Building Scalable Applications using Docker and Kubernetes

ΓΑΛΑΝΗΣ ΑΧΙΛΛΕΑΣ ΑΛΕΞΑΝΔΡΟΣ ΒΑΣΙΛΕΙΟΣ - 02941 and ΓΑΛΑΝΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ ΟΡΕΣΤΗΣ ΒΑΣΙΛΕΙΟΣ - 03074

Information
Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Πανεπιστήμιο Θεσσαλίας, Βόλος
{acgalanis, kogalanis}@e-ce.uth.gr

Περιεχόμενα

1 Τι είναι το Docker

– **Επισκόπηση του Docker**

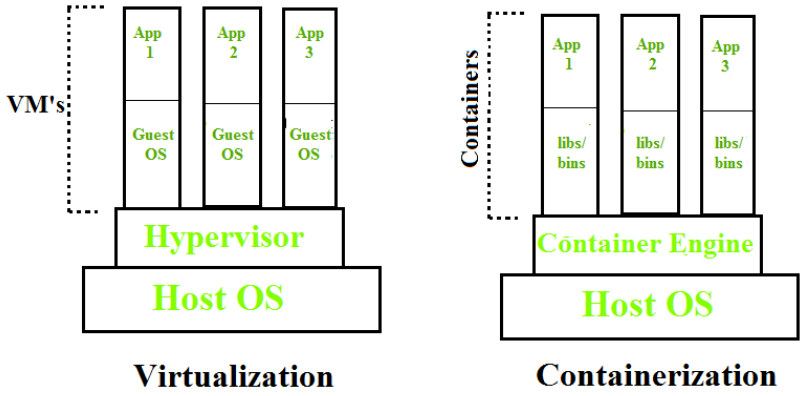
Το Docker είναι μια πλατφόρμα δοχειοποίησης (containerization), γραμμένη σε Go, η οποία συσκευάζει την εφαρμογή σας και όλες τις απαιτούμενες εξαρτήσεις της μαζί σε μορφή ενός docker container. Αποτελεί μια υπηρεσία που έχει σχεδιαστεί για να λύσει τις πολλές προκλήσεις που δημιουργούνται από την αυξανόμενη τάση του DevOps. Το DevOps είναι ένας συνδυασμός πρακτικών ανάπτυξης (Dev) και λειτουργίας λογισμικου (Ops) που στοχεύει στον συνδυασμό τους, επικεντρώντας στη συνεργασία, την αυτοματοποίηση και την συνεχή ενσωμάτωση για τη βελτίωση της ταχύτητας και της ποιότητας της ανάπτυξης λογισμικού. Χρησιμοποιώντας τα **δοχεία (Containers)**, το Docker απλοποιεί τη διαδικασία ανάπτυξης και λειτουργίας εφαρμογών.

– **Containers**

Τα containers είναι ένας κύριος λόγος για τον οποίο το Docker είναι τόσο ελκυστικό για τους σύγχρονους προγραμματιστές. Ενώ ο τρέχων βιομηχανικός προτεινόμενος τρόπος περιλαμβάνει τη χρήση Εικονικών Μηχανών (VMs) για την εκτέλεση εφαρμογών σε επίπεδο λογισμικού, αυτές(VMs) λειτουργούν εντός ενός λειτουργικού συστήματος επισκέπτη σε εικονικό υλικό, υποστηριζόμενο από το λειτουργικό σύστημα του διακομιστή. Αυτή η μέθοδος, ωστόσο, επιφέρει σημαντικό υπολογιστικό κόστος λόγω της εικονικοποίησης του υλικού για το λειτουργικό σύστημα του επισκέπτη.
Αντίθετα, τα containers υιοθετούν μια διαφορετική προσέγγιση. Δημιουργούν έναν χώρο απομόνωσης όπου ενθυλακώνουν την εφαρμογή σας και τις απαιτήσεις της, συμπεριλαμβανομένου ενός μέρους του λειτουργικού συστήματος, του κώδικα της εφαρμογής, του runtime, των εργαλείων του συστήματος και των βιβλιοθηκών του συστήματος. Αυτά τα containers λειτουργούν στον κοινό πυρήνα του λειτουργικού συστήματος της μηχανής που το εδρεύει, επιτρέποντας την εκτέλεση ενός ή περισσότερων διεργασιών εντός κάθε container. Αυτή η προσέγγιση εξαλείφει την

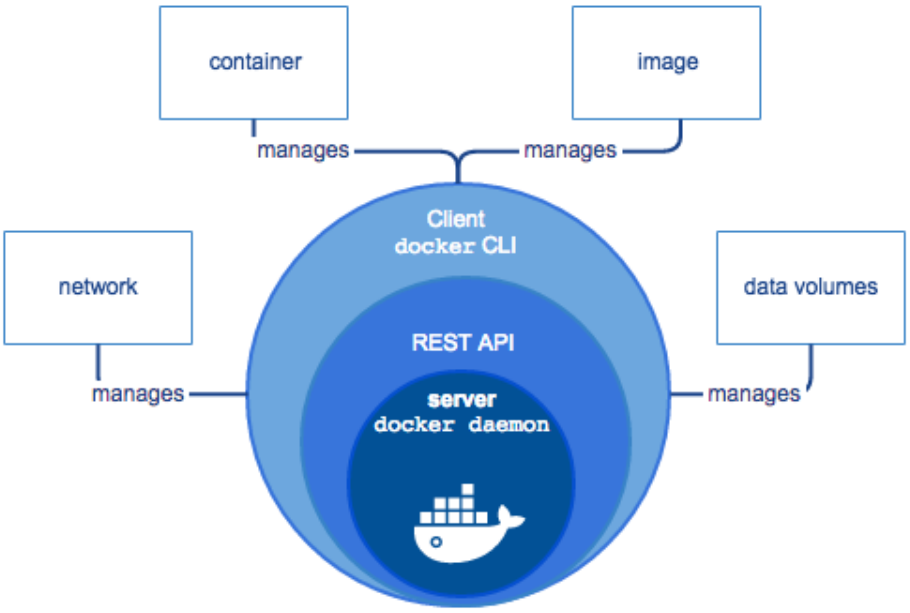
ανάγκη για προ-διάθεση RAM, καθώς η μνήμη διατίθεται δυναμικά κατά τη δημιουργία του container, αντίθετα με τις VMs που απαιτούν προκαθορισμένη.
Το Containerization προσφέρει καλύτερη χρήση πόρων και ταχύτερο χρόνο εκκίνησης σε σύγκριση με τις εικονικές μηχανές, σηματοδοτώντας μια σημαντική πρόοδο στην τεχνολογία εικονικοποίησης. Τα δοχεία παρέχουν σχεδόν το ίδιο επίπεδο απομόνωσης με τις εικονικές μηχανές και απαιτούν πολύ μικρότερο ποσοστό της υπολογιστικής ισχύος.

Docker vs VMs	Dockers	Virtual Machines
Boot time	Dockers can boot in seconds	In often takes minutes for VMs to boot
Execution	Makes use of execution engine	Makes use of hypervisor
Memory	More memory efficient as no space needed to virtualize	Less memory efficient as the entire OS needs to be loaded before startimg the service
Isolation	No provision for isolation of systems and hence are more prone to adversities	Efficient isolation mechanism and hence interference possibility is less
Ease of deployment	Deploying through dockers is extremely easy as only one image, containerized, can be used across different operating systems	Deploying in virtual machines is a comparatively lengthy process where separate instancesare responsible for the execution
Ease of usage	Dockers have comparatively complex usage mechanism which consists of both third party and docker managed tools	The tools associated with a VM are comparatively easier to use and simpler to work with



– **Βασικά στοιχεία του Docker**

Τα Βασικά στοιχεία του Docker είναι τα ακόλουθα: **Μηχανή Docker (Docker Engine), Εικόνες Docker (Docker Images), Δοχεία Docker (Docker Containers).**



- **Μηχανή Docker (Docker Engine)** είναι ένα από τα βασικά στοιχεία του Docker. Είναι υπεύθυνο για τη συνολική λειτουργία της πλατφόρμας Docker. Είναι μια εφαρμογή βασισμένη σε αρχιτεκτονική πελάτη-διακομιστή(client-server) και αποτελείται από 3 κύρια στοιχεία:

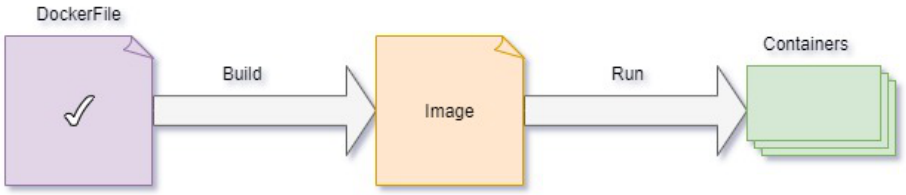
  * **Διακομιστής (Server):** Ο διακομιστής (Server) τρέχει ένα daemon γνωστό ως dockerd (Docker Daemon), ο οποίος δεν είναι τίποτα άλλο από μία διεργασία. Ακούει μόνο αιτήματα API του Docker και είναι υπεύθυνος για τη δημιουργία και διαχείριση των Εικόνων Docker (Docker Images), Δοχείων Docker (Docker Containers), Τόμων Docker (Docker Volumes) και του Δικτύου Docker (Docker Ntework) στην πλατφόρμα. Επίσης επικοινωνεί με άλλα daemon για τη διαχείριση των υπηρεσιών Docker.

  * **REST API:** Το REST API ορίζει τις μεθόδους μέσω των οποίων οι εφαρμογές μπορούν να επικοινωνούν με τον διακομιστή (Server), καθοδηγώντας τον να εκτελέσει τις απαιτούμενες εργασίες τους.

  * **Πελάτης (Client):** Ο πελάτης (Client) δεν είναι τίποτα άλλο από μια διεπαφή γραμμής εντολών, η οποία επιτρέπει στους χρήστες να αλληλεπιδρούν με το Docker χρησιμοποιώντας εντολές. Όταν χρησιμοποιούμε εντολές, ο πελάτης στέλνει αυτές τις οδηγίες στον dockerd (Docker Daemon), ο οποίος στη συνέχεια τις εκτελεί. Η εντολή που χρησιμοποιείται από το docker εξαρτάται από το Docker API. Στο Docker, ο πελάτης μπορεί να αλληλεπιδράσει με περισσότερες από μία διεργασίες δαίμονα.

- **Εικόνες Docker (Docker Images):** Μια εικόνα Docker (Docker Image) είναι ένα πρότυπο που περιέχει την εφαρμογή και όλες τις απαιτούμενες εξαρτήσεις για να τρέξει αυτή η εφαρμογή στο Docker. Αυτή η εικόνα καθορίζει πώς ένα container πρέπει να δημιουργηθεί, καθορίζοντας ποια στοιχεία λογισμικού θα λειτουργήσουν και πώς. Οι εικόνες δημιουργούνται χρησιμοποιώντας Dockerfiles. Τα Dockerfiles χρησιμοποιούν DSL (Domain Specific Language) και περιέχουν οδηγίες για τη δημιουργία μιας εικόνας Docker. Τα Dockerfiles ορίζουν τις διαδικασίες για τη γρήγορη παραγωγή μιας εικόνας. Κατά τη δημιουργία της εφαρμογής σας, πρέπει να δημιουργήσετε ένα Dockerfile σωστά δομημένο, καθώς ο Docker Daemon εκτελεί όλες τις εντολές από πάνω προς τα κάτω.

- **Δοχεία Docker (Docker Containers):** Όπως αναφέρθηκε νωρίτερα, ένα Docker Container είναι μια λογική οντότητα.Ουσιαστικά είναι μια εκτελούμενη περίπτωση της εικόνας Docker.



- **Τομοι Docker (Docker Volumes):** Οι τόμοι είναι φάκελοι ή αρχεία που υπάρχουν στο σύστημα αρχείων του ξενιστή (host) και προσαρτώνται στα containers για την αποθήκευση δεδομένων που δημιουργούνται ή τροποποιούνται από αυτά. Αποθηκεύονται σε ένα μέρος του συστήματος αρχείων του ξενιστή που διαχειρίζεται ειδικά από το Docker και δεν πρέπει να τροποποιείται από άλλες διαδικασίες. Οι τόμοι αποτελούν τον προτιμότερο τρόπο αποθήκευσης δεδομένων container καθώς προσφέρουν κάλυτερη απόδοση και είναι απομονωμένοι από τις άλλες λειτουργίες του ξενιστή Docker.

- **Δίκτυο Docker (Docker Network):** Ένα δίκτυο είναι μια ομάδα δύο ή περισσότερων συσκευών που μπορούν να επικοινωνούν μεταξύ τους είτε φυσικά είτε εικονικά. Το δίκτυο Docker είναι ένα εικονικό δίκτυο που δημιουργείται από το Docker για να επιτρέπει την επικοινωνία μεταξύ των Docker containers. Αν δύο containers λειτουργούν στον ίδιο ξενιστή, μπορούν να επικοινωνούν μεταξύ τους χωρίς την ανάγκη για την αποκάλυψη θυρών (ports) στον φιλοξενούμενο υπολογιστή. Μπορείτε να χρησιμοποιήσετε το Docker για να διαχειριστείτε τους Docker ξενιστές σε οποιαδήποτε λειτουργικό, ανεξάρτητα από το αν λειτουργούν με Windows, Linux ή συνδυασμό των δύο. Το Docker υποστηρίζει αρκετούς τύπους δικτύων, ο καθένας εξυπηρετεί διαφορετικές περιπτώσεις χρήσης:

  * **Δίκτυο Γέφυρας (Bridge Network):** Ο προεπιλεγμένος τρόπος δικτύου όταν εκτελείτε ένα container. Δημιουργεί ένα ιδιωτικό εσωτερικό δίκτυο στον ξενιστή, και τα containers που συνδέονται σε αυτό το δίκτυο λαμβάνουν μια εσωτερική διεύθυνση IP. Ιδανικό για αυτόνομα containers που χρειάζονται να επικοινωνήσουν.

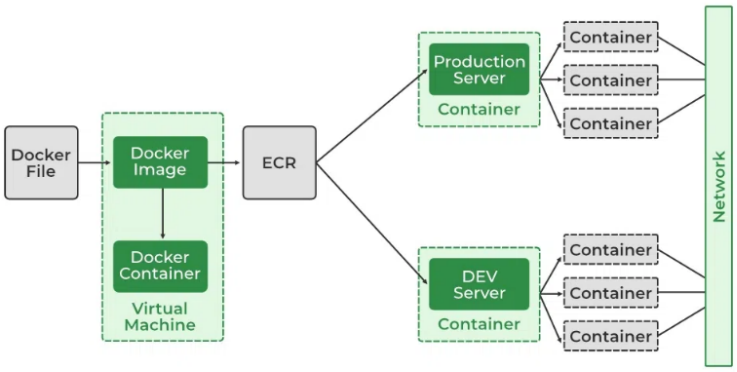
  * **Δίκτυο Ξενιστή (Host Network):** Αφαιρεί την απομόνωση δικτύου μεταξύ του container και του ξενιστή Docker. Το container μοιράζεται το δικτυακό περιβάλλον του ξενιστή και είναι άμεσα προσβάσιμο στο δίκτυο του ξενιστή.

  * **Κανένα Δίκτυο (None Network):** Απενεργοποιεί όλη τη δικτύωση για το container. Χρησιμοποιείται όταν απαιτείται πλήρη δικτυακή απομόνωση.

  * **Δίκτυο Overlay (Overlay Network):** Επιτρέπει τη δικτύωση μεταξύ πολλαπλών Docker Daemon και είναι ιδανικό για το Docker Swarm, υποστη-

ρίζοντας δικτύωση πολλαπλών ξενιστών. (Το Docker Swarm είναι ένα εργαλείο ενορχήστρωσης containers).

* **Προσαρμοσμένα Δίκτυα (Custom Networks):** Μπορείτε να δημιουργήσετε προσαρμοσμένα δίκτυα χρησιμοποιώντας τους δικτυακούς οδηγούς του Docker ή άλλων υπηρεσιών για να προσαρμόσετε τα δικτυακά περιβάλλοντα.



– **Μητρώα Docker (Docker Registries - Hub)**

Ένα μητρώο Docker είναι ένα σύστημα για την αποθήκευση και διανομή εικόνων Docker με συγκεκριμένα ονόματα. Μπορεί να υπάρχουν αρκετές εκδόσεις της ίδιας εικόνας, η καθεμία με το δικό της σετ ετικετών. Ένα μητρώο Docker διαχωρίζεται σε αποθετήρια Docker (Docker repositories), το καθένα από τα οποία περιέχει όλες τις τροποποιήσεις της εικόνας. Το μητρώο μπορεί να χρησιμοποιηθεί από τους χρήστες του Docker για να ανακτήσουν εικόνες τοπικά και να ανεβάσουν νέες εικόνες στο μητρώο (αν επιτρέπεται απο τα κατάλληλα δικαιώματα πρόσβασης όταν είναι εφικτό). Το μητρώο είναι μια εφαρμογή server-side που αποθηκεύει και διανέμει εικόνες Docker και είναι εξαιρετικά κλιμακούμενο.

- **Κύρια χαρακτηριστικά των Μητρώων Docker**

  * **Αποθήκευση των Εικόνων Docker (Storage of Docker Images):** Τα μητρώα αποθηκεύουν εικόνες Docker, οι οποίες μπορεί να περιλαμβάνουν προ-ρυθμισμένα λειτουργικά συστήματα, εφαρμογές και εξαρτήσεις.

  * **Εκδόσεις Εικόνων και Ετικετοποίηση (Image Versioning and Tagging):** Διατηρούν διαφορετικές εκδόσεις των εικόνων χρησιμοποιώντας ετικέτες, επιτρέποντας στους χρήστες να παρακολουθούν και να επανέρχονται σε

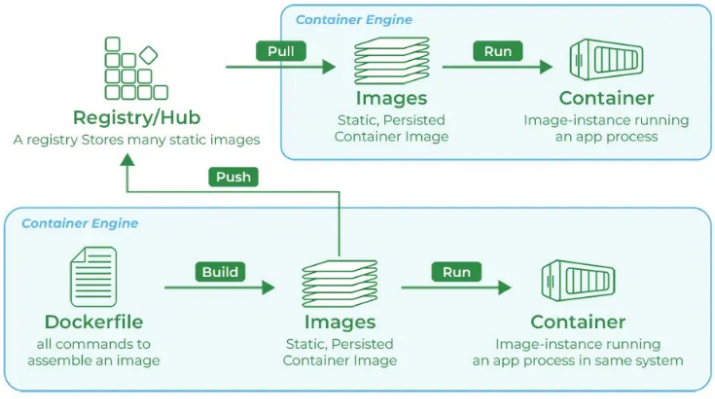
συγκεκριμένες εκδόσεις όποτε χρειάζεται.

* **Δημόσια και Ιδιωτικά Αποθετήρια (Public and Private Repositories):** Τα μητρώα Docker μπορούν να φιλοξενήσουν τόσο δημόσια αποθετήρια, προσβάσιμα από όλους, όσο και ιδιωτικά αποθετήρια για προσωπικές εικόνες Docker που πρέπει να διατηρούνται ιδιωτικές.

- **Τύποι Μητρώων Docker**
  * **Docker Hub:** Το προεπιλεγμένο δημόσιο μητρώο για εικόνες Docker. Φιλοξενεί έναν μεγάλο αριθμό επίσημων εικόνων από διάφορους παρόχους λογισμικού και εικόνες από ατομικούς προγραμματιστές.

  * **Ιδιωτικά Μητρώα (Private Registries):** Οι οργανισμοί μπορούν να δημιουργήσουν τα δικά τους ιδιωτικά μητρώα για να αποθηκεύουν και να διαχειρίζονται εικόνες. Αυτό συχνά χρησιμοποιείται για ιδιόκτητο λογισμικό ή εικόνες που περιέχουν ιδιωτικές πληροφορίες.

  * **Μητρώα τρίτων (Third-Party Registries):** Άλλοι πάροχοι cloud και υπηρεσίες τρίτων προσφέρουν υπηρεσίες μητρώου Docker, όπως το AWS Elastic Container Registry (ECR), το Google Container Registry (GCR), και το Azure Container Registry (ACR).



– **Εκδόσεις και Συνδρομές Docker**

Το Docker προσφέρει διάφορες εκδόσεις για να καλύψει μια ευρεία γκάμα χρηστών, από ατομικούς προγραμματιστές έως μεγάλες επιχειρήσεις. Ακολουθεί μια επισκόπηση των διαθέσιμων εκδόσεων Docker:

- **Docker Community Edition (CE):** Αυτή είναι η ανοικτού κώδικα, δωρεάν έκδοση του Docker, ιδανική για ατομικούς προγραμματιστές, μικρές ομάδες και οποιονδήποτε ενδιαφέρεται να πειραματιστεί με τη διαμόρφωση σε δοχεία (containerization). Προσφέρει μια ισχυρή πλατφόρμα για τη δημιουργία και διαχείριση δοχείων και παρέχει πρόσβαση σε υποστήριξη από την κοινότητα του Docker, καθώς και πιστοποιημένα δοχεία και επιπρόσθετα στοιχεία διαθέσιμα στο Docker Store.

- **Docker Enterprise Edition (EE):** Αυτή η λύση επιχειρησιακής κλίμακας είναι προσαρμοσμένη για οργανισμούς που απαιτούν προηγμένα χαρακτηριστικά, ασφάλεια, υψηλή κλιμάκωση και υποστήριξη. Διατίθεται σε τρεις προσφορές:

  * **Docker EE Basic:** Περιλαμβάνει το Docker Engine για πιστοποιημένη υποδομή και υποστήριξη από την Docker Inc., καθώς και πρόσβαση σε πιστοποιημένα δοχεία και επιπρόσθετα στοιχεία.

  * **Docker EE Standard:** Προσφέρει προηγμένες δυνατότητες διαχείρισης εικόνων και δοχείων, ενσωμάτωση LDAP και role-based access control (RBAC).

  * **Docker EE Enterprise:** Η υψηλότερη κατηγορία, που προσφέρει ενισχυμένα χαρακτηριστικά ασφάλειας (όπως η σάρωση ευπαθειών και η υπογραφή εικόνων - vulnerability scanning and image signing), υποστήριξη για Kubernetes και ειδική επιχειρησιακή υποστήριξη.

Οι συνδρομές Docker προσφέρουν μια προσαρμοσμένη γκάμα υπηρεσιών και χαρακτηριστικών, καλύπτοντας τις διάφορες ανάγκες από ατομικούς προγραμματιστές έως μεγάλες επιχειρήσεις, διασφαλίζοντας ότι κάθε χρήστης μπορεί να βρει τα κατάλληλα εργαλεία για τα projects του σχετικά με τη διαμόρφωση σε δοχεία (containerization).

- **Docker Personal:** Ιδανικό για κοινότητες ανοικτού κώδικα, ατομικούς προγραμματιστές, εκπαίδευση και μικρές επιχειρήσεις. Περιλαμβάνει απεριόριστα δημόσια αποθετήρια, διακριτικά πρόσβασης, συνεργάτες για δημόσια αποθετήρια και το Docker Scout Free για ασφάλεια της αλυσίδας εφοδιασμού λογισμικού.

- **Docker Pro:** Στοχεύει σε ατομικούς προγραμματιστές, προσφέροντας περισσότερο έλεγχο στο περιβάλλον ανάπτυξης. Περιλαμβάνει όλα τα χαρακτηριστικά του Docker Personal συν απεριόριστα ιδιωτικά αποθετήρια, 5000 ανακτήσεις εικόνων ανά ημέρα, εως 5 ταυτόχρονες αυτόματες κατασκευές και 300 σαρώσεις ευπαθειών.

- **Docker Team:** Προσφέρει χαρακτηριστικά συνεργασίας, παραγωγικότητας και ασφάλειας για ομάδες προγραμματιστών. Περιλαμβάνει όλα όσα το Docker

Pro, συν απεριόριστες ομάδες, εως 15 ταυτόχρονες αυτόματες κατασκευές, απεριόριστη σάρωση ευπαθειών και προηγμένα εργαλεία συνεργασίας και διαχείρισης.

- **Docker Business:** Σχεδιασμένο για επιχειρήσεις που χρησιμοποιούν το Docker σε μεγάλη κλίμακα, προσφέροντας κεντρική διαχείριση και προηγμένα χαρακτηριστικά ασφάλειας. Περιλαμβάνει όλα όσα στο Docker Team, μαζί με το hardened Docker Desktop (ενισχυμένη έκδοση του Docker Desktop), διαχείριση πρόσβασης σε εικόνες και αποθετήρια, και εργαλεία για τη διαχείριση πολλαπλών οργανισμών και ρυθμίσεων.

Κάθε έκδοση και συνδρομή του Docker είναι σχεδιασμένη για να ικανοποιεί συγκεκριμένες ανάγκες, διασφαλίζοντας μια σειρά επιλογών για χρήστες διαφορετικών επιπέδων και απαιτήσεων. Από την ατομική ανάπτυξη έως τις εκτεταμένες εταιρικές εγκαταστάσεις, το Docker παρέχει μια σειρά από εργαλεία και υπηρεσίες για την αποτελεσματική δημιουργία, αποστολή και εκτέλεση, διαμορφωμένων σε δοχεία, εφαρμογών.

– **Απαιτήσεις Συστήματος**

- **Windows**

  * **WSL 2 backend**

    · WSL version 1.1.3.0 or later.

    · Windows 11 64-bit: Home or Pro version 21H2 or higher, or Enterprise or Education version 21H2 or higher.

    · Windows 10 64-bit: We recommend Home or Pro 22H2 (build 19045) or higher, or Enterprise or Education 22H2 (build 19045) or higher. Minimum required is Home or Pro 21H2 (build 19044) or higher, or Enterprise or Education 21H2 (build 19044) or higher.

    · Turn on the WSL 2 feature on Windows. For detailed instructions, refer to the Microsoft documentation.

    · The following hardware prerequisites are required to successfully run WSL 2 on Windows 10 or Windows 11: **1)** 64-bit processor with Second Level Address Translation (SLAT - Second Level Address Translation). **2)** 4GB system RAM. **3)** Enable hardware virtualization in BIOS. For more information, see Virtualization.

  * **Hyper-V backend and Windows containers**

- Windows 11 64-bit: Pro version 21H2 or higher, or Enterprise or Education version 21H2 or higher.
- Windows 10 64-bit: We recommend Home or Pro 22H2 (build 19045) or higher, or Enterprise or Education 22H2 (build 19045) or higher. Minimum required is Home or Pro 21H2 (build 19044) or higher, or Enterprise or Education 21H2 (build 19044) or higher. For Windows 10 and Windows 11 Home, see the system requirements in the WSL 2 backend tab.
- Turn on Hyper-V and Containers Windows features.
- The following hardware prerequisites are required to successfully run Client Hyper-V on Windows 10: **1)** 64-bit processor with Second Level Address Translation (SLAT - Second Level Address Translation). **2)** 4GB system RAM. **3)** Turn on BIOS-level hardware virtualization support in the BIOS settings. For more information, see Virtualization.

- **Linux**

  Docker provides .deb and .rpm packages from the following Linux distributions and architectures: Ubuntu, Fedora, Debian.

  * 64-bit kernel and CPU support for virtualization.
  * KVM virtualization support. Follow the KVM virtualization support instructions to check if the KVM kernel modules are enabled and how to provide access to the KVM device.
  * QEMU must be version 5.2 or later. We recommend upgrading to the latest version.
  * systemd init system.
  * Gnome, KDE, or MATE Desktop environment. For many Linux distros, the Gnome environment does not support tray icons. To add support for tray icons, you need to install a Gnome extension. For example, AppIndicator.
  * At least 4 GB of RAM.
  * Enable configuring ID mapping in user namespaces, see File sharing.
  * Recommended: Initialize pass for credentials management.

- **Mac**

* **Mac with Intel chip**
    - A supported version of macOS.
    - At least 4 GB of RAM.
* **Mac with Apple silicon**
    - A supported version of macOS.
    - At least 4 GB of RAM.

      Beginning with Docker Desktop 4.3.0, we have removed the hard requirement to install Rosetta 2. There are a few optional command line tools that still require Rosetta 2 when using Darwin/AMD64. See Known issues. However, to get the best experience, we recommend that you install Rosetta 2. To install Rosetta 2 manually from the command line, run the following command:

Για περισσότερες πληροφορίες, παρακαλώ ανατρέξτε στις Windows, Linux και Mac σελίδες απαιτήσεων συστήματος.

– **Εγκατάσταση και εκπαιδευτικό υλικό**

Για τη διαδικασία εγκατάστασης σε διάφορα λειτουργικά συστήματα, αναζητήθηκε ολοκληρωμένη καθοδήγηση από το επίσημο εγχειρίδιο του Docker για Windows, Linux, and Mac. Το εκπαιδευτικό υλικό σχετικά με τη χρήση του Docker είναι προσβάσιμο μέσω των παρακάτω συνδέσμων: GeeksforGeeks, freeCodeCamp, και Docker Curriculum. Επιπλέον, συνιστούμε την παρακολούθηση ενός εκαπιδευτικού βίντεο που παρέχεται εδώ.

2 Τι είναι το Kubernetes

– **Επισκόπηση του Kubernetes**

Όπως αναφέρει ο ορισμός, το Kubernetes ή k8s είναι ένα ανοιχτού κώδικα σύστημα ενορχήστρωσης και διαχείρισης συστοιχιών για εφαρμογές βασισμένες σε δοχεία που συντηρείται από το Cloud Native Computing Foundation. Με απλά λόγια, το Kubernetes βοηθά στη διαχείριση εφαρμογών που βρίσκονται σε δοχεία σε διάφορους τύπους φυσικών, εικονικών και cloud περιβαλλόντων. Επίσης, καθιστά την τοποθέτηση δοχείων πολύ εύκολη χρησιμοποιώντας ένα δηλωτικό αρχείο YAML. Καθορίζετε πώς θέλετε να γίνει η τοποθέτηση του δοχείου, και το Kubernetes φροντίζει για αυτό διαβάζοντας τις πληροφορίες που παρέχονται στο YAML. Σύμφωνα

με την state of Kubernetes report απο την Splunk, το 96% των οργανισμών χρησιμοποιούν ή αξιολογούν το Kubernetes και 5,6 εκατομμύρια προγραμματιστές χρησιμοποιούν το Kubernetes σήμερα. Επιπλέον, υπήρξε αύξηση πάνω από 300% στη χρήση δοχείων σε παραγωγικά περιβάλλοντα τα τελευταία 5 χρόνια.

– **Χαρακτηριστικά του Kubernetes**

Το Kubernetes σας βοηθά να ελέγχετε την κατανομή πόρων και τη διαχείριση σειράς εκτέλεσης για εφαρμογές cloud και μικροϋπηρεσίες. Βοηθά επίσης στην απλοποίηση διάφορων πτυχών των υποδομών που βασίζονται στις υπηρεσίες. Το Kubernetes σας επιτρέπει να διασφαλίζετε πού και πότε τρέχουν οι εφαρμογές που βρίσκονται σε δοχεία και σας βοηθά να βρείτε τους πόρους και τα εργαλεία με τα οποία θέλετε να εργαστείτε. Ακολουθούν τα κύρια χαρακτηριστικά του Kubernetes:

- **Αυτοματοποιημένος Προγραμματισμός (Automated Scheduling):** Αναθέτει αυτόματα πόρους και προγραμματίζει δοχεία με βάση τις απαιτήσεις τους και τους διαθέσιμους πόρους, βελτιστοποιώντας την αποδοτικότητα.

- **Ικανότητες Αυτοθεραπείας (Self-Healing Capabilities):** Αντικαθιστά ή επανεκκινεί αυτόματα δοχεία που αποτυγχάνουν, δεν ανταποκρίνονται ή δεν πληρούν τους ορισμένους ελέγχους ποιότητας.

- **Αυτοματοποιημένες Αναπτύξεις & Επαναφορές (Automated Rollouts & Rollbacks):** Διαχειρίζεται την τοποθέτηση νέων εκδόσεων εφαρμογών και αυτόματα επαναφέρει σε μια προηγούμενη σταθερή έκδοση σε περίπτωση αποτυχίας.

- **Οριζόντια Κλιμάκωση & Ισορροπία Φόρτου(Horizontal Scaling & Load Balancing):** Διευκολύνει την κλιμάκωση των εφαρμογών προς τα πάνω ή προς τα κάτω ανάλογα με τη ζήτηση και ελέγχει την σειρά εκέλεσης στο δίκτυο για να εξασφαλίσει τη σταθερότητα.

- **Προσφέρει ένα συνεπές περιβάλλον για Ανάπτυξη, Δοκιμές και Παραγωγή:** Εξασφαλίζει ότι οι εφαρμογές λειτουργούν συνεπώς σε διαφορετικά περιβάλλοντα ανάπτυξης, δοκιμών και παραγωγής.

- **Η Υποδομή Είναι Χαλαρά Συνδεδεμένη:** Κάθε συστατικό λειτουργεί ανεξάρτητα, το οποίο αυξάνει τη συνολική ανθεκτικότητα και ευελιξία του συστήματος.

- **Παρέχει υψηλότερη πυκνότητα αξιοποίησης πόρων:** Αποδοτικά μεγιστοποιεί τη χρήση των πόρων, μειώνοντας τη σπατάλη και το κόστος.

- **Προσφέρει χαρακτηριστικά έτοιμα για επιχειρησιακή χρήση:** Περιλαμβάνει χαρακτηριστικά όπως ασφάλεια, έλεγχο και επεκτασιμότητα που είναι απα-

ραίτητα για εγκατάσταση σε επιχειρησιακό επίπεδο.

- **Διαχείριση επικεντρωμένη στην εφαρμογή:** Επικεντρώνεται στη διαχείριση της εφαρμογής αντί των μεμονωμένων μηχανημάτων, καθιστώντας ευκολότερη την τοποθέτηση περίπλοκων εφαρμογών.

- **Αυτόματα κλιμακούμενη υποδομή:** Προσαρμόζει αυτόματα τους υπολογιστικούς πόρους όπως απαιτείται, με βάση τις απαιτήσεις του φορτίου εργασίας.

– **Βασικά χαρακτηριστικά του Kubernetes (Kubernetes Basics)**

- **Συστοιχία (Cluster):** Είναι μια συλλογή από διακομιστές (servers) που σας βοηθά να συγκεντρώσετε τους διαθέσιμους πόρους τους. Αυτό περιλαμβάνει τη μνήμη RAM, την CPU και τις συσκευές τους σε μια χρηστική δεξαμενή.

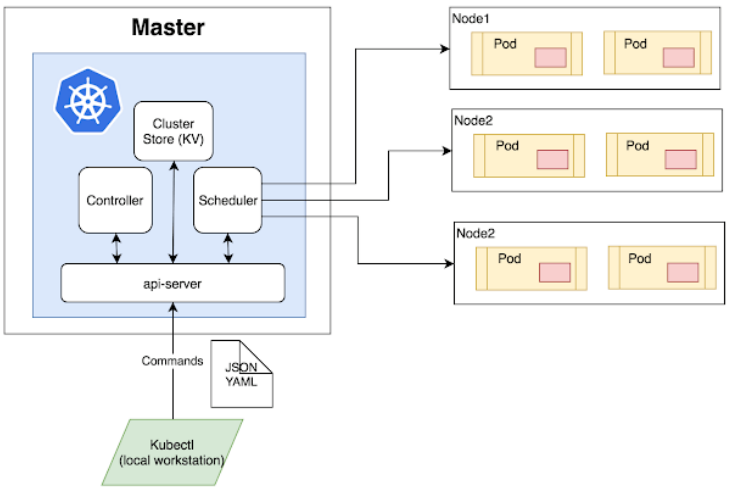
- **Master:** Το master είναι μια συλλογή από εξαρτήματα που αποτελούν το κέντρο ελέγχου του Kubernetes. Αυτά χρησιμοποιούνται για να πάρουν όλες τις αποφάσεις σχετικά με τη συστοιχία. Περιλαμβάνει τόσο τον προγραμματισμό όσο και την ανταπόκριση στα συμβάντα της συστοιχίας.

- **Κόμβος (Node):** Είναι ένας μοναδικός διακομιστής ο οποίος είναι ικανός να λειτουργεί σε φυσικό ή εικονικό μηχάνημα. Ένας κόμβος θα πρέπει να εκτελεί τόσο το kube-proxy (ένας proxy δικτύου που εκτελείται σε κάθε κόμβο της συστοιχίας.), το minikube (ελαφρύτερη έκδοση του Kubernetes), όσο και το kubelet (ο μηχανισμός του Kubernetes για τη δημιουργία δοχείων σε έναν εργατικό κόμβο), τα οποία θεωρούνται μέρος της συστοιχίας.

- **Namespace:** Είναι ένας λογικός Cluster ή περιβάλλον. Είναι μια ευρέως διαδεδομένη μέθοδος που χρησιμοποιείται για την περιορισμό της πρόσβασης ή τον διαχωρισμό μιας συστοιχίας.

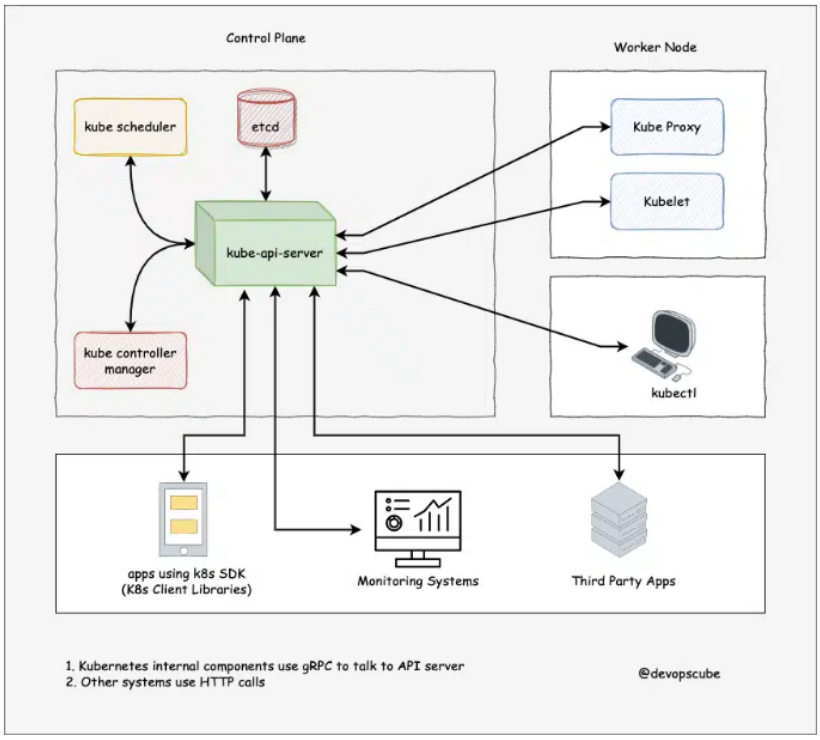
– **Αρχιτεκτονική του Kubernetes (Kubernetes Architecture)**

- **Ελεγκτικό Επίπεδο - Κόμβος Master (Control Plane - Master Node):** Ο κόμβος master είναι το πρώτο και πιο σημαντικό συστατικό που είναι υπεύθυνο για τη διαχείριση της συστοιχίας Kubernetes. Αποτελεί το σημείο εισόδου για όλους τους τύπους διοικητικών εργασιών. Μπορεί να υπάρχουν περισσότεροι από ένας κόμβος master στη συστοιχία για να εξασφαλιστεί η ανοχή σε σφάλματα. Ο κόμβος master έχει διάφορα συστατικά όπως τον Διακομιστή API(API Server), τον Διαχειριστή Ελεγκτών(Controller Manager), τον Προγραμματιστή(Scheduler) και το ETCD. Ας ρίξουμε μια ματία σε αυτά:

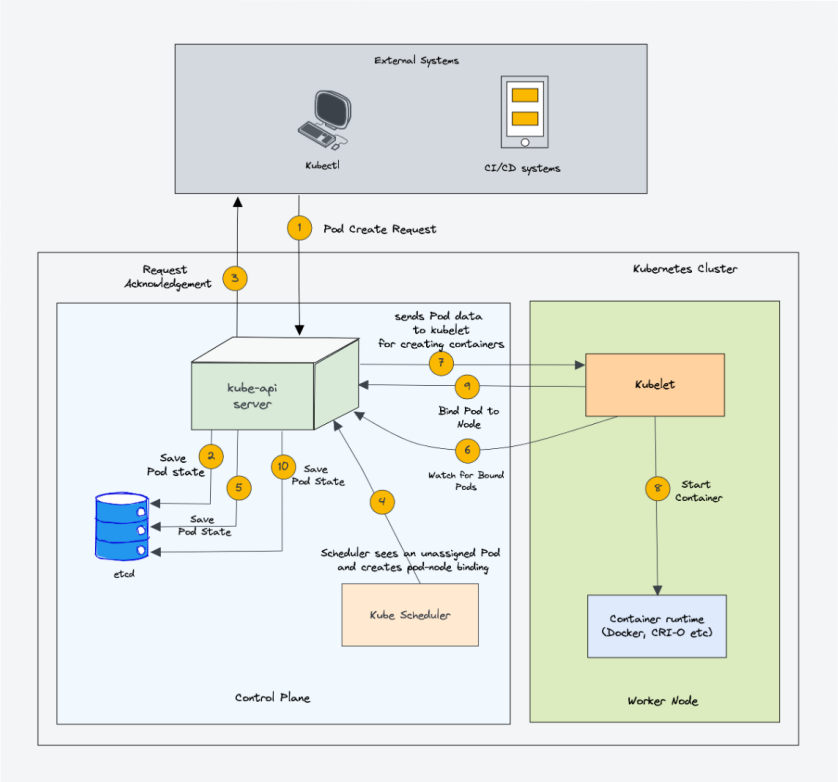


Kubernetes Architecture Diagram

* **Διακομιστής API(API Server):** Ο διακομιστής API λειτουργεί ως σημείο εισόδου για όλες τις εντολές REST που χρησιμοποιούνται για τον έλεγχο της συστοιχίας. Έτσι, όταν χρησιμοποιείτε το kubectl (Kubernetes command-line tool) για τη διαχείριση της συστοιχίας, στο τομέα του backend επικοινωνείτε πραγματικά με τον διακομιστή API μέσω των HTTP REST APIs. Ωστόσο, τα εσωτερικά συστατικά της συστοιχίας όπως ο προγραμματιστής, ο διαχειριστής ελέγχων κλπ. επικοινωνούν με τον διακομιστή API χρησιμοποιώντας gRPC.



* **Προγραμματιστής (Scheduler):** Ο προγραμματιστής (scheduler) προγραμματίζει τα tasks στον εργατικό κόμβο (slave node). Αποθηκεύει πληροφορίες σχετικά με τη χρήση πόρων για κάθε εργατικό κόμβο. Είναι υπεύθυνος για τη διανομή του φορτίου εργασίας. Βοηθά επίσης να παρακολουθείτε πώς αντιμετωπίζεται ο φόρτος εργασίας στους κόμβους της συστοιχίας. Σας βοηθά να τοποθετήσετε το φόρτο εργασίας σε πόρους που είναι διαθέσιμοι και τον δέχονται. Ακολουθεί πώς λειτουργεί ο προγραμματιστής.



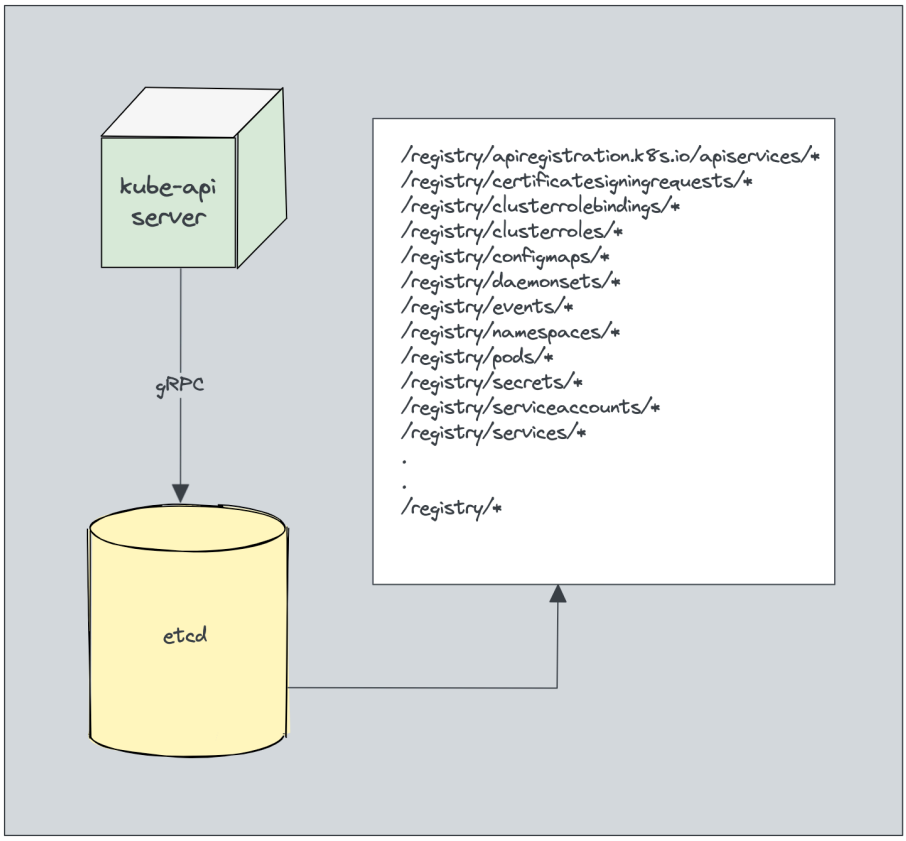
* **etcd:** Το Kubernetes είναι ένα κατανεμημένο σύστημα και χρειάζεται μια αποδοτική βάση δεδομένων όπως το etcd που υποστηρίζει τη φύση του. Λειτουργεί τόσο ως backend υπηρεσία όσο και ως βάση δεδομένων. Μπορείτε να το αποκαλέσετε τον εγκέφαλο της συστοιχίας Kubernetes. Το Etcd είναι ένα ανοιχτού κώδικα, απόλυτα συνεπές, κατανεμημένο σύστημα αποθήκευσης κλειδιού-τιμής:

    - **Ισχυρά Συνεπές (Strongly consistent):** Αν γίνει μια ενημέρωση σε έναν κόμβο, η ισχυρή συνέπεια θα διασφαλίσει ότι θα ενημερωθεί σε όλους τους άλλους κόμβους της συστοιχίας άμεσα. Επίσης, αν κοιτάξετε το θεώρημα CAP, η επίτευξη 100% διαθεσιμότητας με ισχυρή συνέπεια και ανοχή σε διαμερισμότητα είναι αδύνατη.

    - **Κατανεμημένο (Distributed):** Το etcd είναι σχεδιασμένο να λειτουργεί σε πολλαπλούς κόμβους ως συστοιχία χωρίς να θυσιάζει τη συνέπεια.

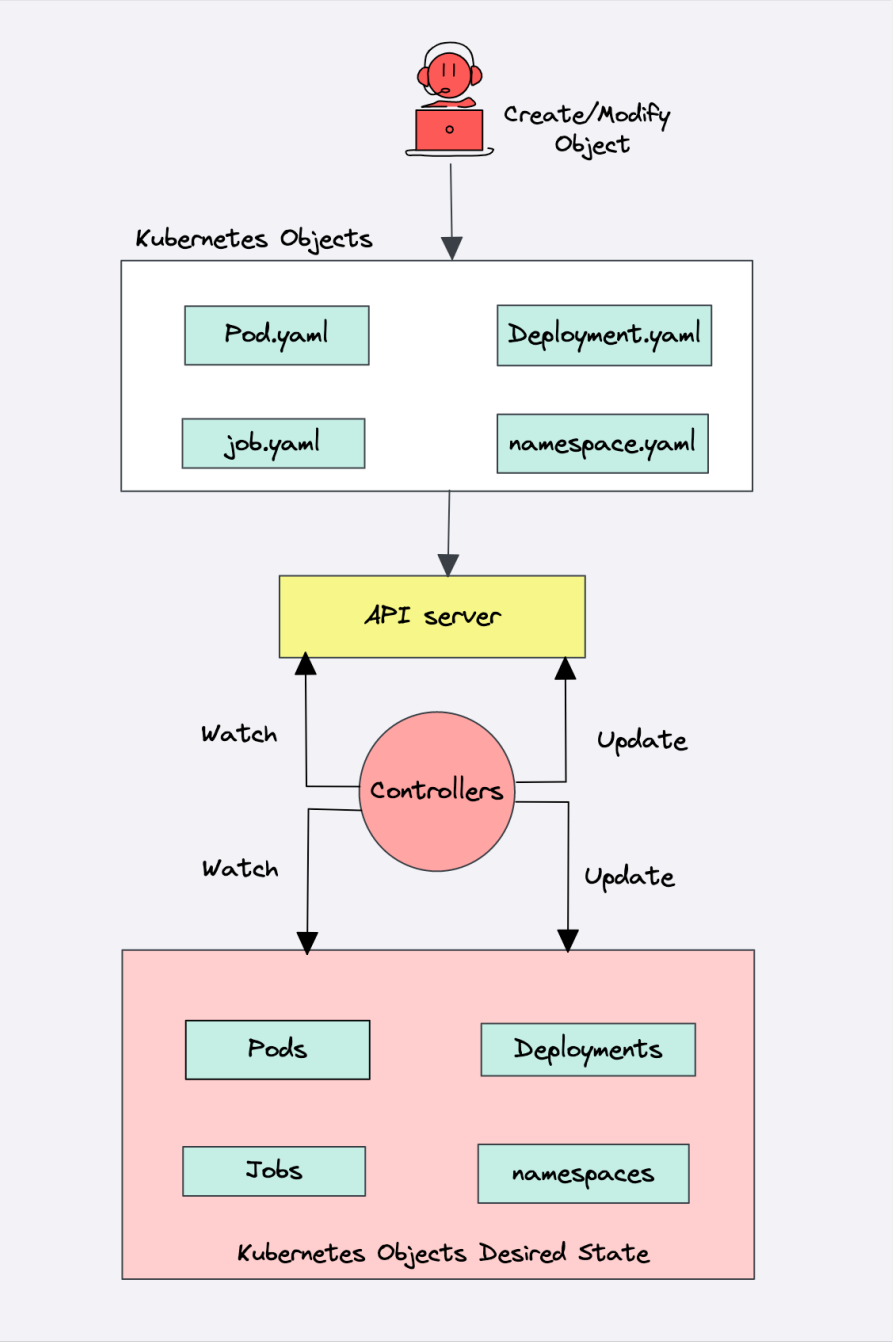
- **Key Value Store:** Μια μη σχεσιακή βάση δεδομένων που αποθηκεύει δεδομένα ως κλειδιά και τιμές. Επίσης, παρέχει μια διεπαφή API κλειδιού-τιμής. Η αποθήκη δεδομένων είναι δομημένη πάνω σε BboltDB η οποία είναι ένα παρακλάδι του BoltDB.

Το Etcd χρησιμοποιεί τον αλγόριθμο συναίνεσης raft για ισχυρή συνέπεια και διαθεσιμότητα. Λειτουργεί με τρόπο leader-member για υψηλή διαθεσιμότητα και για να αντέξει σε πολλαπλές αποτυχίες κόμβων.

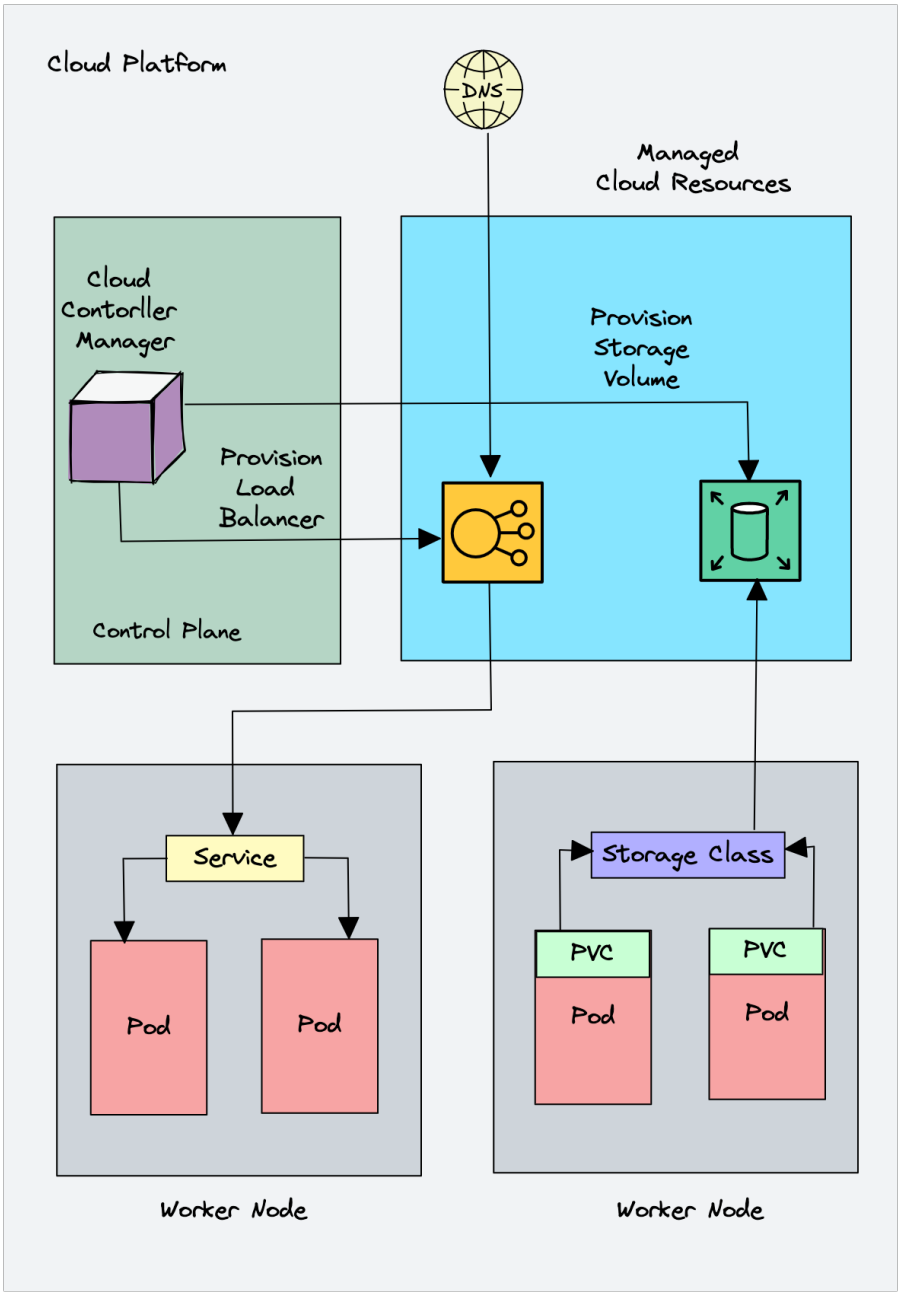


* **Διαχειριστής Ελεγκτών (Controller Manager):** Είναι ένα στοιχείο που διαχειρίζεται όλους τους ελεγκτές του Kubernetes. Πόροι/αντικείμενα του Kubernetes όπως τα pods (σηλλογή απο containers), τα namespaces, τα jobs (διάφορες λειτουργίες) και τα ReplicaSets (χρησιμοποιείται για τη διατήρηση ενός σταθερού σετ αντιγραφών pods που λειτουργούν εντός μιας συστοιχίας σε κάθε στιγμή) διαχειρίζονται από τους αντίστοιχους ελεγκτές.

Επίσης, ο προγραμματιστής (scheduler) είναι ένας ελεγκτής που διαχειρίζεται ο controller manager. (Οι ελεγκτές είναι προγράμματα που εκτελούν ατέρμονους κύκλους ελέγχου. Δηλαδή λειτουργούν συνεχώς και παρακολουθούν την πραγματική και την επιθυμητή κατάσταση των αντικειμένων. Αν υπάρχει διαφορά μεταξύ της πραγματικής και της επιθυμητής κατάστασης, διασφαλίζουν ότι ο πόρος/αντικείμενο του Kubernetes μετατρέπεται στην επιθυμητή κατάσταση.).


Create/Modify
Object
Kubernetes Objects
Pod.yaml
Deployment.yaml
job.yaml
namespace.yaml
API server
Watch
Update
Controllers
Watch
Update
Pods
Deployments
Jobs
namespaces
Kubernetes Objects Desired State

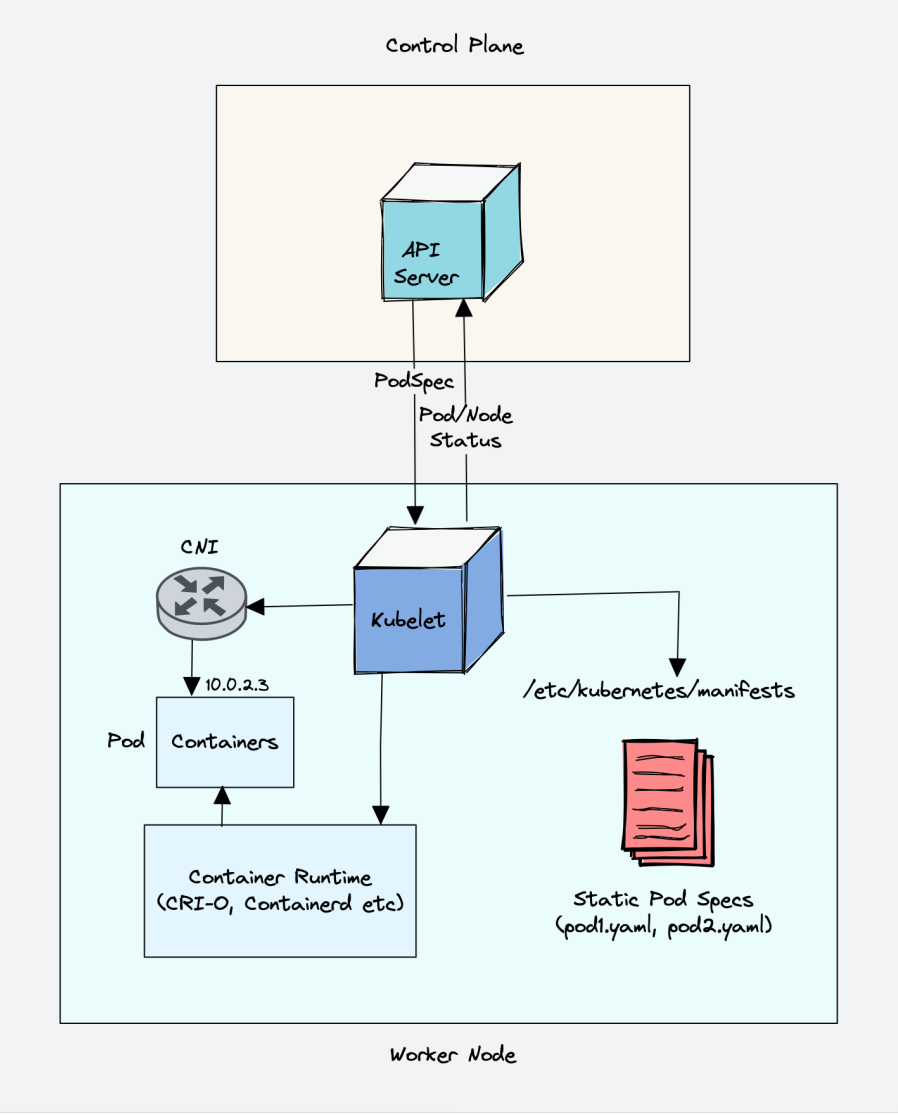
* **Διαχειριστής Ελέγχου Cloud (Cloud Controller Manager):** Όταν το Kubernetes εγκαθίσταται σε περιβάλλοντα cloud, ο διαχειριστής ελέγχου cloud (cloud controller manager) λειτουργεί ως γέφυρα μεταξύ των APIs της πλατφόρμας Cloud και της συστοιχίας Kubernetes. Με αυτόν τον τρόπο, τα βασικά συστατικά του Kubernetes μπορούν να λειτουργούν ανεξάρτητα και να επιτρέπουν στους παρόχους cloud να ενσωματώνονται με το Kubernetes μέσω επιπρόσθετων στοιχείων (plugins)(Για παράδειγμα, μια διεπαφή μεταξύ της συστοιχίας Kubernetes και του cloud API της AWS). Η ενσωμάτωση του διαχειριστή ελέγχου cloud επιτρέπει στη συστοιχία Kubernetes να παρέχει πόρους όπως παρουσίες(instances) για κόμβους, Ισορροπητές Φόρτου (Load Balancers) για υπηρεσίες και Τόμους Αποθήκευσης (Storage Volumes).



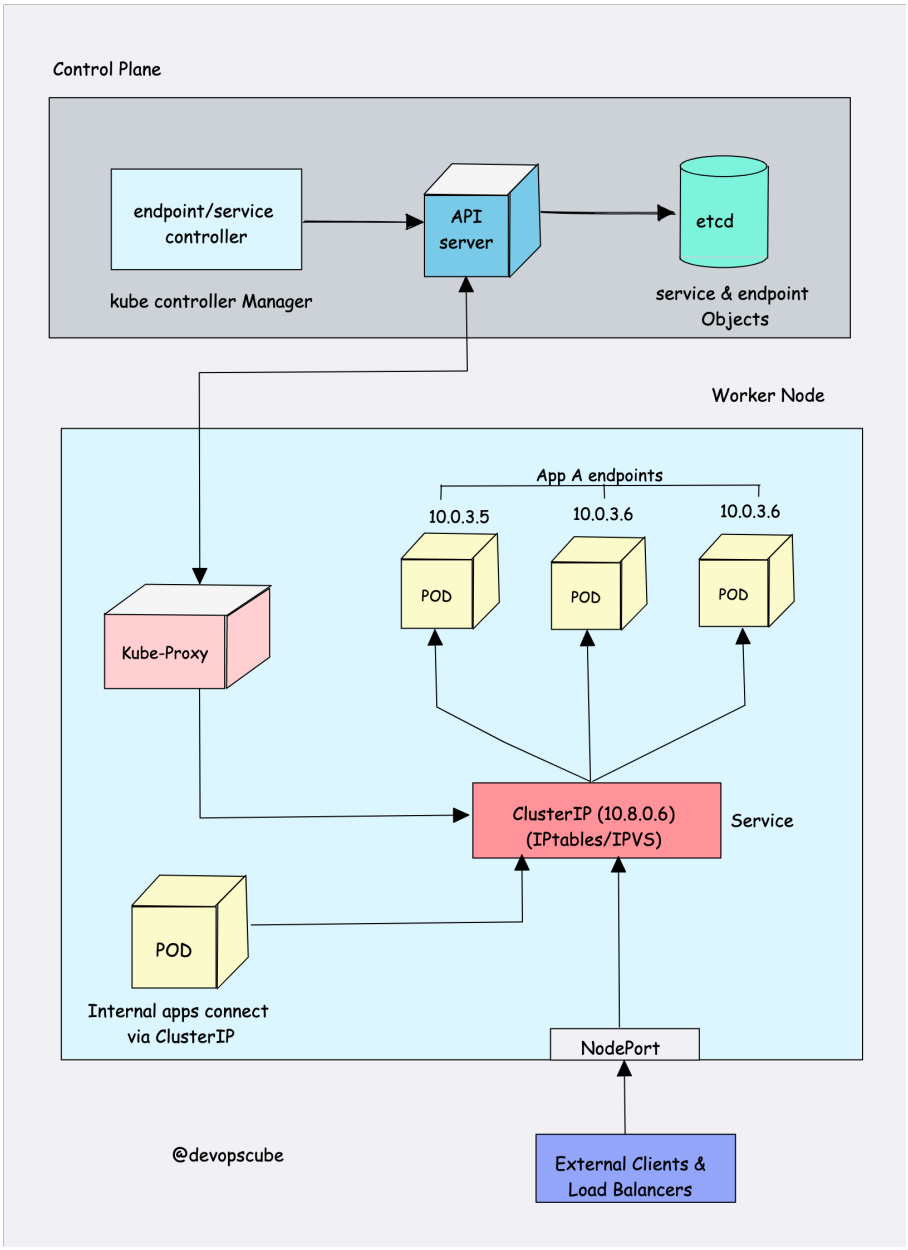
- **Κόμβος - Εργάτης (Worker Node):** Οι κόμβοι - εργάτες είναι οι εργάτες μιας συστοιχίας Kubernetes. Τρέχουν τις εφαρμογές και τα φορτία εργασίας. Κάθε

κόμβος περιέχει:

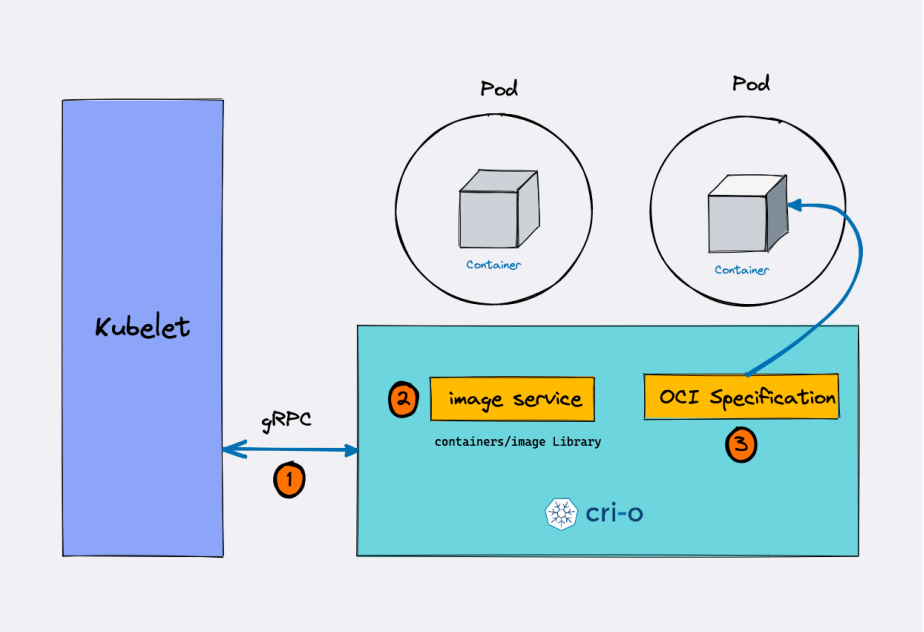
* **Kubelet:** Λαμβάνει τη διαμόρφωση ενός Pod (συλλογή απο containers) από τον διακομιστή API και διασφαλίζει ότι τα περιγραφόμενα δοχεία λειτουργούν και βρίσκονται σε ενεργή κατάσταση.



* **Kube-Proxy:** Ένας proxy δικτύου που λειτουργεί σε κάθε κόμβο, διατηρώντας τους κανόνες δικτύου και επιτρέποντας την επικοινωνία με τα Pods σας από δικτυακές συνεδρίες μέσα ή έξω από τη συστοιχία σας.



    * **Περιβάλλον εκτέλεσης δοχείων (Container Runtime):** Το λογισμικό που είναι υπεύθυνο για την εκτέλεση δοχείων (όπως το Docker).



- **Pods:** Οι μικρότερες εκτελέσιμες μονάδες που δημιουργούνται και διαχειρίζονται από το Kubernetes. Ένα pod είναι μια ομάδα ενός ή περισσοτέρων δοχείων, με κοινόχρηστους πόρους αποθήκευσης/δικτύου και μια προδιαγραφή για το πώς να εκτελεστούν τα δοχεία.

- **Υπηρεσίες και Ingress (Services and Ingress):** Οι υπηρεσίες ορίζουν ένα λογικό σύνολο από Pods και μια πολιτική με την οποία θα τα προσεγγίσουν. Αυτό επιτυγχάνεται χρησιμοποιώντας διευθύνσεις IP και ονόματα DNS. Από την άλλη πλευρά, το Ingress παρέχει εξωτερική πρόσβαση στις υπηρεσίες εντός μιας συστοιχίας, συνήθως μέσω HTTP.

- **ConfigMaps and Secrets**: Σας επιτρέπουν να αποθηκεύετε δεδομένα διαμόρφωσης και ευαίσθητες πληροφορίες ξεχωριστά από τον κώδικα της εφαρμογής σας, κάτι που είναι ιδιαίτερα χρήσιμο στην αρχιτεκτονική μικροϋπηρεσιών.

- **Αποθηκευτικός χώρος**

    * **Τόμοι (Volumes):** Παρέχουν έναν τρόπο αποθήκευσης δεδομένων που παράγονται και χρησιμοποιούνται από τα Pods, ανεξάρτητα από τον κύκλο

ζωής τους.

* **PersistentVolume:** Ένα PersistentVolume (PV) αντιπροσωπεύει έναν πόρο αποθήκευσης. Τα PV συνδέονται συνήθως με έναν πόρο αποθήκευσης υποστήριξης, NFS, GCEPersistentDisk, RBD κ.λπ. και παρέχονται εκ των προτέρων. Ο κύκλος ζωής τους χειρίζεται ανεξάρτητα από ένα pod.

* **PersistentVolumeClaim:** Ένα PersistentVolumeClaim (PVC) είναι ένα αίτημα για αποθήκευση που ικανοποιεί ένα σύνολο απαιτήσεων αντί να αντιστοιχίζεται απευθείας σε έναν πόρο αποθήκευσης. Χρησιμοποιείται συνήθως με δυναμικά εφοδιασμένο χώρο αποθήκευσης.

* **PersistentVolumeClaim:** Οι κλάσεις αποθήκευσης είναι μια αφαίρεση πάνω από έναν εξωτερικό πόρο αποθήκευσης. Αυτές περιλαμβάνουν έναν προμηθευτή, παραμέτρους διαμόρφωσης προμηθευτή καθώς και μια πολιτική ανάκτησης PV.

– **Υψηλή διαθεσιμότητα του Kubernetes (Kubernetes High Availability)**

Η Υψηλή Διαθεσιμότητα (High Availability) στο Kubernetes είναι ένα κρίσιμο χαρακτηριστικό για τη διασφάλιση ότι οι εφαρμογές και οι υπηρεσίες που λειτουργούν στη συστοιχία είναι διαθέσιμες και προσβάσιμες σε όλες τις περιστάσεις, ελαχιστοποιώντας τον χρόνο προβλημάτων και τις διακοπές των υπηρεσιών. Ας εξετάσουμε πώς το Kubernetes επιτυγχάνει υψηλή διαθεσιμότητα:

- **Αυξημένο επιπέδου ελέγχου (Control Plane Redundancy):**

  * Σε μια ρύθμιση υψηλής διαθεσιμότητας, τα συστατικά του ελεγκτικού επιπέδου του Kubernetes (Διακομιστής API, etcd, Διαχειριστής Ελέγχου, Χρονοδιάγραμμα) αντιγράφονται σε πολλαπλούς κόμβους. Αυτό διασφαλίζει ότι αν ένας ή περισσότεροι κόμβοι αποτύχουν, το ελεγκτικό επίπεδο παραμένει λειτουργικό.

  * Η αποθήκη etcd, η οποία είναι ένα κρίσιμο συστατικό του ελεγκτικού επιπέδου, συχνά λειτουργεί σε κατάσταση συστοιχίας διασκορπισμένη σε πολλαπλούς κόμβους για να αποτρέπει την απώλεια δεδομένων και να ενισχύει τη διαθεσιμότητα.

- **Ισορροπία Φόρτου Διακομιστή API(API Server Load Balancing):**

  * Οι αιτήσεις προς τον διακομιστή API του Kubernetes είναι ισορροπημένες σε φόρτο μεταξύ πολλαπλών εκδόσεων του διακομιστή. Αυτό αποτρέπει οποιοδήποτε ενιαίο σημείο αποτυχίας και διασφαλίζει ότι ο διακομιστής API είναι πάντα προσβάσιμος.

  - Σε περιβάλλοντα cloud, αυτό συχνά διαχειρίζεται από cloud load balancers, ενώ σε περιβάλλοντα εντός των εγκαταστάσεων, μπορεί να περιλαμβάνει ειδικό υλικό ή λύσεις ισορροπίας φορτίου βασισμένες σε λογισμικό.

- **Αυξημένος αριθμός κόμβων (Node Redundancy):**

  - Η υψηλή διαθεσιμότητα επεκτείνεται επίσης στους εργατικούς κόμβους. Λειτουργώντας πολλαπλούς κόμβους, το Kubernetes μπορεί να αντέξει την αποτυχία ενός ή περισσότερων κόμβων, διατηρώντας παράλληλα τις εφαρμογές σε λειτουργία.

  - Ο αλγόριθμος προγραμματισμού του Kubernetes λαμβάνει υπόψη τη διαθεσιμότητα των κόμβων, διανέμοντας τα Pods σε διαφορετικούς κόμβους για να μειώσει τον αντίκτυπο μιας μεμονωμένης αποτυχίας κόμβου.

- **Αυτοματοποιημένη ανακατεύθυνση σε περίπτωση Αποτυχίας (Automated Failover):**

  - Το Kubernetes παρακολουθεί συνεχώς την λειτουργικότητα των κόμβων και των pods. Αν ένας κόμβος ή ένα pod αποτύχει, το Kubernetes αναπρογραμματίζει αυτόματα τα pods σε λειτουργικούς κόμβους.

  - Οι υπηρεσίες στο Kubernetes χρησιμοποιούν επιλογείς για να ανακατευθύνουν αυτόματα την εργασία σε διαθέσιμα pods, διασφαλίζοντας ότι οι υπηρεσίες της εφαρμογής είναι συνεχώς προσβάσιμες ακόμη και αν κάποια αποτύχουν.

- **ReplicaSets and Ανάπτυξη:**Τα ReplicaSets διασφαλίζουν ότι ένας συγκεκριμένος αριθμός αντιγράφων των pods λειτουργούν σε κάθε δοσμένη στιγμή. Αν ένα pod αποτύχει, το ReplicaSet δημιουργεί αυτόματα ένα νέο.

- **StatefulSets:** Για εφαρμογές σε συνεχή λειτουργία (όπως βάσεις δεδομένων), τα StatefulSets διασφαλίζουν ότι η κατάσταση της εφαρμογής διατηρείται ακόμη και όταν τα pods επαναπρογραμματίζονται. Αυτό είναι κρίσιμο για εφαρμογές που απαιτούν μόνιμη λειτουργία και σταθερό δίκτυο.

- **Μόνιμη Αποθήκευση (Persistent Storage):** Το Kubernetes υποστηρίζει μόνιμη αποθήκευση, επιτρέποντας την αποθήκευση δεδομένων ανεξάρτητα από τα pods. Αυτό σημαίνει ότι ακόμα και αν ένα pod επαναπρογραμματιστεί σε έναν νέο κόμβο, μπορεί ακόμη να έχει πρόσβαση στα δεδομένα του, πράγμα που είναι ουσιώδες για την υψηλή διαθεσιμότητα σε εφαρμογές που απαιτούν μόνιμη λειτουργία.

- **Ομοσπονδία Συστοιχιών (Cluster Federation):** Για περιβάλλοντα που απαιτούν ακόμη υψηλότερα επίπεδα διαθεσιμότητας, το Kubernetes υποστηρίζει την ομοσπονδία συστοιχιών, επιτρέποντας τη σύνδεση πολλαπλών συστοιχιών

Kubernetes μεταξύ τους. Αυτό παρέχει ανοχή σε σφάλματα σε διαφορετικές περιοχές ή κέντρα δεδομένων.

– **Απαιτήσεις Συστήματος**

- **Windows**
  - **CPU**: 2-core processor.
  - **Memory**:4 GB RAM minimum (8 GB recommended).
  - **Storage**: 20 GB of free space.
  - **Operating System**: Windows 10 Pro, Enterprise, or Education; Build 15063 or later.
  - **Container Runtime**: Docker Desktop for Windows (includes Kubernetes support), or Windows Subsystem for Linux 2 (WSL2) for a more Linux-like experience.
  - **Network**: General broadband internet or LAN connection.
  - **Virtualization**: Hardware virtualization support with Hyper-V and Containers Windows features enabled.
  - Unique hostname, MAC address, and product_uuid for every node. See here for more details.
- **Linux**
  - **CPU**: 2-core processor.
  - **Memory**: 2 GB RAM minimum (4 GB recommended).
  - **Storage**: 20 GB of free space.
  - **Operating System**: Most modern distributions like Ubuntu 20.04, CentOS 7, Fedora, Debian, etc.
  - **Container Runtime**: Docker, containerd, or any runtime compatible with Kubernetes CRI (Container Runtime Interface).
  - **Network**: General broadband internet or LAN connection.
  - **Virtualization Support (for Minikube or similar)**: Enabled in BIOS.

    - Unique hostname, MAC address, and product_uuid for every node. See here for more details.

- **Mac**

    - **CPU**: Intel or Apple Silicon processor.
    - **Memory**: 4 GB RAM minimum (8 GB recommended).
    - **Storage**: 30 GB of free space.
    - **Operating System**: macOS 10.15 (Catalina) or later.
    - **Container Runtime**: Docker Desktop for Mac (includes Kubernetes support).
    - **Network**: General broadband internet or LAN connection.
    - **Virtualization**: Docker Desktop provides a virtual environment.
    - Unique hostname, MAC address, and product_uuid for every node. See here for more details.

– **Εγκατάσταση και εκπαιδευτικό υλικό**

Τηρήσαμε τις επίσημες οδηγίες του Kubernetes για τη διαδικασία εγκατάστασης σε διάφορα λειτουργικά συστήματα. Συγκεκριμένα, ακολουθήσαμε τις οδηγίες για την εγκατάσταση στο Windows, Linux, and Mac. Επιπρόσθετα, για μια πιο πλήρη κατανόηση, ανατρέξαμε σε εξωτερικό εκπαιδευτικό υλικό όπως το Guru99 Kubernetes Tutorial και το DevOpsCube Kubernetes Guide for Beginners. Τέλος, συνιστούμε την παρακολούθηση ενός εκπαιδευτικού βίντεο που παρέχεται εδω

3 Docker and Kubernetes: αλληλοϋποστηριζόμενα

Αρχικά, είναι σημαντικό να κατανοήσουμε ότι το Docker και το Kubernetes είναι διαφορετικές τεχνολογίες, η καθεμία με τον δικό της σκοπό. Δεν βρίσκονται όμως σε ανταγωνισμό το ένα με το άλλο, αντιθέτως είναι συμπληρωματικά εργαλεία που συχνά χρησιμοποιούνται από κοινού. Το ερώτημα δεν θα πρέπει να είναι για την επιλογή του ενός έναντι του άλλου ή τη σύγκριση των δυνατοτήτων τους. Αντίθετα, πρόκειται για την αναγνώριση των ατομικών τους ρόλων εντός του τοπίου του DevOps και το πώς μπορούν να λειτουργήσουν συνεργατικά. Το Docker ειδικεύεται στον εγκλεισμό της εφαρμογής σας σε δοχεία. Ο βασικός του ρόλος είναι στη συσκευασία και διανομή της εφαρμογής σας. Αντιθέτως, το Kubernetes λειτουργεί ως σύστημα ενορχήστρωσης δοχείων. Η κύρια χρήση του είναι στην ανάπτυξη και κλιμάκωση της εφαρμογής σας. Όταν το Docker και το Kubernetes συνδυάζονται, σχηματίζουν ένα αποδοτικό δίδυμο για τις διαδικασίες DevOps. Όπως αναφέρθηκε, η απλή εκτέλεση δοχείων δεν είναι

αρκετή για περιβάλλοντα παραγωγής, απαιτείται και αποτελεσματική διαχείριση. Εδώ είναι που το Kubernetes ξεχωρίζει, προσφέροντας χαρακτηριστικά που είναι ζωτικής σημασίας για την αποτελεσματική διαχείριση του κύκλου ζωής των δοχείων.

4 Συμπέρασμα

Εξερευνήσαμε τους συμπληρωματικούς ρόλους του Docker και του Kubernetes στον DevOps. Το Docker παρέχει αποτελεσματική διαμόρφωση σε δοχεία, ενώ το Kubernetes διαχειρίζεται αυτές τις εφαρμογές σε μεγάλη κλίμακα. Μαζί, δημιουργούν ένα ισχυρό σύστημα στον DevOps, βελτιώνοντας τη λειτουργική αποδοτικότητα και εξασφαλίζοντας υψηλή διαθεσιμότητα και επεκτασιμότητα. Επιπλέον, αντιμετωπίζουν μελλοντικές προκλήσεις, όπως η ασφάλεια και η ενσωμάτωση νέων τεχνολογιών, ενισχύοντας το DevOps. Τελικά, προσφέρουν μια ολοκληρωμένη λύση για σύγχρονη ανάπτυξη λογισμικού στην εποχή του cloud.

5 Πηγές

- Docker Documentation
- Docker Documentation
- Docker Documentation
- Docker Documentation
- Docker Documentation
- Docker Documentation
- Kubernetes Documentation
- Kubernetes Documentation
- Kubernetes Documentation
- Easy installation