ανάπτυξης και παράδοσης λογισμικού, καθώς και η ενσωμάτωσή τους στην δομή. Η αλλαγή αυτή μπορεί να αντιμετωπιστεί αρνητικά από ορισμένα μέλη της ομάδας που ενδεχομένως να αντισταθούν στην αλλαγή ή να αντιμετωπίσουν δυσκολίες στην προσαρμογή τους. Επιπλέον, η εφαρμογή του CI/CD μπορεί να απαιτήσει σημαντικές επενδύσεις σε υλικό και λογισμικό, καθώς και σε εκπαίδευση του προσωπικού για την κατανόηση και την αξιοποίηση των νέων εργαλείων και μεθόδων.

Οι προκλήσεις που προκύπτουν από την εφαρμογή CI/CD πρέπει να αντιμετωπιστούν με προσοχή και κατάλληλη στρατηγική. Μεγάλοι οργανισμοί που χρησιμοποιούν αυτή τη τεχνολογία πρέπει να επενδύουν σε εκπαίδευση και επικοινωνία για τη διασφάλιση της αποδοχής και της συμμετοχής όλων των μελών της ομάδας. Επιπλέον, η σωστή διαχείριση των αλλαγών και η σταδιακή υλοποίηση των νέων διαδικασιών μπορεί να βοηθήσει στη μείωση της αντίστασης και της πιθανής αρνητικής επίδρασης στην οργάνωση.

Ταυτόχρονα, η επίλυση τεχνικών προκλήσεων, όπως η βελτιστοποίηση της απόδοσης των εργαλείων CI/CD και η ενσωμάτωσή τους στις υπάρχουσες διαδικασίες ανάπτυξης λογισμικού, είναι εξίσου σημαντική. Με τη σωστή διαχείριση των προκλήσεων και την αποτελεσματική υλοποίηση των βελτιωμένων διαδικασιών CI/CD, οι οργανισμοί μπορούν να επωφεληθούν από τα πλεονεκτήματα της συνεχούς ενσωμάτωσης και παράδοσης λογισμικού για την ανάπτυξη και την παραγωγικότητά τους.

5.3 Ανάλυση των πιθανών προβλημάτων και πιθανές λύσεις

Κατά την εφαρμογή του CI/CD, μπορούν να προκύψουν διάφορα προβλήματα που επηρεάζουν την αποτελεσματικότητα και την αξιοπιστία της διαδικασίας ανάπτυξης λογισμικού. Κάποια από αυτά τα προβλήματα περιλαμβάνουν:

Απώλεια δεδομένων: Κατά τη διάρκεια της μεταφοράς ή επεξεργασίας των δεδομένων, ενδέχεται να προκύψουν σφάλματα που οδηγούν σε απώλεια πληροφοριών. Μια λύση είναι η χρήση μέτρων ασφαλείας και τον έλεγχο της ακεραιότητας [30] των δεδομένων κατά τη μεταφορά.

Συμβατότητα εργαλείων: Η συμβατότητα μεταξύ διαφορετικών εργαλείων CI/CD μπορεί να οδηγήσει σε προβλήματα ενσωμάτωσης. Μια λύση είναι η επιλογή και η προσαρμογή εργαλείων που είναι συμβατά μεταξύ τους ή η ανάπτυξη προσαρμοσμένων λύσεων ολοκληρωμένου CI/CD.

Ασφάλεια: Η ανεπαρκής προστασία των δεδομένων και των εφαρμογών κατά τη διάρκεια της διαδικασίας CI/CD μπορεί να εκθέσει το σύστημα σε κινδύνους ασφαλείας. Λύσεις περιλαμβάνουν την υιοθέτηση πρακτικών ασφαλείας, την εκπαίδευση του προσωπικού και τη χρήση εργαλείων διαχείρισης της ασφαλείας.

Αυξημένοι χρόνοι επαναφοράς: Σε περίπτωση σφάλματος κατά τη διάρκεια της διαδικασίας CI/CD, ο χρόνος ανάκαμψης μπορεί να είναι εκτεταμένος. Μια λύση είναι οι μηχανισμοί ανίχνευσης σφαλμάτων και η εκτέλεση πρακτικών επαναφοράς που είναι ταχείες και αξιόπιστες.

Διαχείριση αλλαγών: Οι συνεχείς αλλαγές στο λογισμικό μπορούν να δημιουργήσουν προβλήματα διαχείρισης και παρακολούθησης. Μια λύση είναι η εφαρμογή κατάλληλων διαδικασιών ελέγχου αλλαγών και η δημιουργία προσεκτικών σχεδίων διαχείρισης των αλλαγών.

Μέσω της ανάλυσης των παραπάνω προβλημάτων και των προτεινόμενων λύσεων, μπορούν να διαμορφωθούν στρατηγικές για τη βελτίωση της διαδικασίας ανάπτυξης λογισμικού με τη χρήση της CI/CD. Η αντιμετώπιση αυτών των προβλημάτων είναι κρίσιμη για την επίτευξη αποτελεσματικής και αξιόπιστης διαδικασίας ανάπτυξης και παράδοσης λογισμικού.

5.4 Επίλογος

Καθώς ολοκληρώνεται η αξιολόγηση των πλεονεκτημάτων και των προκλήσεων εφαρμογής του Continuous Integration/ Continuous Deployment (CI/CD), αποκαλύπτεται μια πληθώρα οφελών και δυσκολιών που απαιτούν προσεκτική ανάλυση και διαχείριση.

Από τη μία πλευρά, η εφαρμογή του CI/CD μας δίνει σημαντικά οφέλη όπως η αυξημένη ποιότητα του λογισμικού, η επιτάχυνση του κύκλου ζωής ανάπτυξης και η μείωση του ρίσκου αποτυχίας. Η διαδικασία ενσωμάτωσης και ανάπτυξης ενισχύει τη διαφάνεια και την αξιοπιστία της διαδικασίας ανάπτυξης λογισμικού.

Από την άλλη πλευρά, αναδύονται και προκλήσεις όπως η ανάγκη για κατάλληλη-απαιτητική εκπαίδευση του προσωπικού, η αντιμετώπιση προβλημάτων συμβατότητας εργαλείων και η διαχείριση της ασφάλειας. Ωστόσο, με κατάλληλη προετοιμασία και διαχείριση των προκλήσεων, το CI/CD μπορεί να προσφέρει σημαντικά οφέλη στη διαδικασία ανάπτυξης λογισμικού και στην επίτευξη των στόχων του έργου.

Τέλος, η ανάλυση των πιθανών προβλημάτων και των πιθανών λύσεων αποτελεί σημαντικό στάδιο για την επιτυχή εφαρμογή του CI/CD. Με την αντιμετώπιση των προκλήσεων και την εφαρμογή κατάλληλων λύσεων, η διαδικασία CI/CD μπορεί να ενισχυθεί και να επιτύχει τους στόχους του προγράμματος.

Κεφάλαιο 6ο: Συμπεράσματα και Προτάσεις Βελτίωσης

6.1 Εισαγωγή

Το κεφάλαιο αυτό αποτελεί το κλείσιμο της πτυχιακής εργασίας και συνοψίζει τα κύρια συμπεράσματα που προέκυψαν από την ανάλυση των πλεονεκτημάτων και προκλήσεων της εφαρμογής της CI/CD στο πλαίσιο του έργου. Επιπλέον, παρέχονται προτάσεις για τη βελτίωση της διαδικασίας CI/CD και την αντιμετώπιση πιθανών προβλημάτων που προέκυψαν κατά τη διάρκεια της μελέτης. Αναλύονται οι πιθανές επεκτάσεις και μελλοντικές κατευθύνσεις για την ενίσχυση της διαδικασίας CI/CD και τη βελτίωση της αποτελεσματικότητας του συστήματος ανάπτυξης λογισμικού.

6.2 Συμπεράσματα

Στο πλαίσιο αυτής της εργασίας, εξετάσαμε την εφαρμογή της διαδικασίας CI/CD. Μέσω της ανάλυσης των πλεονεκτημάτων και των προκλήσεων που προέκυψαν κατά τη διάρκεια της μελέτης, καταλήξαμε σε συγκεκριμένα συμπεράσματα που αξίζει να επισημανθούν.

Η εφαρμογή της διαδικασίας CI/CD επέφερε σημαντικά οφέλη στην ανάπτυξη λογισμικού, συμβάλλοντας στην επιτάχυνση του κύκλου ανάπτυξης και την αύξηση της ποιότητας του κώδικα.

Συνεπώς, αποδεικνύεται ότι η εφαρμογή CI/CD μπορεί να είναι υψίστης σημασίας για την ανάπτυξη ενός ποιοτικού - ολοκληρωμένου λογισμικού, εφόσον γίνει με σωστό τρόπο και λαμβάνοντας υπόψη τις ιδιαιτερότητες κάθε έργου. Μελλοντικές εφαρμογές της διαδικασίας αναμένεται να επιφέρουν περαιτέρω βελτιώσεις στην ανάπτυξη λογισμικού και να συμβάλλουν στην επίτευξη ακόμη μεγαλύτερης αποτελεσματικότητας και αξιοπιστίας στη διαδικασία ανάπτυξης.

6.2.1 Ανασκόπηση των Κυριότερων Ευρημάτων

Κατά τη διάρκεια της μελέτης, παρατηρήθηκε ότι η εφαρμογή της διαδικασίας CI/CD συνεισέφερε σημαντικά στη βελτίωση της απόδοσης της ομάδας ανάπτυξης. Η διαδικασία ενσωμάτωσης και ανάπτυξης επέτρεψε την ταχύτερη ενσωμάτωση των αλλαγών στον κώδικα, ενώ παράλληλα διασφάλιζε την ποιότητα και τη συνοχή του λογισμικού. Η συνεχής ενσωμάτωση και ανάπτυξη επέτρεψε επίσης την άμεση ανίχνευση και διόρθωση σφαλμάτων, μειώνοντας έτσι τον χρόνο παράδοσης και το κόστος ανάπτυξης.

6.2.2 Μελλοντικές Εφαρμογές και Αναπτύξεις

Με βάση την εξέλιξη της τεχνολογίας και των αναγκών της αγοράς, παρουσιάζονται διάφορες προοπτικές για τη βελτίωση και την εξέλιξη της διαδικασίας CI/CD:

- **Ενσωμάτωση Τεχνολογιών Τεχνητής Νοημοσύνης (ΤΝ)**: Η εφαρμογή τεχνολογιών όπως η μηχανική μάθηση και οι αλγόριθμοι ΤΝ μπορεί να βελτιώσει την ανίχνευση σφαλμάτων και τις διαδικασίες στο πλαίσιο του CI/CD.

- **Διαδικασίες Διαχείρισης Περιβάλλοντος (DevOps)**: Η συνεχής εξέλιξη της διαδικασίας DevOps συμβάλλει στην διαδικασία ανάπτυξης, δοκιμής και παράδοσης λογισμικού, βελτιώνοντας την απόδοση και την ευελιξία των ομάδων ανάπτυξης.

- **Ενσωμάτωση Εργαλείων Ανάλυσης Ποιότητας Κώδικα**: Η χρήση εργαλείων ανάλυσης κώδικα μπορεί να βοηθήσει στην πρόληψη σφαλμάτων και τη βελτίωση της ποιότητας του κώδικα, προσθέτοντας αξία στη διαδικασία CI/CD.

- **Εφαρμογή Καλύτερων Πρακτικών Ασφάλειας**: Η ανάπτυξη και η εφαρμογή καλύτερων πρακτικών ασφάλειας στο πλαίσιο του CI/CD αποτελεί προτεραιότητα, με στόχο την προστασία των δεδομένων και την αποτροπή επιθέσεων.

Με την εφαρμογή αυτών των μελλοντικών εξελίξεων, η διαδικασία CI/CD μπορεί να ενισχυθεί περαιτέρω, προσφέροντας ακόμη μεγαλύτερη ευελιξία, απόδοση και ποιότητα στην ανάπτυξη λογισμικού.

6.3 Προτάσεις Βελτίωσης

Παρουσίαση των προτάσεων για τη βελτίωση της διαδικασίας CI/CD και τη βελτίωση της απόδοσης των ομάδων ανάπτυξης λογισμικού. Οι προτάσεις αυτές προέρχονται από την ανάλυση των πλεονεκτημάτων και των προκλήσεων που παρουσιάστηκαν στο πλαίσιο της παρούσας έρευνας.

- **Ανάπτυξη Εξειδικευμένων Εργαλείων CI/CD**: Η ανάπτυξη και η ενσωμάτωση εξειδικευμένων εργαλείων για τη διαδικασία CI/CD μπορεί να βελτιώσει την απόδοση και την αξιοπιστία της διαδικασίας.

- **Εκπαίδευση και Ενημέρωση του Προσωπικού**: Η συνεχή εκπαίδευση και ενημέρωση του προσωπικού σχετικά με τις βέλτιστες πρακτικές και τις νέες τεχνολογίες στον τομέα CI/CD είναι ουσιώδης για τη βελτίωση της απόδοσης.

- **Ενίσχυση της Ασφάλειας**: Η ενίσχυση των μέτρων ασφαλείας στο πλαίσιο της διαδικασίας CI/CD είναι κρίσιμη για την προστασία των δεδομένων και την αποτροπή πιθανών επιθέσεων.

- **Βελτίωση Ενσωμάτωσης Τεχνολογιών**: Η συνεχής ενσωμάτωση νέων τεχνολογιών και πρακτικών στο πλαίσιο της διαδικασίας CI/CD μπορεί να βελτιώσει την αποδοτικότητα και την ποιότητα της διαδικασίας.

Με την εφαρμογή των παραπάνω προτάσεων, είναι δυνατό να ενισχυθεί η διαδικασία CI/CD και να επιτευχθεί μια πιο αποτελεσματική και αξιόπιστη διαδικασία ανάπτυξης λογισμικού.

6.3.1 Αντιμετώπιση Ενδεχομένων Προβλημάτων

Σε αυτήν την υποενότητα παρουσιάζουμε μια σειρά από προτάσεις για την αντιμετώπιση ενδεχόμενων προβλημάτων που μπορεί να προκύψουν κατά την εφαρμογή της διαδικασίας CI/CD. Αυτές οι προτάσεις βασίζονται σε κοινά προβλήματα που εντοπίζονται κατά την εφαρμογή της διαδικασίας CI/CD και στρέφονται στην αντιμετώπισή τους με αποτελεσματικό τρόπο.

- **Εκπαίδευση και Επαγγελματική Ανάπτυξη**: Η εκπαίδευση και η επαγγελματική ανάπτυξη του προσωπικού που εμπλέκεται στην υλοποίηση της διαδικασίας CI/CD είναι ουσιώδεις. Η παροχή εκπαίδευσης σχετικά με τη χρήση των εργαλείων, των μεθόδων και των βέλτιστων πρακτικών μπορεί να μειώσει τον αριθμό των προβλημάτων και να αυξήσει την αποτελεσματικότητα της διαδικασίας.

- **Συνεχής Παρακολούθηση και Ενημέρωση**: Η δημιουργία ενός συστήματος συνεχούς παρακολούθησης και ενημέρωσης για τυχόν προβλήματα που προκύπτουν κατά τη διάρκεια της διαδικασίας CI/CD είναι απαραίτητη. Αυτό μπορεί να επιτευχθεί μέσω της εφαρμογής συστημάτων ειδοποίησης και παρακολούθησης για άμεση αντίδραση σε πιθανά προβλήματα.

- **Συνεχής Βελτίωση**: Η διαδικασία CI/CD απαιτεί συνεχή βελτίωση και προσαρμογή. Η συλλογή στατιστικών δεδομένων και η ανάλυση των αποτελεσμάτων μπορεί να βοηθήσει στον εντοπισμό περαιτέρω περιθωρίων βελτίωσης και την υλοποίηση αντίστοιχων μέτρων.

- **Οργάνωση Καθηκόντων**: Η οργάνωση καθηκόντων μπορεί να μειώσει τον ανθρώπινο παράγοντα στη διαδικασία και να μειώσει την πιθανότητα σφαλμάτων.

Με την υλοποίηση αυτών των προτάσεων, μπορεί να επιτευχθεί μια πιο ομαλή και αποδοτική εφαρμογή της διαδικασίας CI/CD, μείωση του χρόνου ανάπτυξης λογισμικού και αύξηση της ποιότητας του τελικού προϊόντος.

6.3.2 Πιθανές Επεκτάσεις και Μελλοντικές Κατευθύνσεις

Μελλοντικές κατευθύνσεις που μπορούν να ακολουθήσουν σε σχέση με την εφαρμογή της διαδικασίας CI/CD:

- **Επέκταση σε Άλλα Έργα:** Η επιτυχής υλοποίηση της διαδικασίας CI/CD σε κάποιο έργο μπορεί να ωθήσει την ομάδα να επεκτείνει τη χρήση της σε άλλα έργα. Με την ανάπτυξη και τη βελτίωση της διαδικασίας σε διαφορετικούς τομείς της εταιρείας, μπορεί να επιτευχθεί μεγαλύτερη αποδοτικότητα και ποιότητα στην παράδοση του λογισμικού.

- **Ενσωμάτωση Νέων Τεχνολογιών**: Η συνεχής εξέλιξη των τεχνολογιών στον χώρο του CI/CD μπορεί να οδηγήσει στην ενσωμάτωση νέων εργαλείων και μεθόδων. Η αξιολόγηση και η υιοθέτηση αυτών των νέων τεχνολογιών μπορεί να βελτιώσει την αποτελεσματικότητα και την απόδοση της διαδικασίας.

- **Τεχνητή Νοημοσύνη**: Η ενσωμάτωση της τεχνητής νοημοσύνης μπορεί να βελτιώσει τη διαδικασία CI/CD περαιτέρω. Η ανάπτυξη εργαλείων που μπορούν να εκτελούν αυτόματα εργασίες όπως ο εντοπισμός σφαλμάτων και η αυτόματη επίλυσή τους μπορεί να μειώσει τον χρόνο παράδοσης και να αυξήσει την ποιότητα του λογισμικού.

6.4 Επίλογος

Στο σύνολο αυτού του κεφαλαίου, εξετάσαμε την εφαρμογή της συνεχούς ενσωμάτωσης και παράδοσης (CI/CD) στο πλάισιο του έργου. Μέσα από την ανάλυση των πλεονεκτημάτων και των προκλήσεων της εφαρμογής CI/CD, καθώς και την ανάλυση των πιθανών προβλημάτων και των πιθανών λύσεων, προέκυψε μια ολοκληρωμένη εικόνα του πώς η εφαρμογή αυτή μπορεί να συμβάλει στη βελτίωση της διαδικασίας παράδοσης και ανάπτυξης λογισμικού.

Αν και η εφαρμογή της CI/CD προσφέρει πολλά οφέλη, εντούτοις υπάρχουν και προκλήσεις που πρέπει να αντιμετωπιστούν. Με την κατάλληλη στρατηγική και τη συνεχή παρακολούθηση και προσαρμογή, ωστόσο, είναι δυνατόν να αξιοποιηθεί πλήρως η δύναμη CI/CD για τη βελτίωση της απόδοσης και της ποιότητας του λογισμικού.

Τέλος, με τη συνεχή έρευνα και ανάπτυξη, είναι πιθανό να δούμε περαιτέρω εξελίξεις που θα ενισχύσουν την αποτελεσματικότητα και την ευελιξία των διαδικασιών ανάπτυξης και παράδοσης.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] S. Newman, Building Microservices: Designing Fine-Grained Systems, O'Reilly Media, 2015.
[2] https://www.thesunflowerlab.com/microservice-architecture-benefits-business-value/.
[3] https://bigblue.academy/gr/ti-einai-to-docker.
[4] https://www.docker.com/resources/what-container/.
[5] https://docs.docker.com/engine/.
[6] https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/.
[7] https://docs.docker.com/develop/develop-images/dockerfile_best-practices/.
[8] https://docs.docker.com/registry/.
[9] https://www.linkedin.com/pulse/introduction-kubernetes-container-orchestration-platform-miqbal/.
[10] https://www.padok.fr/en/blog/kubernetes-essentials-components-pods-services.
[11] https://kubernetes.io/docs/concepts/overview/components/.
[12] https://medium.com/prodopsio/a-6-minute-introduction-to-helm-ab5949bf425.
[13] J. &. F. D. Humble, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, 2010.
[14] https://www.plusserver.com/blog/was-bedeutet-ci-cd-in-der-entwicklung/.
[15] https://martinfowler.com/articles/continuousIntegration.html.
[16] https://www.lambdatest.com/learning-hub/cicd-testing.
[17] https://www.linkedin.com/pulse/how-kubernetes-cicd-simplifies-your-devops-pipeline-kanojia/.
[18] G. D. P. W. J. &. H. J. Kim, The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations, 2016.
[19] https://www.techtarget.com/searchsoftwarequality/CI-CD-pipelines-explained-Everything-you-need-to-know.
[20] https://www.linkedin.com/pulse/how-kubernetes-cicd-simplifies-your-devops-pipeline-kanojia/.
[21] P. D. J. W. κ. J. H. Gene Kim, The DevOps Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations, 2016.
[22] D. F. Jez Humble, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation.
[23] https://github.com/GoogleContainerTools/kaniko.
[24] https://www.jenkins.io/doc/tutorials/.
[25] https://el.wikipedia.org/wiki/Git_(%CE%BB%CE%BF%CE%B3%CE%B9%CF%83%CE%BC%CE%B9%CE%BA%CF%8C).
[26] https://bitbucket.org/product/guides/getting-started/overview#a-brief-overview-of-bitbucket.
[27] https://www.simplilearn.com/tutorials/maven-tutorial/what-is-maven.
[28] https://kubernetes.io/docs/concepts/overview/.
[29] https://medium.com/linux-shots/helm-chart-repository-on-sonatype-nexus-oss-fcf6f7c7498e.
[30] https://www.docker.com/products/docker-hub/
]

[31] S. M. A. G. Paul M. Duvall, Continuous Integration: Improving Software Quality and Reducing Risk.