ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Τεχνικές εικονικοποίησης σε επίπεδο λειτουργικού συστήματος: Μελέτη και σενάρια χρήσης-αξιοποίησης μέσω Docker

Διπλωματική Εργασία

του

Παναγιώτη-Αίαντα Νικολάου Δρακόπουλου

ΑΜ: 711141020

Επιβλέπων: Βασίλειος Μάμαλης, Καθηγητής

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Τεχνικές εικονικοποίησης σε επίπεδο λειτουργικού συστήματος: Μελέτη και σενάρια χρήσης-αξιοποίησης μέσω Docker

Διπλωματική Εργασία

του

Παναγιώτη - Αίαντα Νικολάου Δρακόπουλου

Επιβλέπων: Βασίλειος Μάμαλης, Καθηγητής

Εγκρίθηκε από την κάτωθι τριμελή επιτροπή την 15 Οκτωβρίου 2021.

ΜΑΜΑΛΗΣ ΒΑΣΙΛΕΙΟΣ	ΚΑΝΤΖΑΒΕΛΟΥ ΙΩΑΝΝΑ	ΚΑΡΚΑΖΗΣ ΠΑΝΑΓΙΩΤΗΣ
Vasileios Mamalis Digitally signed by Vasileios Mamalis Date: 2021.10.29 23:14:41 +03'00'	Ioanna Kantzavelou Digitally signed by Ioanna Kantzavelou Date: 2021.11.02 21:11:52 +02'00'	Panagiotis Karkazis Digitally signed by Panagiotis Karkazis Date: 2021.11.01 09:17:45 +02'00'

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Δρακόπουλος Παναγιώτης Αίας του Νικολάου, με αριθμό μητρώου 141020 του Τμήματος Μηχανικών της Σχολής Πληροφορικής και Υπολογιστών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω ότι:

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

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

Επιθυμώ την απαγόρευση πρόσβασης στο πλήρες κείμενο της εργασίας μου μέχρι 31/12/2021 και έπειτα από αίτηση μου στη Βιβλιοθήκη και έγκριση του επιβλέποντα καθηγητή.

Ο Δηλών

Περίληψη

Στη παρακάτω πτυχιακή εργασία του τελειόφοιτου φοιτητή Παναγιώτη – Αίαντα Δρακόπουλου γίνεται μια συνολική απεικόνιση των εννοιών της εικονικοποίησης (virtualization), πού συναντάται, και ποιες είναι οι κατηγορίες εικονικοποίησης . Στη συνέχεια γίνεται εκτενή αναφορά στις έννοιες του Containerization και του εργαλείου Docker. Τίθενται σε σύγκριση οι μέθοδοι της εικονικοποίησης και του Containerization και παρατίθενται τα πλεονεκτήματα της χρήσης της εικονικοποίησης με βάση το Docker. Μετέπειτα παρουσιάζεται ολόκληρη η αρχιτεκτονική του Docker, πώς επιτυγχάνεται η επικοινωνία των containers μεταξύ τους αλλά και με τον έξω κόσμο και πώς μπορεί ο χρήστης να προχωρήσει στη δημιουργία δικού του αποθετηρίου (repository) στο Docker Hub. Ακολουθεί αναφορά στη λειτουργία σμήνους στο Docker (docker swarm), πώς δημιουργείται ένα docker swarm, πώς επιτυγχάνεται η ενορχήστρωση των container, πώς λειτουργούν οι κόμβοι ανάλογα το ρόλο που τους έχει ανατεθεί και πώς προστίθενται στο σμήνος το οποίο έχει δημιουργηθεί. Κλείνοντας παρουσιάζεται η δημιουργία μιας υπηρεσίας nginx πάνω σε ένα cluster Docker Swarm και επίσης δημιουργήσουμε το δικό μας εξατομικευμένο Word Press Blog το οποίο
δεν θα «τρέχει» στον υπολογιστή μας αλλά σε docker container μέσω του Docker.
Ένας φυσικός υπολογιστής είναι μια συσκευή βασισμένη σε υλικό, όπως ένας προσωπικός υπολογιστής. Ο όρος γενικά χρησιμοποιείται για τη διαφοροποίηση υπολογιστών με βάση το υλικό από εικονικές μηχανές (Virtual Machine) που βασίζονται σε λογισμικό. Ένας φυσικός υπολογιστής διαθέτει επεξεργαστή (CPU), σκληρό δίσκο (Hard Drive), μνήμες RAM και σύνδεση στο δίκτυο. Στα πλαίσια του virtualization ο φυσικός υπολογιστής ονομάζεται Host. Virtualization είναι η διαδικασία χρήσης ενός «ειδικού λογισμικού» σε μια φυσική μηχανή για να δημιουργηθεί μια εικονική μηχανή (VM). Αυτό το «ειδικό λογισμικό» καλείται ως Hypervisor και η εικονική μηχανή που δημιουργείται καλείται ως Guest. Πριν τους Hypervisors, οι περισσότεροι φυσικοί υπολογιστές μπορούσαν να τρέξουν μόνο ένα λειτουργικό σύστημα, δηλαδή το υλικό (hardware) χειριζόταν αιτήματα μόνο από το συγκεκριμένο λειτουργικό σύστημα. Το μειονέκτημα αυτής της προσέγγισης ήταν ότι σπαταλούσε πόρους καθώς το λειτουργικό σύστημα δεν μπορούσε πάντα να χρησιμοποιήσει όλη την υπολογιστική του ισχύ. Οι Hypervisors λύνουν αυτό το πρόβλημα. Πρόκειται για ένα μικρό στρώμα λογισμικού που επιτρέπει σε πολλαπλά λειτουργικά συστήματα να τρέχουν παράλληλα, μοιράζοντας τους ίδιους φυσικούς πόρους υπολογιστών. Αυτά τα λειτουργικά συστήματα είναι τα εικονικά μηχανήματα (VMs), τα οποία μιμούνται ένα ολόκληρο περιβάλλον υπολογιστικού υλικού στο λογισμικό. Αν και κάποια μορφή εικονικοποίησης υπήρχε από τα μέσα

της δεκαετίας του 1960, εξελίχθηκε με την πάροδο του χρόνου, παραμένοντας όμως κοντά στις ρίζες της. Μεγάλο μέρος της εξέλιξης στην εικονικοποίηση έχει συμβεί τα τελευταία χρόνια, με νέους τύπους να αναπτύσσονται και να εμπορευματοποιούνται. Οι διαφορετικοί τύποι εικονικοποίησης περιορίζονται σε εικονικοποίηση επιφάνειας εργασίας (Desktop Virtualization), εικονικοποίηση εφαρμογών (Application Virtualization), εικονικοποίηση διακομιστή (Server Virtualization), εικονικοποίηση αποθήκευσης (Storage Virtualization) και εικονικοποίηση δικτύου (Network Virtualization).
Το Containerization επιτρέπει στους προγραμματιστές να δημιουργούν και να αναπτύσσουν εφαρμογές γρηγορότερα και με μεγαλύτερη ασφάλεια. Με τις παραδοσιακές μεθόδους, ο κώδικας αναπτύσσεται σε ένα συγκεκριμένο υπολογιστικό περιβάλλον το οποίο, όταν μεταφέρεται σε μια νέα τοποθεσία, συχνά οδηγεί σε σφάλματα (bugs, errors). Για παράδειγμα, όταν ένας προγραμματιστής μεταφέρει κώδικα από επιτραπέζιο υπολογιστή σε εικονική μηχανή (VM) ή από Linux σε λειτουργικό σύστημα Windows. Το Containerization εξαλείφει αυτό το πρόβλημα ομαδοποιώντας τον κωδικό εφαρμογής μαζί με τα σχετικά αρχεία διαμόρφωσης, βιβλιοθήκες και εξαρτήσεις που απαιτούνται για την εκτέλεση του. Αυτό το ενιαίο πακέτο λογισμικού ή κοντέινερ αφαιρείται από το λειτουργικό σύστημα του κεντρικού υπολογιστή, και ως εκ τούτου, είναι μόνο του και γίνεται φορητό - ικανό να τρέχει σε οποιαδήποτε πλατφόρμα ή cloud, χωρίς προβλήματα. Η έννοια του Containerization είναι δεκαετιών, αλλά η εμφάνιση του ανοιχτού κώδικα Docker Engine το 2013, ένα βιομηχανικό πρότυπο για κοντέινερ με απλά εργαλεία προγραμματιστή και μια καθολική προσέγγιση packaging, επιτάχυνε την υιοθέτηση αυτής της τεχνολογίας. Το κοντέινερ (container) είναι μια τυπική μονάδα λογισμικού που συσκευάζει κώδικα και όλες τις εξαρτήσεις του, έτσι ώστε η εφαρμογή να εκτελείται γρήγορα και αξιόπιστα από το ένα υπολογιστικό περιβάλλον στο άλλο. Μια εικόνα κοντέινερ (Container Image) είναι ένα ελαφρύ, αυτόνομο, εκτελέσιμο πακέτο λογισμικού που περιλαμβάνει όλα όσα χρειάζονται για την εκτέλεση μιας εφαρμογής: κωδικός, χρόνος εκτέλεσης, εργαλεία συστήματος, βιβλιοθήκες συστήματος και ρυθμίσεις. Η τεχνολογία του Containerization προσφέρει σημαντικά οφέλη για προγραμματιστές και ομάδες ανάπτυξης. Μεταξύ αυτών είναι η φορητότητα, η ευελιξία, η ταχύτητα, η απομόνωση βλαβών, η ευκολία διαχείρισης, η αποδοτικότητα και η ασφάλεια. Αφού εξετάσαμε καθεμία από τις τεχνολογίες ξεχωριστά, το ερώτημα είναι, ποια πρέπει να προτιμάται; Η απάντηση εξαρτάται από πολλά διαφορετικά σημεία. Ουσιαστικά, κάθε επιχείρηση ή εφαρμογή έχει διαφορετικές ανάγκες, απαιτήσεις και σκοπό. Η επιλογή του virtualization έναντι του containerization εξαρτάται από την επιχειρηματική ανάπτυξη, το επιχειρησιακό μοντέλο ή τον τρόπο σύνταξης και παραγωγής των εφαρμογών. Και οι δύο είναι τεχνολογίες λογισμικού που δημιουργούν αυτόνομα εικονικά πακέτα, αλλά για να επιλέξουμε αυτό που θα ταιριάζει καλύτερα στις ανάγκες του χρήστη, θα εξετάσουμε τα ακόλουθα σημεία: ταχύτητα, διαχείριση πόρων,

ασφάλεια, φορητότητα, το κύκλο ζωής της εφαρμογής και τις απαιτήσεις λειτουργικού συστήματος. Ωστόσο, είναι σημαντικό να σημειωθεί ότι υπάρχουν τρόποι συνδυασμού containerization και virtualization έτσι ώστε τα πλεονεκτήματα και των δύο τεχνολογιών να συνδυάζονται.
Το Docker είναι μια ανοιχτή πλατφόρμα για ανάπτυξη, αποστολή και εκτέλεση εφαρμογών. Μας επιτρέπει να διαχωρίζουμε τις εφαρμογές (Application) από τη δομή (infrastructure), ώστε να μπορούμε να παραδίδουμε το λογισμικό γρήγορα. Με το Docker, μπορούμε να διαμερίζουμε την δομή μας με τον ίδιο τρόπο που διαχειριζόμαστε τις εφαρμογές μας. Αξιοποιώντας τις μεθοδολογίες του Docker για αποστολή, δοκιμή και ανάπτυξη κώδικα, μπορούμε να μειώσουμε σημαντικά την καθυστέρηση μεταξύ σύνταξης κώδικα και εκτέλεσής του στην παραγωγή. Συνεχίζοντας θα εξετάσουμε την αρχιτεκτονική του Docker και τα σχετικά στοιχεία της. Θα εξετάσουμε επίσης πώς λειτουργεί κάθε στοιχείο για να κάνει το Docker να λειτουργεί. Η αρχιτεκτονική του Docker χρησιμοποιεί ένα μοντέλο διακομιστή-πελάτη και περιλαμβάνει τα στοιχεία Docker Client, Docker Host, Network and Storage και το Docker Registry / Hub. Τα κοντέινερ είναι ενθυλακωμένα περιβάλλοντα στα οποία εκτελούνται οι εφαρμογές. Το κοντέινερ ορίζεται από την εικόνα και τυχόν πρόσθετες επιλογές διαμόρφωσης που παρέχονται κατά την εκκίνηση του κοντέινερ, και δεν περιορίζονται στις συνδέσεις δικτύου και στις επιλογές αποθήκευσης. Τα κοντέινερ έχουν πρόσβαση μόνο σε πόρους που ορίζονται στην εικόνα, εκτός εάν ορίζεται πρόσθετη πρόσβαση κατά την κατασκευή της εικόνας σε κοντέινερ. Μπορούμε επίσης να δημιουργήσουμε μια νέα εικόνα με βάση την τρέχουσα κατάσταση ενός κοντέινερ. Δεδομένου ότι τα κοντέινερ είναι πολύ μικρότερα από τα VM, μπορούν να περιστραφούν σε λίγα δευτερόλεπτα και να έχουν πολύ καλύτερη πυκνότητα διακομιστή (server density). Φτάνοντας προς το τέλος της εργασίας, συναντάμε την έννοια του σμήνους (swarm) και εμβαθύνουμε στο πώς λειτουργούν οι κόμβοι είτε πρόκειται για manager είτε για worker nodes. Η λειτουργία σμήνους μας επιτρέπει να διαχειριζόμαστε ένα σύμπλεγμα Docker Engines, εγγενώς στην πλατφόρμα Docker. Μπορούμε να χρησιμοποιούμε το Docker CLI για να δημιουργήσουμε ένα σμήνος, να αναπτύξουμε υπηρεσίες εφαρμογών σε ένα σμήνος και να διαχειριστούμε τη συμπεριφορά του. Κλείνοντας, περιγράφετε βήμα-βήμα η δημιουργία μιας υπηρεσίας nginx σε ένα Docker Swarm Cluster και επίσης δημιουργήσουμε το δικό μας εξατομικευμένο Word Press Blog το οποίο δεν θα «τρέχει» στον υπολογιστή μας αλλά σε docker container μέσω του Docker.

Λέξεις-Κλειδιά: εικονικοποίηση, containerization, εικονική μηχανή (virtual machine), Docker, swarm, manager, worker, cluster

Abstract

In the dissertation of the senior student Panagiotis - Ajax Drakopoulos, an overview is made of the concepts of virtualization, where it occurs, and what are the categories of virtualization. The concepts of container and Docker tool are then discussed extensively. The methods of virtualization and containerization are compared and the advantages of using Docker-based virtualization are listed. Then the whole Docker architecture is presented, how the communication of the containers is achieved between them but also with the outside world and how the user can proceed to the creation of his own repository in the Docker Hub. The following is a reference to the swarm function in the Docker (docker swarm), how a docker swarm is created, how the orchestration of the containers is achieved, how the nodes work depending on the role assigned to them and how they are added to the swarm that has been created. In closing, the creation of a nginx service on a Docker Swarm cluster is presented.

A physical computer is a hardware-based device, like a personal computer. The term is generally used to differentiate hardware-based computers from software-based Virtual Machines. A physical computer has a processor (CPU), hard drive, RAM and network connection. In the context of virtualization, the physical computer is called the Host. Virtualization is the process of using "special software" on a physical machine to create a virtual machine (VM). This "special software" is called a Hypervisor and the virtual machine that is created is called a Guest. Before Hypervisors, most physical computers could only run one operating system, meaning that hardware only handled requests from that operating system. The disadvantage of this approach was that it was a waste of resources as the operating system could not always use all of its computing power. Hypervisors solve this problem. It is a small layer of software that allows multiple operating systems to run in parallel, sharing the same physical resources of computers. These operating systems are virtual machines (VMs) that mimic an entire computer hardware environment in software. Although some form of virtualization has existed since the mid-1960s, it has evolved over time, but remains close to its roots. Much of the evolution in virtualization has taken place in recent years, with new types being developed and commercialized. The different types of virtualization are limited to Desktop Virtualization, Application Virtualization, Server Virtualization, Storage Virtualization, and Network Virtualization.

Containerization allows developers to create and develop applications faster and more securely. With traditional methods, code is developed

in a specific computing environment which, when transferred to a new location, often leads to errors (bugs, errors). For example, when a programmer transfers code from a desktop computer to a virtual machine (VM) or from Linux to a Windows operating system. Containerization eliminates this problem by grouping the application code along with the relevant configuration files, libraries, and dependencies required to execute it. This single software package or container is removed from the host operating system, and therefore, it is self-contained and becomes portable - capable of running on any platform or cloud, without any problems. The concept of containerization is decades old, but the advent of open source Docker Engine in 2013, an industry standard for containers with simple programmer tools and a universal packaging approach, accelerated the adoption of this technology. A container is a standard piece of software that packs code and all its dependencies so that the application runs quickly and reliably from one computing environment to another. A Container Image is a lightweight, standalone, executable software package that includes everything you need to run an application: password, runtime, system tools, system libraries, and settings. Containerization technology offers significant benefits for developers and development teams. These include portability, flexibility, speed, fault isolation, ease of management, efficiency and safety. After looking at each of the technologies separately, the question is, which one should be preferred? The answer depends on many different points. Essentially, every business or application has different needs, requirements and purpose. The choice of virtualization over containerization depends on the business development, the business model or the way the applications are written and produced. Both are software technologies that create standalone virtual packages, but to choose the one that best suits the user's needs, we will look at the following points: speed, resource management, security, portability, application lifecycle and operating system requirements systemic. However, it is important to note that there are ways to combine containerization and virtualization so that the advantages of both technologies are combined.

Docker is an open platform for developing, sending and executing applications. It allows us to separate the application from the structure, so that we can deliver the software quickly. With Docker, we can partition our structure the same way we manage our applications. By utilizing Docker methodologies for sending, testing, and developing code, we can significantly reduce the delay between writing code and executing it in production. Continuing we will look at the Docker architecture and its related elements. We will also look at how each

component works to make Docker work. The Docker architecture wanted a client server model and included Docker Client, Docker Host, Network and Storage, and the Docker Registry / Hub. Containers are encapsulated environments in which applications run. The container is defined by the image and any additional configuration options provided at the startup of the container, and is not limited to network connections and storage options. Containers only have access to resources defined in the image, unless additional access is specified when constructing the image in a container. We can also create a new image based on the current state of a container. Since containers are much smaller than VMs, they can rotate in a matter of seconds and have a much better server density. Towards the end of the work, we meet the concept of swarm and go deeper into how the nodes work whether it is a manager or worker nodes. The swarm mode allows us to manage a set of Docker Engines, native to the Docker platform. We can use Docker CLI to create a swarm, deploy application services to a swarm, and manage its behavior. In closing, you describe step by step the creation of a nginx service in a Docker Swarm Cluster.

Keywords: virtualization, containerization, virtual machine, Docker, swarm, manager, worker, cluster.

Ευχαριστίες

Η εκπόνηση της Διπλωματικής μου εργασίας σηματοδοτεί το τέλος των προπτυχιακών σπουδών μου στη Σχολή Μηχανικών Πληροφορικής και Υπολογιστών του Πανεπιστημίου Δυτικής Αττικής. Πραγματοποιήθηκε στα πλαίσια του Εργαστηρίου Υπολογιστικών Νέφους, με επιβλέποντα καθηγητή τον κ. Βασίλειο Μάμαλη, τον οποίο και θα ήθελα να ευχαριστήσω πρωτίστως για την ευκαιρία που μου έδωσε να ασχοληθώ με το συγκεκριμένο επιστημονικό πεδίο, αλλά και για τα άτομα με τα οποία ο ίδιος με έφερε σε επικοινωνία και υπήρξαν αρωγοί στην προσπάθειά μου, τον κ. Ξυδά Ιωάννη και τον κ. Απόστολο Αναγνωστόπουλο.

Θα ήθελα επίσης να ευχαριστήσω τον καθηγητή κ. Καρκαζή Παναγιώτη και την καθηγήτρια κ. Καντζάβελου Ιωάννα που συμπλήρωσαν την τριμελή επιτροπή.

Θα αποτελούσε παράλειψη να μην ευχαριστήσω όλους τους φίλους μου που με την στήριξή τους με βοηθούν όλα αυτά τα χρόνια να πετύχω τους στόχους μου.

Τέλος, ευχαριστώ βαθύτατα τους γονείς μου Νικόλαο και Κατερίνα για την αγάπη, την υπομονή και την στήριξη που μου έχουν προσφέρει όλα αυτά τα χρόνια.

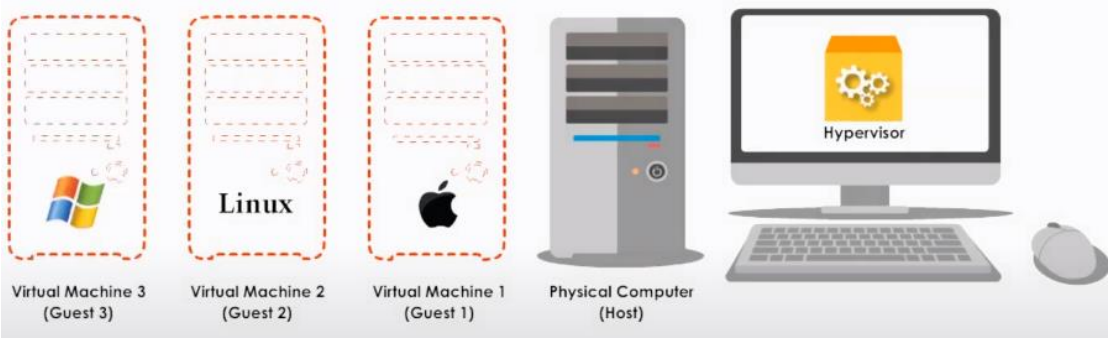
Πίνακας Περιεχομένων

1. Εισαγωγή

1.1 Εικονικοποίηση

Ένας φυσικός υπολογιστής (μερικές φορές ονομάζεται φυσικό μηχάνημα ή φυσικό κουτί) είναι μια συσκευή βασισμένη σε υλικό, όπως ένας προσωπικός υπολογιστής. Ο όρος γενικά χρησιμοποιείται για τη διαφοροποίηση υπολογιστών με βάση το υλικό από εικονικές μηχανές (Virtual Machines) που βασίζονται σε λογισμικό. Ένας φυσικός υπολογιστής διαθέτει επεξεργαστή (CPU), σκληρό δίσκο (Hard Drive), μνήμες Ram και σύνδεση στο δίκτυο. Στα πλαίσια του virtualization ο φυσικός υπολογιστής ονομάζεται Host.

Virtualization είναι η διαδικασία χρήσης ενός «ειδικού λογισμικού» σε μια φυσική μηχανή για να δημιουργηθεί μια εικονική μηχανή (VM). Αυτό το «ειδικό λογισμικό» καλείται ως Hypervisor και η εικονική μηχανή που δημιουργείται καλείται ως Guest.



Υπάρχουν μερικά σημαντικά σημεία σχετικά με τις εικονικές μηχανές. Καταρχάς έχουμε την δυνατότητα να δημιουργήσουμε όσες εικονικές μηχανές επιθυμούμε εφόσον όμως η επεξεργαστική ισχύς (CPU), η RAM και γενικά οι πόροι του Host μας το επιτρέπουν. Η RAM αποτελεί, στις περισσότερες περιπτώσεις, τον περιοριστικό παράγοντα. Μαζί, όλες οι εικονικές μηχανές που δημιουργήσαμε, χρησιμοποιούν τους πόρους του Host, αλλά η κάθε μια λειτουργεί ξεχωριστά. Επίσης, μια εικονική μηχανή μπορεί να ρυθμιστεί έτσι ώστε να χρησιμοποιεί όχι μόνο διαφορετικό λειτουργικό σύστημα (OS) αλλά και διαφορετικό τύπο επεξεργαστή (CPU) ή και μονάδα αποθήκευσης ή ακόμα και διαφορετικό τύπο NIC (Network Interface Controller) από αυτόν του Host. Ένας ελεγκτής διεπαφής δικτύου (NIC, επίσης γνωστός ως κάρτα διασύνδεσης δικτύου, προσαρμογέας δικτύου ή προσαρμογέας LAN) είναι ένα στοιχείο υλικού υπολογιστή που συνδέει έναν υπολογιστή με ένα δίκτυο υπολογιστών. Η απεικόνιση μιας φυσικής NIC σε μια οπτική NIC (vNIC) μπορεί να επιτρέψει σε μια εικονική μηχανή (VM) να συνδεθεί στο Internet. Μια εικονική μηχανή μπορεί να αποτελείται από αρχεία στον σκληρό δίσκο αλλά για το χρήστη δεν διαφέρει από έναν φυσικό υπολογιστή.

1.2 Η έννοια του Hypervisor

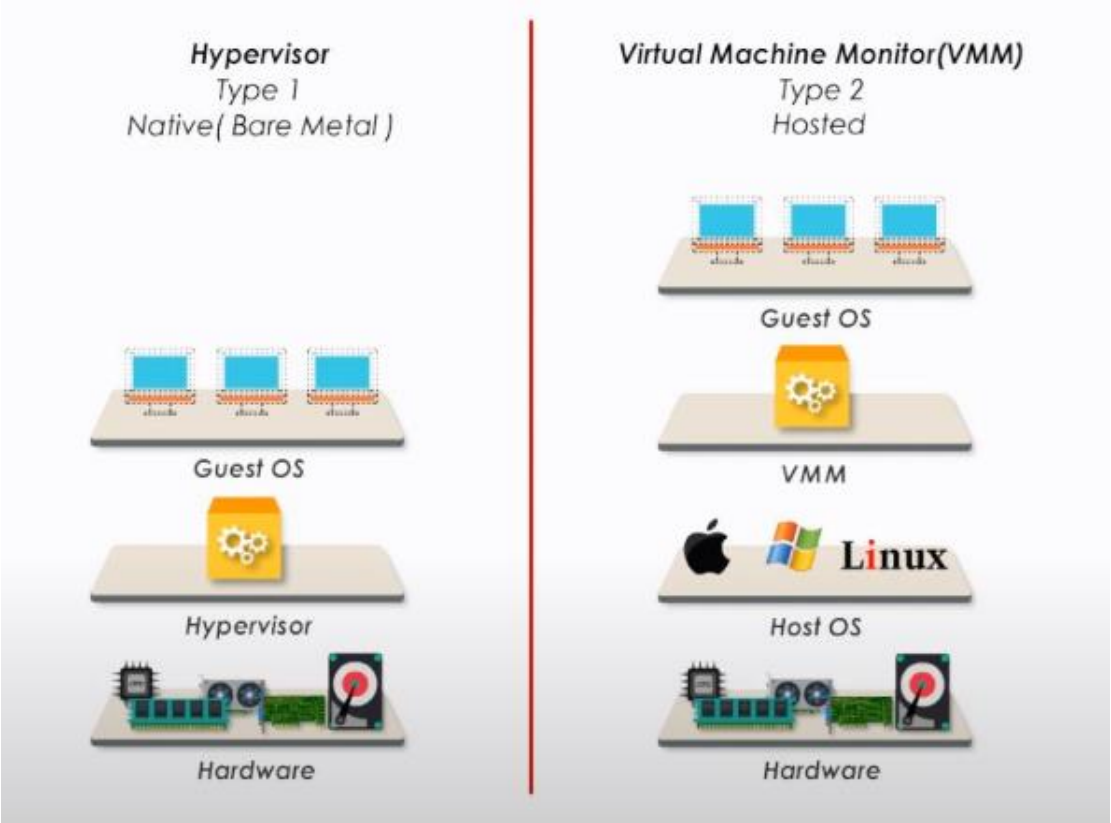
Πριν τους Hypervisors, οι περισσότεροι φυσικοί υπολογιστές μπορούσαν να τρέξουν μόνο ένα λειτουργικό σύστημα, δηλαδή το υλικό (hardware) χειριζόταν αιτήματα μόνο από το

συγκεκριμένο λειτουργικό σύστημα. Το μειονέκτημα αυτής της προσέγγισης ήταν ότι σπαταλούσε πόρους καθώς το λειτουργικό σύστημα δεν μπορούσε πάντα να χρησιμοποιήσει όλη την υπολογιστική του ισχύ. Οι Hypervisors λύνουν αυτό το πρόβλημα. Πρόκειται για ένα μικρό στρώμα λογισμικού που επιτρέπει σε πολλαπλά λειτουργικά συστήματα να τρέχουν παράλληλα, μοιράζοντας τους ίδιους φυσικούς πόρους υπολογιστών. Αυτά τα λειτουργικά συστήματα είναι τα εικονικά μηχανήματα (VMs), τα οποία μιμούνται ένα ολόκληρο περιβάλλον υπολογιστικού υλικού στο λογισμικό. Ο Hypervisor, ο οποίος είναι γνωστός και ως εικονική μηχανή παρακολούθησης (VMM), διαχειρίζεται τα VMs καθώς τρέχουν παράλληλα. Διαχωρίζει τα VM μεταξύ τους λογικά, αναθέτοντας στον καθένα τη δική του φέτα της υποκείμενης υπολογιστικής ισχύος, της μνήμης και της αποθήκευσης. Αυτό εμποδίζει τα VM να παρεμβαίνουν μεταξύ τους. οπότε αν, για παράδειγμα, ένα λειτουργικό σύστημα υποστεί συντριβή ή συμβιβασμό ασφαλείας, οι άλλοι επιβιώνουν.

Υπάρχουν δύο μεγάλες κατηγορίες, οι Hypervisors Type 1 και οι Hypervisors Type 2.

- Ένας Hypervisors Type 1, τρέχει απευθείας στο φυσικό υλικό του υποκείμενου υπολογιστή, αλληλεπιδρώντας άμεσα με την CPU, τη μνήμη και την φυσική αποθήκευση. Για το λόγο αυτό, οι Type 1 Hypervisors αναφέρονται επίσης ως υπερμετασχηματιστές γυμνού μεταλλικού στοιχείου(bare-metal Hypervisors). Ένας Type 1 Hypervisor αντικαθιστά το λειτουργικό σύστημα του κεντρικού υπολογιστή. Οι Hypervisors Type 1 είναι εξαιρετικά αποδοτικοί επειδή έχουν άμεση πρόσβαση στο φυσικό υλικό. Αυτό αυξάνει επίσης την ασφάλειά τους, καθώς δεν υπάρχει τίποτα μεταξύ αυτών και της CPU, που ο επιτιθέμενος θα μπορούσε να θέσει σε κίνδυνο. Όμως ένας Type 1 Hypervisor χρειάζεται συχνά μια ξεχωριστή μηχανή διαχείρισης για τη διαχείριση διαφορετικών VM και τον έλεγχο του υλικού του κεντρικού υπολογιστή.
- Ένας Hypervisor Type 2 δεν τρέχει απευθείας στο υποκείμενο υλικό. Αντίθετα, τρέχει ως εφαρμογή σε ένα λειτουργικό σύστημα. Σπάνια εμφανίζονται σε περιβάλλοντα που βασίζονται σε διακομιστές (server-based environments). Αντ' αυτού, είναι κατάλληλα για μεμονωμένους χρήστες υπολογιστών που χρειάζονται να εκτελούν πολλαπλά λειτουργικά συστήματα. Παραδείγματα αποτελούν οι μηχανικοί, επαγγελματίες ασφαλείας που αναλύουν κακόβουλα λογισμικά και διάφοροι άλλοι επιχειρηματικοί χρήστες που χρειάζονται πρόσβαση σε εφαρμογές που είναι διαθέσιμες μόνο σε άλλες πλατφόρμες λογισμικού. Οι Hypervisor Type 2 διαθέτουν συχνά πρόσθετα εργαλεία (toolkits) για τους χρήστες ώστε να εγκαταστήσουν στο λειτουργικό σύστημα επισκεπτών (Guest OS). Αυτά τα εργαλεία παρέχουν βελτιωμένες συνδέσεις μεταξύ του φιλοξενούμενου(Guest OS) και του κεντρικού λειτουργικού συστήματος(Host), επιτρέποντας έτσι ενέργειες όπως η πρόσβαση σε αρχεία και φακέλους του Host μέσα από το Guest VM ή τη δυνατότητα «cut and paste» μεταξύ των δύο, Host και Guest. Ένας Type 2 Hypervisor επιτρέπει γρήγορη και εύκολη πρόσβαση σε ένα εναλλακτικό λειτουργικό σύστημα φιλοξενουμένων (Guest OS) παράλληλα με το πρωτεύον(Host) που λειτουργεί στο σύστημα κεντρικού υπολογιστή. Αυτό το καθιστά εξαιρετικό για την παραγωγικότητα των τελικών χρηστών. Παραδείγματος χάρη, ένας χρήστης μπορεί να το χρησιμοποιήσει για να αποκτήσει πρόσβαση στα εργαλεία ανάπτυξης που χρησιμοποιεί, τα οποία βασίζονται στο Linux,

χρησιμοποιώντας ένα σύστημα υπαγόρευσης ομιλίας που βρίσκεται στα Windows. Ένας Type 2 Hypervisor όμως, πρέπει να έχει πρόσβαση σε μνήμη και πόρους δικτύου μέσω του κεντρικού λειτουργικού συστήματος (Host OS), το οποίο έχει πρωτεύουσα πρόσβαση στο φυσικό μηχάνημα (Host). Αυτό δημιουργεί πρόβλημα καθυστέρησης (latency issue) επηρεάζοντας την απόδοση. Δημιουργεί επίσης πιθανούς κινδύνους για την ασφάλεια εάν ένας εισβολέας υπονομεύσει το λειτουργικό σύστημα του κεντρικού υπολογιστή (Host OS), καθώς θα είχε την δυνατότητα να χειριστεί οποιοδήποτε λειτουργικό σύστημα επισκέπτη (Guest OS) λειτουργεί στον Hypervisor Type 2.



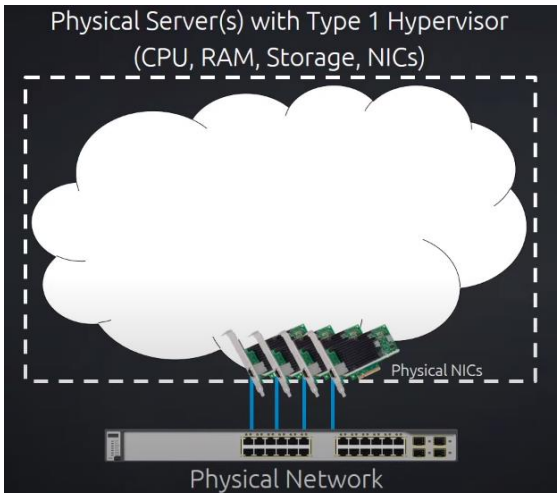
1.3 Εικονικοποιημένη δικτύωση

Η εικονικοποίηση δικτύου (NV) αναφέρεται στην απόσυρση πόρων δικτύου που παραδίδονται παραδοσιακά σε υλικό σε λογισμικό. Το NV μπορεί να συνδυάσει πολλά φυσικά δίκτυα σε ένα εικονικό δίκτυο που βασίζεται σε λογισμικό ή μπορεί να χωρίσει ένα φυσικό δίκτυο σε ξεχωριστά, ανεξάρτητα εικονικά δίκτυα. Η NV αποσυνδέει υπηρεσίες δικτύου από το υποκείμενο υλικό και επιτρέπει την εικονική παροχή ολόκληρου του δικτύου. Οι φυσικοί πόροι δικτύου, όπως διακόπτες (Switches) και δρομολογητές (Routers), συγκεντρώνονται και είναι προσβάσιμοι από οποιονδήποτε χρήστη μέσω ενός κεντρικού συστήματος διαχείρισης. Η εικονικοποίηση δικτύου επιτρέπει επίσης την αυτοματοποίηση πολλών διοικητικών εργασιών, τη μείωση των χειροκίνητων σφαλμάτων και τον χρόνο παροχής. Μπορεί να προσφέρει μεγαλύτερη παραγωγικότητα και αποδοτικότητα δικτύου. Υπάρχουν δύο τύποι εικονοποιημένης δικτύωσης:

- Εξωτερική εικονικοποίηση (External Virtualization) : Συνδυάζει πολλά δίκτυα ή τμήματα δικτύων σε μια εικονική μονάδα.
- Εσωτερική εικονικοποίηση (Internal virtualization): Χρησιμοποιεί «κοντέινερ» λογισμικού (software containers)για να μιμείται ή να παρέχει τη λειτουργικότητα ενός μεμονωμένου φυσικού δικτύου.

Παράδειγμα εικονοποιημένης δικτύωσης

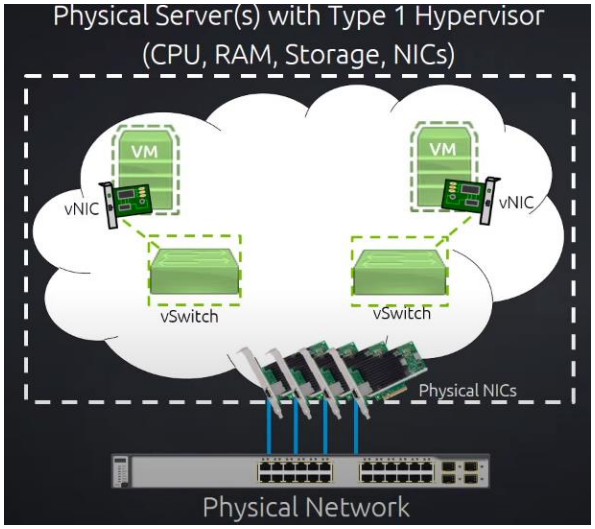
Διαθέτουμε έναν φυσικό διακομιστή (physical server) με τέσσερις κάρτες διασύνδεσης δικτύου (Network Interface Cards) εγκατεστημένες, οι οποίες συνδέονται στο φυσικό μας δίκτυο. Ο φυσικός διακομιστής διαθέτει και CPU, RAM, αποθηκευτικό χώρο (storage). Επίσης, έχουμε εγκαταστήσει έναν επόπτη τύπου 1 (Type 1 Hypervisor).



Εικόνα 3: Virtualized networking

Αναπόσπαστο «κομμάτι» της εικονοποιημένης δικτύωσης είναι ένας εικονικός διακόπτης (Virtual switch). Οι vSwitch που διαθέτουμε συνδέονται με το φυσικό μας δίκτυο (Physical NICs) και με αυτό τον τρόπο το φυσικό μας δίκτυο (Physical Network) είναι συνδεδεμένο στο εικονικό μας δίκτυο (Virtual Network).

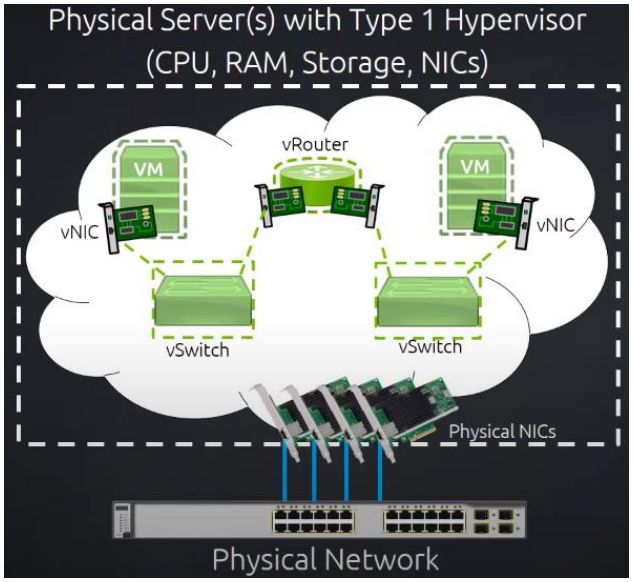
Μέσα στον Server μας έχουμε τη δυνατότητα να δημιουργήσουμε πολλαπλές εικονικές μηχανές (Virtual Machines). Μόλις δημιουργήσουμε το εικονικό μας μηχανήματα (Virtual Machine) πρέπει να δημιουργήσουμε ένα εικονικό NIC (Virtual Network Interface Card, vNIC). Συγκεντρωτικά, η εικονική μας μηχανή διαθέτει μια vNIC η οποία είναι συνδεδεμένη με το vSwitch, το οποίο είναι συνδεδεμένο (mapped) με τις φυσικές μας NICs (Physical NICs). Όλα τα παραπάνω αποτελούν τη βασική αρχή της εικονοποιημένης δικτύωσης (Virtualized networking).



Εικόνα 4: Virtualized networking

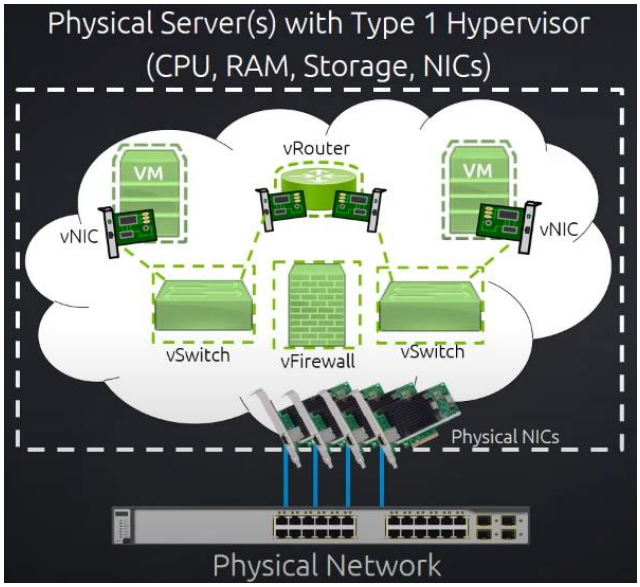
Στο προαναφερθέν παράδειγμα δρομολογήσαμε δύο εικονικά δίκτυα στο φυσικό μας δίκτυο ώστε να υπάρχει επικοινωνία μεταξύ τους. Παρακάτω θα αναλύσουμε πώς είναι δυνατό αυτή η δρομολόγηση να επιτευχθεί εντός του εικονοποιημένου περιβάλλοντος.

Για να επιτευχθεί η δρομολόγηση δύο εικονικών δικτύων εντός του εικονικού περιβάλλοντος χρησιμοποιείται ένας εικονικός δρομολογητής (Virtual Router, vRouter). Ο εικονικός δρομολογητής συνδέεται χρησιμοποιώντας δύο vNICs στους δύο vSwitch και παρέχει δρομολόγηση μεταξύ των δύο διαφορετικών εικονικών δικτύων



Εικόνα 5: Virtualized networking

Κλείνοντας, να αναφέρουμε πως έχουμε τη δυνατότητα να εγκαταστήσουμε ένα εικονικό «τείχος προστασίας» (Virtual Firewall, vFirewall), το οποίο μπορούμε να το εγκαταστήσουμε έτσι ώστε να παρέχει τις υπηρεσίες του μεταξύ των διαφορετικών εικονικών δικτύων ή και ως front-end βγαίνοντας προς το φυσικό δίκτυο



Εικόνα 6: Virtualized networking

2. Κατηγορίες Εικονικοποίησης

Αν και κάποια μορφή εικονικοποίησης υπήρχε από τα μέσα της δεκαετίας του 1960, εξελίχθηκε με την πάροδο του χρόνου, παραμένοντας όμως κοντά στις ρίζες της. Μεγάλο μέρος της εξέλιξης στην εικονικοποίηση έχει συμβεί τα τελευταία χρόνια, με νέους τύπους να αναπτύσσονται και να εμπορευματοποιούνται. Οι διαφορετικοί τύποι εικονικοποίησης περιορίζονται σε εικονικοποίηση επιφάνειας εργασίας (Desktop Virtualization), εικονικοποίηση εφαρμογών (Application Virtualization), εικονικοποίηση διακομιστή (Server Virtualization), εικονικοποίηση αποθήκευσης (Storage Virtualization) και εικονικοποίηση δικτύου (Network Virtualization).

2.1 Εικονικοποίηση επιφάνειας εργασίας

Ένα λειτουργικό σύστημα επιφάνειας εργασίας (OS), όπως παραδείγματος χάρη, τα Windows 7, θα λειτουργεί ως εικονική μηχανή (VM) σε έναν φυσικό διακομιστή (Physical Server) με άλλους εικονικούς επιτραπέζιους υπολογιστές (Virtual Desktops). Η επεξεργασία πολλαπλών εικονικών επιτραπέζιων υπολογιστών πραγματοποιείται σε έναν ή μερικούς φυσικούς διακομιστές (Physical Server), συνήθως στο κεντρικό κέντρο δεδομένων. Το αντίγραφο του λειτουργικού συστήματος και των εφαρμογών που χρησιμοποιεί κάθε τελικός χρήστης θα αποθηκεύεται συνήθως στη μνήμη ως μία εικόνα (image) στον φυσικό διακομιστή (Physical Server). Ένα από τα μεγαλύτερα πλεονεκτήματα της εικονικοποίησης της επιφάνειας εργασίας είναι ότι οι χρήστες μπορούν να έχουν πρόσβαση σε όλα τα προσωπικά τους αρχεία και εφαρμογές σε οποιονδήποτε υπολογιστή, πράγμα που σημαίνει ότι μπορούν να εργαστούν από οπουδήποτε χωρίς να χρειάζεται να φέρουν τον υπολογιστή εργασίας τους. Μειώνει επίσης το κόστος αδειοδότησης λογισμικού και ενημερώσεων. Η συντήρηση και η διαχείριση ενημερώσεων κώδικα είναι απλές, καθώς όλοι οι εικονικοί υπολογιστές φιλοξενούνται στην ίδια τοποθεσία.

Η εικονική υποδομή επιφάνειας εργασίας (Virtual desktop infrastructure, VDI) ορίζεται ως η φιλοξενία των επιτραπέζιων υπολογιστών σε κεντρικό διακομιστή. Είναι μια μορφή εικονικοποίησης επιφάνειας εργασίας, καθώς οι συγκεκριμένες εικόνες επιφάνειας εργασίας εκτελούνται σε εικονικές μηχανές (VM) και παραδίδονται σε τελικούς πελάτες μέσω ενός δικτύου. Αυτά τα τελικά σημεία μπορεί να είναι υπολογιστές ή άλλες συσκευές, όπως tablet ή τερματικά.

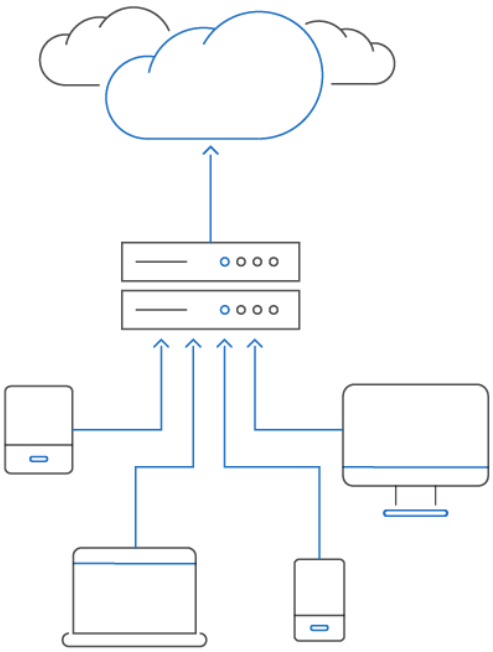
Σε όλες τις εφαρμογές VDI, ισχύουν τα ακόλουθα χαρακτηριστικά:

1. Οι εικονικοί επιτραπέζιοι υπολογιστές (virtual desktops) «ζουν» εντός VM σε έναν κεντρικό διακομιστή (centralized Server).
2. Κάθε εικονική επιφάνεια εργασίας περιλαμβάνει μια εικόνα λειτουργικού συστήματος (Operation System Image), συνήθως Microsoft Windows.
3. Τα VM βασίζονται σε κεντρικούς υπολογιστές (host-based), που σημαίνει ότι πολλές παρουσίες μπορούν να φιλοξενηθούν στον ίδιο διακομιστή στο κέντρο δεδομένων.
4. Οι τελικοί πελάτες (end client) πρέπει να συνδέονται συνεχώς με τον κεντρικά διαχειριζόμενο διακομιστή, προκειμένου να διατηρείται η πρόσβαση στους εικονοποιημένους επιτραπέζιους υπολογιστές που φιλοξενεί.
5. Ο «μεσίτης σύνδεσης» της εφαρμογής VDI (VDI implementation's Connection broker) βρίσκει μια εικονική επιφάνεια εργασίας μέσα στο

χώρο συγκέντρωσης πόρων, για κάθε πελάτη, για σύνδεση με την επιτυχή πρόσβασή του στο περιβάλλον VDI.

6. Εν τω μεταξύ, ένας Hypervisor, δημιουργεί, εκτελεί και διαχειρίζεται τα διάφορα VMs που ενσωματώνουν τα μεμονωμένα περιβάλλοντα εικονικής επιφάνειας εργασίας

Στους σύγχρονους ψηφιακούς χώρους εργασίας στους οποίους απαιτείται πρόσβαση σε πολλές εφαρμογές κατ 'απαίτηση, το VDI διευκολύνει την ασφαλή και βολική απομακρυσμένη πρόσβαση που βοηθά στην αύξηση της παραγωγικότητας των εργαζομένων.



Εικόνα 7: VDI

Δεν χρησιμοποιούν όλοι οι τύποι εικονικοποίησης επιφάνειας εργασίας VM που βασίζονται σε κεντρικούς υπολογιστές όπως το VDI. Επίσης, το VDI δεν είναι συνώνυμο με την εικονικοποίηση της επιφάνειας εργασίας ως κατηγορία. Αντ 'αυτού, είναι μια εναλλακτική λύση σε άλλες μορφές παράδοσης εικονικής επιφάνειας εργασίας, συμπεριλαμβανομένων των φιλοξενούμενων κοινόχρηστων λύσεων που συνδέουν υπολογιστές και πελάτες σε κοινόχρηστη επιφάνεια εργασίας, καθώς και τοπική εικονικοποίηση επιφάνειας εργασίας, στην οποία το περιβάλλον της επιφάνειας εργασίας (local desktop virtualization) εκτελείται απευθείας στον πελάτη (client).

Υποθέτοντας ότι όλες οι υποστηρικτικές υποδομές λειτουργούν όπως προορίζεται, ένας χρήστης που αποκτά πρόσβαση σε μια λύση VDI από το τελικό σημείο μπορεί να διαχειριστεί το λειτουργικό σύστημα και τις εφαρμογές και τα δεδομένα σε αυτό σαν να λειτουργούσαν τοπικά. Αυτή η εγκατάσταση επιτρέπει στους εργαζόμενους να έχουν ασφαλή πρόσβαση σε ό, τι χρειάζονται σχεδόν από οποιαδήποτε συσκευή, χωρίς να απαιτούν συγκεκριμένο υλικό. Εμπλουτισμένο με λύσεις για βολική απλή σύνδεση (SSO) και ασφαλή απομακρυσμένη πρόσβαση, οι εικονικοί επιτραπέζιοι υπολογιστές μπορούν επίσης να εκτελούνται και να διαχειρίζονται παράλληλα με το αυξανόμενο φάσμα εφαρμογών cloud, web και mobile που είναι

ενσωματωμένο στις σύγχρονες ροές εργασίας. Οι εργαζόμενοι έχουν μια ενοποιημένη εμπειρία που επιτρέπει μεγαλύτερη παραγωγικότητα, ενώ η πληροφορική αποφεύγει τα σιλό και μετριάζει τον κίνδυνο μη εξουσιοδοτημένων συνδέσεων. Με άλλα λόγια, το VDI στο πλαίσιο μιας πλατφόρμας ψηφιακού χώρου εργασίας συμβάλλει σε ένα ανώτερο εργασιακό περιβάλλον χωρίς συμβιβασμούς στην ασφάλεια.

Υπάρχουν δύο υλοποιήσεις VDI:

«Επίμονο» (Persistent) VDI:

1. Σε έναν χρήστη εκχωρείται μια τυποποιημένη επιφάνεια εργασίας από την ομάδα πόρων την πρώτη φορά που συνδέεται
2. Κάθε επόμενη φορά που αποκτούν πρόσβαση στο περιβάλλον VDI, συνδέονται στην ίδια επιφάνεια εργασίας, με όλες τις αλλαγές τους να διατηρούνται στην εικόνα του εικονικού λειτουργικού συστήματος ακόμα και μετά την επανεκκίνηση της σύνδεσης.
3. Για εργαζόμενους με πολύπλοκες και γρήγορες ψηφιακές ροές εργασίας, αυτό σημαίνει ότι μπορούν εύκολα να παραλάβουν από εκεί που σταμάτησαν και να επωφεληθούν από την εκτενή εξατομίκευση των εικονικών εφαρμογών και ρυθμίσεων της επιφάνειας εργασίας.

«Μη ανθεκτικό» (Non -Persistent) VDI:

1. Ο τελικός πελάτης μπορεί να συνδέεται στην ίδια επιφάνεια εργασίας κάθε φορά ή σε έναν τυχαίο από την ομάδα.
2. Σε κάθε περίπτωση, δεν αποθηκεύονται αλλαγές κατά την επανεκκίνηση.
3. Μια μη συνεχής εφαρμογή VDI είναι κατάλληλη για εφάπαξ πρόσβαση σε επιτραπέζιο υπολογιστή, αλλά όχι για χρήση εικονικής επιφάνειας εργασίας ακριβώς όπως ένα προσωπικό φυσικό ισοδύναμο, το οποίο απαιτεί «επίμονο» Persistent VDI.

Δεδομένου ότι τίποτα δεν αποθηκεύεται μετά τη διακοπή της σύνδεσης, το IT δεν χρειάζεται να διατηρεί μεγάλο αριθμό προσαρμοσμένων εικόνων λειτουργικού συστήματος, επιτρέποντας απλοποιημένη διαχείριση του κέντρου δεδομένων και μειωμένο κόστος. Το «Μη ανθεκτικό» (Non - Persistent) VDI βελτιστοποιεί επίσης τη διαχείριση συσκευών για εργαζόμενους που δεν χρειάζεται να αποθηκεύσουν τίποτα.

Το VDI υποστηρίζει βελτιωμένη κινητικότητα χρήστη και απομακρυσμένη πρόσβαση, καθώς μπορεί να επιτευχθεί μια τυποποιημένη επιφάνεια εργασίας από σχεδόν οποιοδήποτε εγκεκριμένο και συμβατό τελικό σημείο σε οποιαδήποτε τοποθεσία. Για εργαζόμενους που βρίσκονται συχνά εν κινήσει και χρειάζονται μια εικονική επιφάνεια εργασίας που περιέχει ένα πλήρες φάσμα εικονικών εφαρμογών και δεδομένων, το VDI είναι σαν να υπάρχει διαθέσιμο γραφείο κατ 'απαίτηση. Από αυτήν την άποψη, ταιριάζει απευθείας στις ροές εργασίας του ψηφιακού χώρου εργασίας που διαθέτουν ήδη παρόμοια, τακτική κατανάλωση εφαρμογών cloud, ιστού και κινητών σε πολλά περιβάλλοντα, ειδικά αν είναι «επίμονο» (Persistent) VDI.

Από την πλευρά της ασφάλειας, το VDI προσφέρει κάποια βελτίωση σε σχέση με τη λειτουργία ενός λειτουργικού συστήματος και όλων των λειτουργιών σε τοπικό επίπεδο. Όλα τα δεδομένα από μια σύνδεση VDI «ζουν» στον διακομιστή, όχι στον τελικό πελάτη, πράγμα που σημαίνει ότι εάν το τελικό σημείο έχει κλαπεί, δεν υπάρχει τίποτα για κλοπή από την τοπική του αποθήκευση. Επιπλέον, το περιβάλλον VDI ελέγχεται πλήρως και κεντρικά από ένα κέντρο δεδομένων. Οι διαχειριστές μπορούν να εφαρμόσουν ενημερώσεις κώδικα και ενημερώσεις λογισμικού, να αλλάξουν τις διαμορφώσεις και να επιβάλουν πολιτικές για όλους τους εικονικούς επιτραπέζιους υπολογιστές. Με αυτόν τον τρόπο, το VDI επιτρέπει τον ακριβή έλεγχο και την ασφαλή απομόνωση των εικόνων λειτουργικού συστήματος από έναν κεντρικό διακομιστή, η οποία είναι μια λιγότερο περίπλοκη ρύθμιση που διαχειρίζεται φορητούς υπολογιστές που εκτελούν τοπικά τα λειτουργικά τους συστήματα.

2.2 Εικονικοποίηση εφαρμογών

Η εικονικοποίηση εφαρμογών είναι η τεχνολογία που επιτρέπει στους χρήστες να έχουν πρόσβαση και να χρησιμοποιούν μια εφαρμογή από έναν ξεχωριστό υπολογιστή από αυτόν στον οποίο είναι εγκατεστημένη η εφαρμογή. Χρησιμοποιώντας λογισμικό εικονικοποίησης εφαρμογών, οι διαχειριστές IT μπορούν να ρυθμίσουν απομακρυσμένες εφαρμογές σε έναν διακομιστή και στη συνέχεια να παραδώσουν τις εφαρμογές στον υπολογιστή, το κινητό ή το tablet ενός τελικού χρήστη. Για τον χρήστη, η εμπειρία της εικονοποιημένης εφαρμογής είναι ίδια με τη χρήση της εγκατεστημένης εφαρμογής σε ένα φυσικό μηχάνημα.

Ο πιο συνηθισμένος τρόπος εικονικοποίησης εφαρμογών είναι η προσέγγιση που βασίζεται σε διακομιστή. Αυτό σημαίνει ότι ένας διαχειριστής πληροφορικής εφαρμόζει απομακρυσμένες εφαρμογές σε διακομιστή μέσα στο κέντρο δεδομένων ενός οργανισμού ή μέσω μιας υπηρεσίας φιλοξενίας. Στη συνέχεια, ο διαχειριστής IT χρησιμοποιεί λογισμικό εικονικοποίησης εφαρμογών για να παραδώσει τις εφαρμογές στην επιφάνεια εργασίας ενός χρήστη ή σε άλλη συνδεδεμένη συσκευή. Ο χρήστης μπορεί πλέον να έχει πρόσβαση και να χρησιμοποιήσει την εφαρμογή σαν να ήταν τοπικά εγκατεστημένη στον υπολογιστή του και οι ενέργειες του χρήστη μεταφέρονται πίσω στον διακομιστή για εκτέλεση. Η εικονικοποίηση εφαρμογών είναι ένα σημαντικό μέρος των ψηφιακών χώρων εργασίας και της εικονικοποίησης επιφάνειας εργασίας.

Εικόνα 8: Application Virtualization

Οφέλη της εικονικοποίησης εφαρμογών:

- Διαχείριση εφαρμογών: Η εικονικοποίηση εφαρμογών διευκολύνει πολύ τα τμήματα πληροφορικής να διαχειρίζονται και να συντηρούν εφαρμογές σε έναν οργανισμό. Αντί να εγκαταστήσετε χειροκίνητα εφαρμογές σε κάθε μηχανή χρηστών, η εικονικοποίηση εφαρμογών επιτρέπει στους διαχειριστές IT να εγκαταστήσουν μια εφαρμογή μία φορά σε έναν κεντρικό διακομιστή και στη συνέχεια να αναπτύξουν την εφαρμογή όπως απαιτείται σε συσκευές χρηστών. Εκτός από την εξοικονόμηση χρόνου εγκατάστασης, αυτό καθιστά επίσης απλούστερη την ενημέρωση ή την ενημέρωση εφαρμογών, επειδή το IT πρέπει να το κάνει μόνο σε έναν διακομιστή.

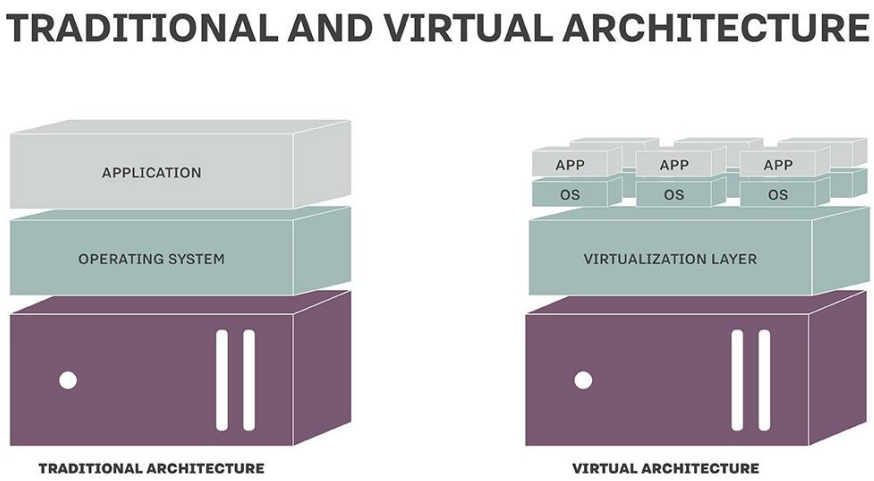
- Επεκτασιμότητα: Η εικονικοποίηση εφαρμογών επιτρέπει στους διαχειριστές IT να αναπτύξουν εικονικές εφαρμογές σε όλα τα είδη συνδεδεμένων συσκευών, ανεξάρτητα από τα λειτουργικά συστήματα ή τον αποθηκευτικό χώρο αυτών των συσκευών. Αυτό επιτρέπει τη παροχή πελάτη στην οποία οι χρήστες έχουν πρόσβαση σε μια εφαρμογή σε μηχανή χαμηλού κόστους, ενώ οι κεντρικοί διακομιστές χειρίζονται όλη την υπολογιστική ισχύ που απαιτείται για την εκτέλεση αυτής της εφαρμογής. Ως αποτέλεσμα, ο οργανισμός ξοδεύει πολύ λιγότερο για υπολογιστικό υλικό, επειδή οι εργαζόμενοι απαιτούν μόνο βασικά μηχανήματα για πρόσβαση στις εφαρμογές που χρειάζονται για εργασία. Η εικονικοποίηση εφαρμογών επιτρέπει επίσης στους χρήστες να έχουν πρόσβαση σε εφαρμογές που κανονικά δεν θα λειτουργούσαν στο λειτουργικό σύστημα των μηχανημάτων τους, επειδή η εφαρμογή εκτελείται στην πραγματικότητα στον κεντρικό διακομιστή. Αυτό χρησιμοποιείται συνήθως για την εκτέλεση μιας εφαρμογής Windows σε λειτουργικό σύστημα Linux.

- Ασφάλεια: Το λογισμικό εικονικοποίησης εφαρμογών δίνει στους διαχειριστές πληροφορικής κεντρικό έλεγχο για το ποιοι χρήστες μπορούν να έχουν πρόσβαση σε ποιες εφαρμογές. Εάν αλλάξουν τα δικαιώματα εφαρμογής ενός χρήστη σε έναν οργανισμό, ο διαχειριστής IT μπορεί απλώς να καταργήσει την πρόσβαση αυτού του χρήστη σε μια εφαρμογή. Χωρίς εικονικοποίηση εφαρμογής, ο διαχειριστής IT θα πρέπει να απεγκαταστήσει φυσικά την εφαρμογή από τη συσκευή του χρήστη. Αυτός ο κεντρικός έλεγχος πρόσβασης στην εφαρμογή είναι ιδιαίτερα σημαντικός εάν οι συσκευές ενός χρήστη χαθούν ή κλαπούν, επειδή ο διαχειριστής IT μπορεί να ανακαλέσει την απομακρυσμένη πρόσβαση σε ευαίσθητα δεδομένα χωρίς να χρειάζεται να εντοπίσει τη συσκευή που λείπει.

2.3 Εικονικοποίηση Υλικού

Πρόκειται για την αφαίρεση υπολογιστικών πόρων από το λογισμικό που χρησιμοποιεί αυτούς τους πόρους. Η εικονικοποίηση υλικού εγκαθιστά έναν hypervisor ή έναν διαχειριστή εικονικής μηχανής (VMM), ο οποίος δημιουργεί ένα επίπεδο αφαίρεσης (abstraction layer) μεταξύ του λογισμικού και του υποκείμενου υλικού. Μόλις δημιουργηθεί ο hypervisor, το λογισμικό βασίζεται σε εικονικές αναπαραστάσεις των υπολογιστικών στοιχείων, όπως οι εικονικοί επεξεργαστές και όχι οι φυσικοί επεξεργαστές. Οι δημοφιλείς hypervisors περιλαμβάνουν το vSphere του VMware,

βασισμένο στο ESXi και το Hyper-V της Microsoft. Οι εικονικοποιημένοι υπολογιστικοί πόροι παρέχονται σε απομονωμένα VM, όπου μπορούν να εγκατασταθούν λειτουργικά συστήματα και εφαρμογές. Τα εικονικοποιημένα συστήματα μπορούν να φιλοξενήσουν πολλαπλά VM ταυτόχρονα, αλλά κάθε VM είναι απομονωμένο από κάθε άλλο VM. Αυτό σημαίνει ότι μια επίθεση από κακόβουλο λογισμικό ή ένα σφάλμα ενός VM δεν θα επηρεάσει τα άλλα VM. Η υποστήριξη πολλαπλών VM αυξάνει σημαντικά τη χρήση και την αποδοτικότητα του συστήματος. Αυτή η βελτιωμένη χρήση υλικού είναι ένα σημαντικό πλεονέκτημα της εικονικοποίησης και υποστηρίζει τεράστιες δυνατότητες ενοποίησης του συστήματος, μειώνοντας τον αριθμό των διακομιστών και τη χρήση ισχύος σε κέντρα δεδομένων επιχειρήσεων.



Εικόνα 9: The virtualization layer abstracts resources from the underlying hardware.

Δεδομένου ότι ένας Hypervisor ή VMM είναι εγκατεστημένος απευθείας σε υπολογιστικό υλικό και άλλα λειτουργικά συστήματα και οι εφαρμογές εγκαθίστανται αργότερα, η εικονικοποίηση υλικού συχνά αναφέρεται ως bare-metal virtualization. Αυτό οδήγησε τους Hypervisors να θεωρούνται λειτουργικά από μόνα τους, αν και ένας εικονικοποιημένος διακομιστής συνήθως θα αναπτύξει ένα VM με ένα κεντρικό λειτουργικό σύστημα, όπως το Windows Server 2012 R2 και τα εργαλεία διαχείρισης θα εκτελέσουν τον διακομιστή πριν δημιουργήσουν άλλα VM για να φιλοξενήσουν πραγματικούς φόρτους εργασίας. Η εναλλακτική λύση σε μια προσέγγιση γυμνού μετάλλου περιλαμβάνει την εγκατάσταση ενός κεντρικού λειτουργικού συστήματος πρώτα και στη συνέχεια την εγκατάσταση ενός hypervisor πάνω από το λειτουργικό σύστημα του κεντρικού υπολογιστή. Αυτό είναι γνωστό ως εικονικοποίηση κεντρικού υπολογιστή (host virtualization) και έχει εγκαταλειφθεί σε μεγάλο βαθμό για VM, αν και η σύγχρονη εικονικοποίηση κοντέινερ (container virtualization) έχει αναστήσει αυτήν την προσέγγιση. Πέρα από τη βελτιωμένη χρήση του υπολογιστικού υλικού, η εικονικοποίηση βελτιώνει επίσης την ευελιξία στην ανάπτυξη εφαρμογών και την προστασία. Με έναν κοινό Hypervisor, οι VM δεν συνδέονται πλέον με έναν διακομιστή με τον τρόπο που μια φυσική εφαρμογή μπορεί να συνδεθεί με μια παραδοσιακή εγκατάσταση διακομιστή. Αντ 'αυτού, ένα VM σε έναν διακομιστή μπορεί να μετεγκατασταθεί σε άλλο εικονικοποιημένο διακομιστή στο τοπικό κέντρο δεδομένων ή διακομιστές σε οποιαδήποτε απομακρυσμένη τοποθεσία, ενώ η εφαρμογή εξακολουθεί να εκτελείται. Αυτή η ζωντανή μετεγκατάσταση επιτρέπει τη μετακίνηση των VM, όπως απαιτείται, για τον εξορθολογισμό της απόδοσης του διακομιστή, σε περίπτωση

εξισορρόπησης φορτίου ή την ανακούφιση ενός διακομιστή από τους φόρτους εργασίας του, προκειμένου να αντικαταστήσει ή να συντηρήσει το σύστημα, αλλά να μην διαταράξει τις εφαρμογές, οι οποίες μπορούν να συνεχίσουν να εκτελούνται σε άλλα συστήματα. Επιπλέον, τα VM μπορούν να προστατευτούν με αντίγραφα ασφαλείας και στιγμιότυπα χρονικής στιγμής, τα οποία μπορούν και τα δύο να αποκατασταθούν σε οποιονδήποτε εικονικοποιημένο διακομιστή, ανεξάρτητα από το υποκείμενο υλικό.

Υπάρχουν πολλοί τύποι εικονικοποίησης υλικού, με διαδικασίες που περιλαμβάνουν πλήρη εικονικοποίηση (full virtualization), paravirtualization και εικονικοποίηση υποβοηθούμενη από υλικό (hardware-assisted virtualization).

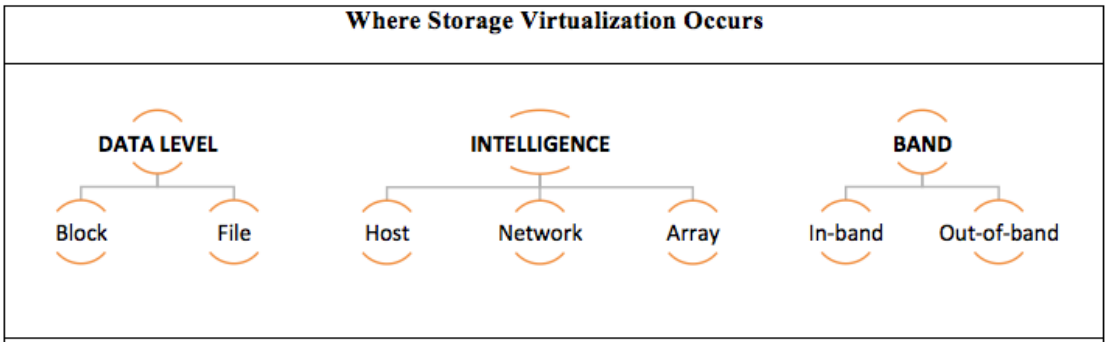
- Πλήρης εικονικοποίηση (full virtualization): Προσομοιώνει πλήρως το υλικό για να επιτρέψει σε ένα λειτουργικό σύστημα επισκέπτη (Guest OS) να εκτελείται σε απομονωμένη παρουσία. Σε μια πλήρως εικονικοποιημένη περίπτωση, μια εφαρμογή θα εκτελείται πάνω από ένα λειτουργικό σύστημα επισκέπτη, το οποίο θα λειτουργούσε πάνω από τον επόπτη και, τέλος, το λειτουργικό σύστημα και το υλικό του κεντρικού υπολογιστή. Η πλήρης εικονικοποίηση δημιουργεί ένα περιβάλλον παρόμοιο με ένα λειτουργικό σύστημα που λειτουργεί σε έναν μεμονωμένο διακομιστή. Η αξιοποίηση της πλήρους εικονικοποίησης επιτρέπει στους διαχειριστές να εκτελούν ένα εικονικό περιβάλλον αμετάβλητο με το φυσικό του αντίστοιχο. Η πλήρης εικονικοποίηση επιτρέπει στους διαχειριστές να συνδυάζουν τόσο υπάρχοντα όσο και νέα συστήματα. Ωστόσο, κάθε δυνατότητα που διαθέτει το υλικό πρέπει επίσης να εμφανίζεται σε κάθε VM για να θεωρείται η διαδικασία πλήρης εικονικοποίηση. Αυτό σημαίνει, για την ενσωμάτωση παλαιότερων συστημάτων, το υλικό πρέπει να αναβαθμιστεί ώστε να ταιριάζει με τα νεότερα συστήματα.
- Paravirtualization: Το Paravirtualization θα εκτελέσει μια τροποποιημένη και ανακατασκευασμένη έκδοση των λειτουργικών επισκεπτών σε ένα VM. Αυτή η τροποποίηση επιτρέπει στο VM να διαφέρει κάπως από το υλικό. Το υλικό δεν προσομοιώνεται απαραίτητα στην παραθυροποίηση (paravirtualization), αλλά χρησιμοποιεί μια διεπαφή προγράμματος εφαρμογής (API) που μπορεί να τροποποιήσει τα Guest OSes. Για να τροποποιηθεί το λειτουργικό σύστημα επισκέπτη, ο πηγαίος κώδικας για το λειτουργικό σύστημα επισκέπτη πρέπει να είναι προσβάσιμος ώστε να είναι δυνατή η αντικατάσταση τμημάτων κώδικα. Στη συνέχεια, το λειτουργικό σύστημα ανακατασκευάζεται για να χρησιμοποιήσει τις νέες τροποποιήσεις. Στη συνέχεια, ο hypervisor θα παρέχει εντολές που αποστέλλονται από το λειτουργικό σύστημα στον hypervisor, που ονομάζονται hypercalls και χρησιμοποιούνται για λειτουργίες πυρήνα, όπως η διαχείριση της μνήμης. Το Paravirtualization απαιτεί την τροποποίηση του λειτουργικού συστήματος, το οποίο δημιουργεί επίσης μια μεγάλη εξάρτηση μεταξύ του λειτουργικού συστήματος και του επόπτη που θα μπορούσε ενδεχομένως να περιορίσει περαιτέρω ενημερώσεις.
- Hardware-assisted virtualization: Η εικονικοποίηση με υποβοήθηση υλικού χρησιμοποιεί το υλικό ενός υπολογιστή ως αρχιτεκτονική υποστήριξη για τη δημιουργία και διαχείριση ενός πλήρως εικονικοποιημένου VM. Η εικονικοποίηση με υποβοήθηση υλικού εισήχθη για πρώτη φορά από την IBM το 1972 με το IBM System / 370. Η δημιουργία ενός VMM σε λογισμικό επέβαλε σημαντική επιβάρυνση στο σύστημα κεντρικού υπολογιστή. Οι σχεδιαστές συνειδητοποίησαν σύντομα ότι οι λειτουργίες εικονικοποίησης θα μπορούσαν να εφαρμοστούν πολύ

πιο αποτελεσματικά σε υλικό παρά σε λογισμικό, οδηγώντας στην ανάπτυξη εκτεταμένων συνόλων εντολών για επεξεργαστές Intel και AMD, όπως επεκτάσεις Intel VT και AMD-V. Έτσι, ο hypervisor μπορεί απλά να κάνει κλήσεις στον επεξεργαστή, ο οποίος στη συνέχεια κάνει τη «βαριά ανύψωση» της δημιουργίας και της συντήρησης των VM. Η επιβάρυνση του συστήματος μειώνεται σημαντικά, επιτρέποντας στο σύστημα κεντρικού υπολογιστή να φιλοξενεί περισσότερα VM και να παρέχει μεγαλύτερη απόδοση VM για πιο απαιτητικούς φόρτους εργασίας. Η εικονικοποίηση με υποβοήθηση υλικού είναι η πιο κοινή μορφή εικονικοποίησης.

2.4 Εικονικοποίηση αποθήκευσης

Η εικονικοποίηση αποθήκευσης (επίσης μερικές φορές ονομάζεται αποθήκευση που καθορίζεται από λογισμικό ή ένα εικονικό SAN) είναι η συγκέντρωση πολλαπλών συστοιχιών φυσικής αποθήκευσης από SAN και καθιστώντας τα να εμφανίζονται ως μία ενιαία εικονική συσκευή αποθήκευσης. Η εικονικοποίηση του χώρου αποθήκευσης διαχωρίζει το λογισμικό διαχείρισης αποθήκευσης από την υποκείμενη υποδομή υλικού, προκειμένου να παρέχει μεγαλύτερη ευελιξία και κλιμακωτές ομάδες πόρων αποθήκευσης. Η εικονικοποίηση χώρου αποθήκευσης εκτελείται σε πολλές συσκευές αποθήκευσης, κάνοντάς τις να φαίνονται σαν να ήταν μία ομάδα αποθήκευσης. Οι συγκεντρωτικές συσκευές αποθήκευσης μπορούν να προέρχονται από διαφορετικά δίκτυα. Η μηχανή εικονικοποίησης αποθήκευσης (storage virtualization engine) προσδιορίζει τη διαθέσιμη χωρητικότητα αποθήκευσης από πολλές συστοιχίες και μέσα αποθήκευσης, την συγκεντρώνει, τη διαχειρίζεται και την παρουσιάζει σε εφαρμογές. Το λογισμικό εικονικοποίησης λειτουργεί παρεμποδίζοντας τις αιτήσεις εισόδου / εξόδου (I/O) συστήματος αποθήκευσης σε διακομιστές. Αντί η CPU να επεξεργάζεται το αίτημα και να επιστρέφει δεδομένα στον χώρο αποθήκευσης, η μηχανή χαρτογραφεί φυσικά αιτήματα στο εικονικό χώρο αποθήκευσης και αποκτά πρόσβαση στα ζητούμενα δεδομένα από τη φυσική της θέση. Μόλις ολοκληρωθεί η διαδικασία του υπολογιστή, η μηχανή εικονικοποίησης στέλνει το I / O από τη CPU στη φυσική της διεύθυνση και ενημερώνει την εικονική χαρτογράφηση.

Η ανατομία της εικονικοποίησης αποθήκευσης



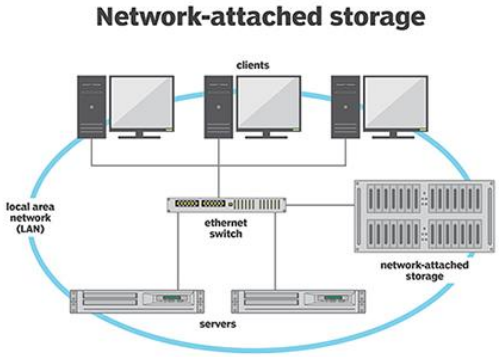
Εικόνα 10: Storage Virtualization Anatomy

Data Level:

- Η Block-Based εικονικοποίηση αποθήκευσης είναι ο πιο κοινός τύπος εικονικοποίησης αποθήκευσης. Αφαιρεί τη λογική αποθήκευση του

συστήματος αποθήκευσης από τα φυσικά του στοιχεία. Τα φυσικά συστατικά περιλαμβάνουν μπλοκ μνήμης και μέσα αποθήκευσης, ενώ τα λογικά στοιχεία περιλαμβάνουν χωρίσματα μονάδας δίσκου (disc partitions). Η μηχανή εικονικοποίησης αποθήκευσης ανακαλύπτει όλα τα διαθέσιμα μπλοκ ανεξάρτητα από τη φυσική τοποθεσία του συστήματος αποθήκευσης, τα λογικά διαμερίσματα ή τον κατασκευαστή. Η μηχανή αφήνει τα δεδομένα στη φυσική τους θέση και χαρτογραφεί τη διεύθυνση στην εικονική «ομάδα» αποθήκευσης (Virtual storage pool).

- Η εικονικοποίηση σε επίπεδο αρχείου (File level) λειτουργεί μέσω συσκευών NAS για τη συγκέντρωση και διαχείριση ξεχωριστών συσκευών NAS. Ο συνδεδεμένος χώρος αποθήκευσης δικτύου (NAS) είναι ένας διακομιστής αποθήκευσης δεδομένων υπολογιστή σε επίπεδο αρχείου (σε αντίθεση με το επίπεδο μπλοκ) που είναι συνδεδεμένος σε ένα δίκτυο υπολογιστών και παρέχει πρόσβαση δεδομένων σε μια ετερογενή ομάδα πελατών. Τα συστήματα NAS είναι δικτυωμένες συσκευές που περιέχουν μία ή περισσότερες μονάδες αποθήκευσης, συχνά τοποθετημένες σε containers αποθήκευσης ή RAID. Αποθηκευτικός χώρος συνδεδεμένος στο δίκτυο καταργεί την ευθύνη που περιέχει το file serving από άλλους διακομιστές στο δίκτυο. Συνήθως παρέχουν πρόσβαση σε αρχεία χρησιμοποιώντας πρωτόκολλα κοινής χρήσης αρχείων δικτύου, όπως NFS, SMB ή AFP. Τα πιθανά οφέλη του αποκλειστικού χώρου αποθήκευσης που συνδέεται με το δίκτυο, σε σύγκριση με τους διακομιστές γενικής χρήσης που εξυπηρετούν επίσης αρχεία, περιλαμβάνουν ταχύτερη πρόσβαση δεδομένων, ευκολότερη διαχείριση και απλή διαμόρφωση.



Εικόνα 11: Network-attached storage (NAS)

Ενώ η διαχείριση ενός ενιαίου NAS δεν είναι ιδιαίτερα δύσκολη, η διαχείριση πολλαπλών συσκευών είναι χρονοβόρα και δαπανηρή. Οι συσκευές NAS είναι φυσικά και λογικά ανεξάρτητες μεταξύ τους, κάτι που απαιτεί ατομική διαχείριση, βελτιστοποίηση (optimization) και provisioning. Αυτό αυξάνει την πολυπλοκότητα και απαιτεί από τους χρήστες να γνωρίζουν το φυσικό όνομα διαδρομής για πρόσβαση σε ένα αρχείο.

Μία από τις πιο χρονοβόρες λειτουργίες με πολλές συσκευές NAS είναι η μετεγκατάσταση δεδομένων μεταξύ τους. Καθώς οι οργανισμοί ξεπερνούν τις παλαιότερες συσκευές NAS, αγοράζουν συχνά μια νέα και μεγαλύτερη. Αυτό απαιτεί συχνά μετεγκατάσταση δεδομένων από παλαιότερες συσκευές που πλησιάζουν τα όρια χωρητικότητας. Αυτό με τη σειρά του απαιτεί σημαντικό χρόνο

διακοπής λειτουργίας για τη διαμόρφωση της νέας συσκευής, τη μετεγκατάσταση δεδομένων από τη συσκευή παλαιού τύπου και τη δοκιμή των δεδομένων μετεγκατάστασης πριν από τη ζωντανή προβολή. Ωστόσο, ο χρόνος διακοπής επηρεάζει τους χρήστες και τα έργα και ο εκτεταμένος χρόνος διακοπής λειτουργίας για τη μετεγκατάσταση δεδομένων μπορεί να επηρεάσει οικονομικά τον οργανισμό. Η εικονικοποίηση δεδομένων σε επίπεδο αρχείου αποκρύπτει την πολυπλοκότητα της διαχείρισης πολλών συσκευών NAS και επιτρέπει στους διαχειριστές να συγκεντρώνουν πόρους αποθήκευσης αντί να τους περιορίζουν σε συγκεκριμένες εφαρμογές ή ομάδες εργασίας. Η εικονικοποίηση των συσκευών NAS καθιστά επίσης περιττή τη διακοπή κατά τη μετεγκατάσταση δεδομένων. Η μηχανή εικονικοποίησης διατηρεί τις σωστές φυσικές διευθύνσεις και εκ νέου χάρτες αλλαγών διευθύνσεων στην εικονική ομάδα. Ένας χρήστης μπορεί να αποκτήσει πρόσβαση σε ένα αρχείο από την παλιά συσκευή και να το αποθηκεύσει χωρίς να ξέρει ποτέ ότι έγινε μετεγκατάσταση.

Intelligence: Host, Network, Array

Η μηχανή εικονικοποίησης μπορεί να βρίσκεται σε διαφορετικά στοιχεία υπολογιστών. Τα τρία πιο συνηθισμένα είναι κεντρικός υπολογιστής, δίκτυο και πίνακας. Καθένα εξυπηρετεί μια διαφορετική περίπτωση χρήσης εικονικοποίησης αποθήκευσης.

- Host-Based: Κύρια περίπτωση χρήσης για εικονικοποίηση χώρου αποθήκευσης για περιβάλλοντα VM και διαδικτυακές εφαρμογές. Ορισμένοι διακομιστές παρέχουν εικονικοποίηση από το επίπεδο λειτουργικού συστήματος. Το λειτουργικό σύστημα εικονικοποιεί τον διαθέσιμο χώρο αποθήκευσης για βελτιστοποίηση χωρητικότητας και αυτοματοποίηση κλιμακωτών προγραμμάτων αποθήκευσης. Οι πιο συνηθισμένοι διακομιστές εικονικοποίησης αποθήκευσης που βασίζονται σε κεντρικούς υπολογιστές συγκεντρώνουν χώρο αποθήκευσης σε εικονικά περιβάλλοντα και παρουσιάζουν το σύνολο στο λειτουργικό σύστημα επισκεπτών (Guest OS). Μία κοινή εφαρμογή είναι ένα δυναμικά επεκτάσιμο VM που λειτουργεί ως χώρος αποθήκευσης. Δεδομένου ότι οι VM αναμένουν να δουν σκληρούς δίσκους, η μηχανή εικονικοποίησης παρουσιάζει υποκείμενη αποθήκευση στο VM ως σκληρό δίσκο. Στην πραγματικότητα, αυτό που είναι ο «σκληρός δίσκος» είναι η λογική ομάδα αποθήκευσης που δημιουργήθηκε από στοιχεία αποθήκευσης που βασίζονται σε δίσκους και πίνακες. Αυτή η προσέγγιση εικονικοποίησης είναι πιο συνηθισμένη στο χώρο αποθήκευσης cloud και υπερ-σύγκλισης (hyper-converged). Ένας ενιαίος κεντρικός υπολογιστής ή ένα υπερσυγκεντρωμένο σύστημα συγκεντρώνει διαθέσιμο χώρο αποθήκευσης σε εικονικές μονάδες δίσκου και παρουσιάζει τις μονάδες σε μηχανήματα επισκεπτών.
- Network-Based: Η εικονικοποίηση αποθήκευσης βάσει δικτύου είναι ο πιο κοινός τύπος για τους κατόχους SAN, οι οποίοι το χρησιμοποιούν για να επεκτείνουν την επένδυσή τους προσθέτοντας περισσότερο χώρο αποθήκευσης. Το SAN είναι ένα δίκτυο συσκευών αποθήκευσης υψηλής ταχύτητας που συνδέει επίσης αυτές τις συσκευές

αποθήκευσης με διακομιστές. Παρέχει χώρο αποθήκευσης σε επίπεδο μπλοκ που μπορεί να προσεγγιστεί από τις εφαρμογές που εκτελούνται σε οποιονδήποτε διακομιστή δικτύου. Η συσκευή που βασίζεται στο δίκτυο (Network-Based Device) αφαιρεί το I / O αποθήκευσης που εκτελείται σε ολόκληρο το δίκτυο αποθήκευσης και μπορεί να αναπαραγάγει δεδομένα σε όλες τις συνδεδεμένες συσκευές αποθήκευσης.

- Array-Based: Ορισμένα επίπεδα RAID εικονικοποιούνται καθώς αφαιρούν την αποθήκευση από πολλούς φυσικούς δίσκους σε έναν λογικό πίνακα. Η εικονικοποίηση βάσει πίνακα συνήθως αναφέρεται σε έναν εξειδικευμένο ελεγκτή αποθήκευσης που παρακολουθεί αιτήματα εισόδου / εξόδου από δευτερεύοντες ελεγκτές αποθήκευσης και προσαρμόζει αυτόματα δεδομένα σε συνδεδεμένα συστήματα αποθήκευσης. Το εργαλείο δίνει τη δυνατότητα στους διαχειριστές να εκχωρήσουν πολυμέσα σε διαφορετικά επίπεδα αποθήκευσης, συνήθως SSD σε επίπεδα υψηλής απόδοσης και HDD σε κοντινά ή δευτερεύοντα επίπεδα. Αυτή η προσέγγιση εικονικοποίησης είναι πιο περιορισμένη από την εικονικοποίηση που βασίζεται σε κεντρικό υπολογιστή ή σε δίκτυο, καθώς η εικονικοποίηση πραγματοποιείται μόνο μέσω συνδεδεμένων ελεγκτών. Οι δευτερεύοντες ελεγκτές χρειάζονται το ίδιο εύρος ζώνης με τους ελεγκτές αποθήκευσης εικονικοποίησης, οι οποίοι μπορούν να επηρεάσουν την απόδοση. Ωστόσο, εάν μια επιχείρηση έχει επενδύσει σε μεγάλο βαθμό σε έναν προηγμένο υβριδικό πίνακα, η ευφυΐα αποθήκευσης του πίνακα μπορεί να ξεπεράσει αυτό που μπορεί να προσφέρει η εικονικοποίηση αποθήκευσης. Σε αυτήν την περίπτωση, η εικονικοποίηση βάσει πίνακα επιτρέπει στην επιχείρηση να διατηρήσει τις εγγενείς δυνατότητες του πίνακα και να προσθέσει εικονική βαθμίδα για καλύτερη απόδοση.

Band: In-Band, Out-of-Band

- Η εικονικοποίηση αποθήκευσης εντός ζώνης συμβαίνει όταν η μηχανή εικονικοποίησης λειτουργεί μεταξύ του κεντρικού υπολογιστή και του χώρου αποθήκευσης. Τόσο οι αιτήσεις εισόδου / εξόδου όσο και τα δεδομένα περνούν από το επίπεδο εικονικοποίησης, το οποίο επιτρέπει στην μηχανή να παρέχει προηγμένες λειτουργίες όπως αποθήκευση δεδομένων, αντιγραφή και μετεγκατάσταση δεδομένων. Το In-Band καταλαμβάνει λιγότερους πόρους διακομιστή κεντρικού υπολογιστή, επειδή δεν χρειάζεται να βρει και να επισυνάψει πολλές συσκευές αποθήκευσης. Ο διακομιστής βλέπει μόνο τον συγκεντρωτικό χώρο αποθήκευσης στη διαδρομή δεδομένων του. Ωστόσο, όσο μεγαλύτερη η συγκέντρωση, τόσο μεγαλύτερος είναι ο κίνδυνος να επηρεάσει την απόδοση της διαδρομής δεδομένων.
- Η εικονικοποίηση αποθήκευσης εκτός ζώνης χωρίζει τη διαδρομή σε διαδρομές ελέγχου και δεδομένων. Μόνο η διαδρομή ελέγχου διασχίζει το εργαλείο εικονικοποίησης, το οποίο παρακολουθεί αιτήματα εισόδου / εξόδου από τον κεντρικό υπολογιστή, αναζητά και χαρτογραφεί δεδομένα σε τοποθεσίες φυσικής μνήμης και

εκδίδει ένα ενημερωμένο αίτημα εισόδου / εξόδου στον χώρο αποθήκευσης. Τα δεδομένα δεν περνούν μέσω της συσκευής, γεγονός που καθιστά αδύνατη την προσωρινή αποθήκευση. Η εικονικοποίηση εκτός ζώνης εγκαθιστά «πράκτορες» (agents) σε μεμονωμένους διακομιστές για να κατευθύνει το I / O αποθήκευσης στο εργαλείο εικονικοποίησης. Η εικονικοποίηση εκτός ζώνης δεν εμποδίζει δεδομένα (bottleneck data) όπως μπορεί να γίνεται στην εντός ζώνης. Ωστόσο, οι βέλτιστες πρακτικές είναι να αποφεύγεται τη διακοπή της εικονικοποίησης προσθέτοντας επιπλέον περιττές συσκευές εκτός ζώνης.

2.5 Εικονικοποίηση δικτύου

Η εικονικοποίηση δικτύου είναι μια μέθοδος συνδυασμού των διαθέσιμων πόρων σε ένα δίκτυο για την ενοποίηση πολλών φυσικών δικτύων, τη διαίρεση ενός δικτύου σε τμήματα ή τη δημιουργία δικτύων λογισμικού μεταξύ εικονικών μηχανών (VMs). Όσοι χρησιμοποιούν την εικονικοποίηση δικτύου μπορούν να διαχειριστούν το περιβάλλον τους ως ένα μόνο δίκτυο που βασίζεται σε λογισμικό. Η εικονικοποίηση δικτύου αποσκοπεί στη βελτιστοποίηση της ταχύτητας, της αξιοπιστίας, της ευελιξίας, της επεκτασιμότητας και της ασφάλειας του δικτύου. Είναι ιδιαίτερα χρήσιμο σε δίκτυα που αντιμετωπίζουν ξαφνικές, μεγάλες και απρόβλεπτες αυξήσεις στη χρήση. Η εικονικοποίηση δικτύου λειτουργεί συνδυάζοντας τους διαθέσιμους πόρους σε ένα δίκτυο και χωρίζοντας το διαθέσιμο εύρος ζώνης σε κανάλια, καθένα από τα οποία είναι ανεξάρτητο από τα άλλα και καθένα από τα οποία μπορεί να εκχωρηθεί (ή να εκχωρηθεί εκ νέου) σε έναν συγκεκριμένο διακομιστή ή συσκευή σε πραγματικό χρόνο. Κάθε κανάλι είναι ανεξάρτητα ασφαλές. Κάθε συνδρομητής έχει κοινή πρόσβαση σε όλους τους πόρους του δικτύου από έναν μόνο υπολογιστή. Η εικονικοποίηση δικτύου αποσκοπεί στη βελτίωση της παραγωγικότητας, της αποδοτικότητας και της ικανοποίησης από την εργασία του διαχειριστή εκτελώντας πολλές από αυτές τις εργασίες αυτόματα, αποκρύπτοντας έτσι την πραγματική πολυπλοκότητα του δικτύου. Είναι δυνατή η διαχείριση κεντρικών αρχείων, εικόνων, προγραμμάτων και φακέλων από έναν μόνο ιστότοπο. Τα μέσα αποθήκευσης, όπως σκληροί δίσκοι και μονάδες ταινιών, μπορούν εύκολα να προστεθούν ή να εκχωρηθούν εκ νέου. Ο αποθηκευτικός χώρος μπορεί να μοιραστεί ή να ανακατανεμηθεί μεταξύ των διακομιστών.

Υπάρχουν δύο μορφές εικονικοποίησης δικτύου, η εσωτερική εικονικοποίηση (internal) και η εξωτερική εικονικοποίηση (external). Η εσωτερική εικονικοποίηση έχει σχεδιαστεί για τη χρήση κοντέινερ λογισμικού για την αναπαραγωγή της λειτουργικότητας ενός μόνο δικτύου. Το εσωτερικό λογισμικό επιτρέπει στους VM να ανταλλάσσουν δεδομένα σε έναν κεντρικό υπολογιστή χωρίς να χρησιμοποιούν εξωτερικό δίκτυο. Η εξωτερική εικονικοποίηση θα χρησιμοποιήσει εργαλεία όπως διακόπτες, προσαρμογείς ή ένα δίκτυο για να συνδυάσει ένα ή περισσότερα δίκτυα σε εικονικές μονάδες. Συνδυάζει πολλά τοπικά δίκτυα σε ένα μόνο «εικονικό» δίκτυο για τη βελτίωση της αποτελεσματικότητας του δικτύου.

Η εικονικοποίηση δικτύου μπορεί να μειώσει το κόστος αγοράς και συντήρησης υλικού, το οποίο είναι ιδιαίτερα χρήσιμο για οργανισμούς με έντονο φόρτο εργασίας που θα απαιτούσαν υπερβολική τροφοδοσία για να συμβαδίσουν με τη ζήτηση. Επίσης, καθώς αυξάνεται ο όγκος δεδομένων και η ταχύτητα, η ικανότητα κλιμάκωσης επιτρέπει στις ομάδες ασφαλείας να διατηρούν καλύτερη προβολή του δικτύου.

Το λογισμικό εικονικοποίησης δικτύου επιτρέπει στους οργανισμούς να ελέγχουν το είδος της κίνησης που διέρχεται από το φυσικό δίκτυο. Πολλοί επιτιθέμενοι στηρίζονται στο γεγονός ότι μόλις παραβιάσουν την περίμετρο ασφαλείας, υπάρχουν λίγοι έλεγχοι ασφαλείας. Η εικονικοποίηση δικτύου επιτρέπει στους οργανισμούς να καταπολεμούν καλύτερα τις απειλές ασφαλείας δημιουργώντας μικρο-περιμέτρους στο δίκτυο. Με αυτήν την ικανότητα, γνωστή ως μικρο-τμηματοποίηση, μπορούν να διατηρήσουν ευαίσθητα δεδομένα σε ένα συγκεκριμένο εικονικό δίκτυο στο οποίο έχουν πρόσβαση μόνο εξουσιοδοτημένοι χρήστες. Για παράδειγμα, ένας οργανισμός θα μπορούσε να εξασφαλίσει δεδομένα VoIP τοποθετώντας τα μέσα στο δικό του εικονικό δίκτυο με περιορισμένη πρόσβαση χρήστη. Επιπλέον, το λογισμικό εικονικοποίησης δικτύου μπορεί να μειώσει ή ακόμη και να εξαλείψει τις διακοπές λειτουργίας που δημιουργούνται από αστοχίες υλικού και να βελτιώσει τους χρόνους αποκατάστασης καταστροφών. Η αποκατάσταση καταστροφών με παραδοσιακό υλικό δικτύου απαιτεί πολλά χειροκίνητα, χρονοβόρα βήματα, όπως αλλαγή της διεύθυνσης IP του συστήματος και ενημέρωση του τείχους προστασίας. Η εικονικοποίηση δικτύου καταργεί αυτά τα βήματα.

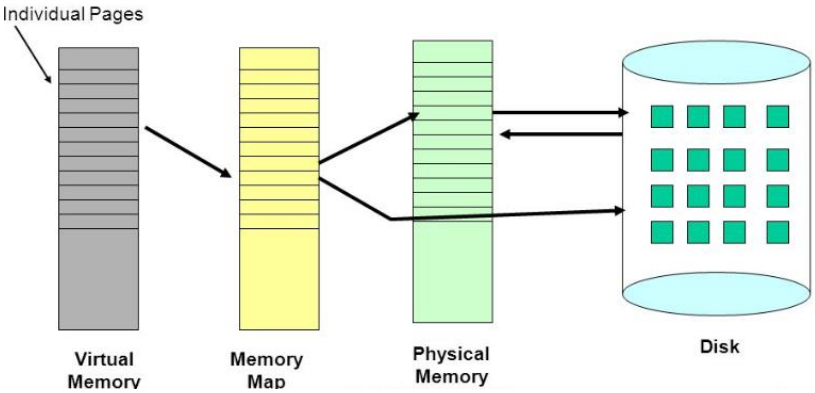
Χωρίς εικονικοποίηση δικτύου, η παροχή δικτύου είναι χρονοβόρα, χειροκίνητη διαδικασία. Ως αποτέλεσμα, κάθε φορά που μια εφαρμογή απαιτεί θεμελιώδεις αλλαγές δικτύου, ο χρόνος ανάπτυξης της εφαρμογής παρατείνεται. Επιπλέον, ο κίνδυνος αποτυχίας ανάπτυξης (deployment failure) αυξάνεται σημαντικά όταν οι οργανισμοί εκτελούν χειροκίνητες αναπτύξεις. Δεδομένου ότι η εικονικοποίηση δικτύου αυτοματοποιεί τη διαμόρφωση δικτύου, μπορούν αντ' αυτού να μειώσουν το χρόνο ανάπτυξης εφαρμογών από εβδομάδες σε λεπτά. Η μείωση του χρόνου ανάπτυξης μπορεί να έχει σημαντικό αντίκτυπο στην κατώτατη γραμμή μιας εταιρείας, επιτρέποντας ταχύτερη διάθεση νέων προϊόντων ή σημαντικές ενημερώσεις εφαρμογών.

2.6 Εικονικοποίηση Μνήμης και Εικονική Μνήμη

Η εικονική μνήμη είναι μια ικανότητα διαχείρισης μνήμης ενός λειτουργικού συστήματος (OS) - το οποίο χρησιμοποιεί υλικό και λογισμικό για να επιτρέψει σε έναν υπολογιστή να αντισταθμίσει τις ελλείψεις φυσικής μνήμης, μεταφέροντας προσωρινά δεδομένα από τη μνήμη τυχαίας πρόσβασης (RAM) στον δίσκο αποθήκευση. Η εικονική μνήμη αναπτύχθηκε λόγο κόστους της φυσικής μνήμης (η εγκατεστημένη μνήμη RAM). Οι υπολογιστές έχουν περιορισμένη ποσότητα μνήμης RAM, οπότε η μνήμη μπορεί να εξαντληθεί, ειδικά όταν εκτελούνται πολλά προγράμματα ταυτόχρονα. Ένα σύστημα που χρησιμοποιεί εικονική μνήμη χρησιμοποιεί ένα τμήμα του σκληρού δίσκου για να μιμηθεί τη μνήμη RAM. Με την εικονική μνήμη, ένα σύστημα μπορεί να φορτώσει μεγαλύτερα προγράμματα ή πολλαπλά

προγράμματα που εκτελούνται ταυτόχρονα, επιτρέποντας σε καθένα να λειτουργεί σαν να έχει απεριόριστη μνήμη και χωρίς να χρειάζεται να αγοράσει περισσότερη μνήμη RAM.

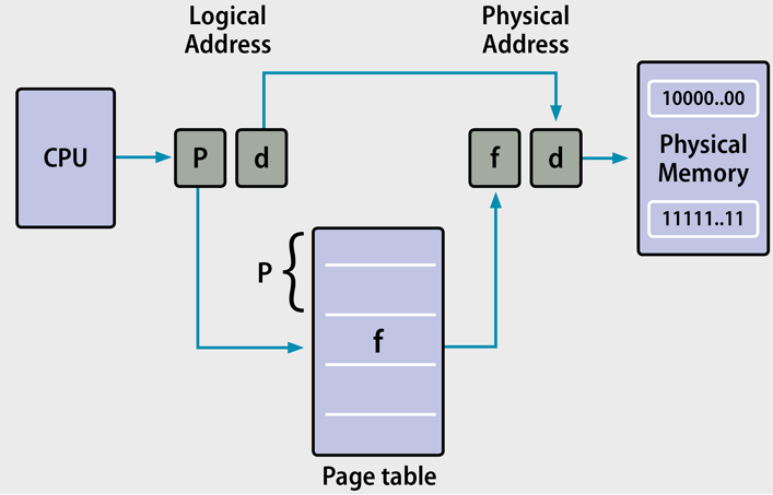
Κατά την αντιγραφή της εικονικής μνήμης στη φυσική μνήμη, το λειτουργικό σύστημα διαιρεί τη μνήμη σε αρχεία σελίδων ή ανταλλάσσει αρχεία με σταθερό αριθμό διευθύνσεων. Κάθε σελίδα αποθηκεύεται σε δίσκο και όταν χρειάζεται η σελίδα, το λειτουργικό σύστημα το αντιγράφει από το δίσκο στην κύρια μνήμη και μεταφράζει τις εικονικές διευθύνσεις σε πραγματικές διευθύνσεις.



Εικόνα 12. αντιγραφή της εικονικής μνήμης στη φυσική μνήμη

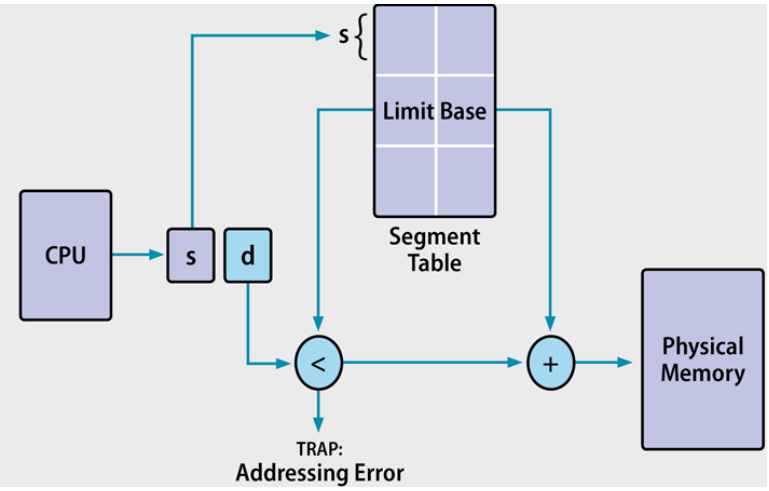
Η μονάδα διαχείρισης μνήμης ενός υπολογιστή (MMU) χειρίζεται λειτουργίες μνήμης, συμπεριλαμβανομένης της διαχείρισης εικονικής μνήμης. Στους περισσότερους υπολογιστές, το υλικό MMU είναι ενσωματωμένο στη CPU. Υπάρχουν δύο τρόποι χειρισμού της εικονικής μνήμης: σελιδοποίηση (paged) και τμηματοποίηση (segmented).

Η σελιδοποίηση διαιρεί τη μνήμη σε ενότητες ή αρχεία σελιδοποίησης, συνήθως σε μέγεθος περίπου 4 KB. Όταν ένας υπολογιστής εξαντλήσει τη μνήμη RAM του, οι σελίδες που δεν χρησιμοποιούνται μεταφέρονται στο τμήμα του σκληρού δίσκου που έχει οριστεί για εικονική μνήμη χρησιμοποιώντας ένα αρχείο ανταλλαγής. Ένα αρχείο ανταλλαγής είναι ένας χώρος που διατίθεται στον σκληρό δίσκο ως επεκτάσεις εικονικής μνήμης της μνήμης RAM του υπολογιστή. Όταν απαιτείται το αρχείο ανταλλαγής, αποστέλλεται πίσω στη μνήμη RAM χρησιμοποιώντας μια διαδικασία που ονομάζεται ανταλλαγή σελίδων. Αυτό το σύστημα διασφαλίζει ότι το λειτουργικό σύστημα και οι εφαρμογές του υπολογιστή δεν εξαντλούνται από την πραγματική μνήμη. Η διαδικασία σελιδοποίησης περιλαμβάνει τη χρήση πινάκων σελίδων, οι οποίοι μεταφράζουν τις εικονικές διευθύνσεις που χρησιμοποιούν το λειτουργικό σύστημα και οι εφαρμογές στις φυσικές διευθύνσεις που χρησιμοποιεί η MMU. Οι καταχωρήσεις στον πίνακα σελίδων υποδεικνύουν εάν η σελίδα βρίσκεται σε πραγματική μνήμη. Εάν το λειτουργικό σύστημα ή ένα πρόγραμμα δεν βρει αυτό που χρειάζεται στη μνήμη RAM, τότε το MMU ανταποκρίνεται στην αναφορά μνήμης που λείπει (missing memory reference) με ένα σφάλμα εξαίρεσης σελίδας (page exception fault) το οποίο λαμβάνει το OS ώστε να μετακινήσει τη σελίδα πίσω στη μνήμη. Όταν η σελίδα βρίσκεται σε μνήμη RAM, η εικονική της διεύθυνση εμφανίζεται στον πίνακα σελίδων.



Εικόνα 13.

Η τμηματοποίηση χρησιμοποιείται επίσης για τη διαχείριση της εικονικής μνήμης. Αυτή η προσέγγιση χωρίζει την εικονική μνήμη σε τμήματα διαφορετικών μηκών. Τα τμήματα που δεν χρησιμοποιούνται στη μνήμη μπορούν να μετακινηθούν σε χώρο εικονικής μνήμης στον σκληρό δίσκο. Οι τμηματοποιημένες πληροφορίες ή οι διαδικασίες παρακολουθούνται σε έναν πίνακα τμημάτων, ο οποίος δείχνει εάν ένα τμήμα υπάρχει στη μνήμη, αν έχει τροποποιηθεί και ποια είναι η φυσική του διεύθυνση.



Εικόνα 14.

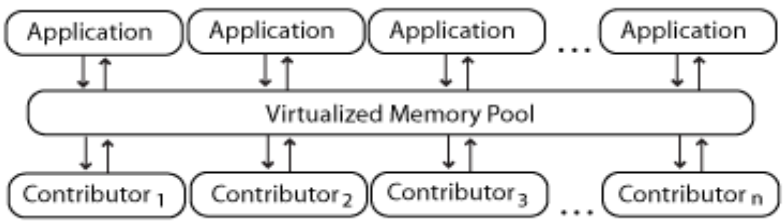
Ορισμένα εικονικά συστήματα μνήμης συνδυάζουν τμηματοποίηση και σελιδοποίηση. Σε αυτήν την περίπτωση, η μνήμη χωρίζεται σε πλαίσια ή σελίδες. Τα τμήματα καταλαμβάνουν πολλές σελίδες και η εικονική διεύθυνση περιλαμβάνει τόσο τον αριθμό τμήματος όσο και τον αριθμό σελίδας. Μεταξύ των βασικών πλεονεκτημάτων της εικονικής μνήμης είναι η ικανότητά της να χειρίζεται διπλάσιες διευθύνσεις από την κύρια μνήμη. Χρησιμοποιεί λογισμικό για να καταναλώνει περισσότερη μνήμη χρησιμοποιώντας τον σκληρό δίσκο ως προσωρινή αποθήκευση, ενώ οι MMU μεταφράζουν διευθύνσεις εικονικής μνήμης σε φυσικές διευθύνσεις μέσω της CPU. Τα προγράμματα χρησιμοποιούν εικονικές διευθύνσεις για την αποθήκευση οδηγιών και δεδομένων. Όταν εκτελείται ένα πρόγραμμα, οι εικονικές διευθύνσεις μετατρέπονται σε πραγματικές διευθύνσεις μνήμης.

2.7 Εικονικοποίηση μνήμης

Στην επιστήμη των υπολογιστών, η εικονικοποίηση μνήμης αποσυνδέει πόρους μνήμης τυχαίας προσπέλασης (RAM) από μεμονωμένα συστήματα στο κέντρο δεδομένων και, στη συνέχεια, συγκεντρώνει αυτούς τους πόρους σε μια εικονική δεξαμενή μνήμης που είναι διαθέσιμη σε οποιονδήποτε υπολογιστή στο σύμπλεγμα. Η δεξαμενή μνήμης είναι προσβάσιμη από το λειτουργικό σύστημα ή εφαρμογές που εκτελούνται πάνω από το λειτουργικό σύστημα. Το κατανεμημένο σύνολο μνήμης μπορεί στη συνέχεια να χρησιμοποιηθεί ως προσωρινή μνήμη υψηλής ταχύτητας, ως επίπεδο μηνυμάτων ή μεγάλος, κοινόχρηστος πόρος μνήμης για μια CPU ή μια εφαρμογή GPU. Η εικονικοποίηση μνήμης επιτρέπει σε διακομιστές δικτύου να μοιράζονται ένα απόθεμα μνήμης για να ξεπεράσουν τους φυσικούς περιορισμούς μνήμης, ένα συνηθισμένο εμπόδιο στην απόδοση του λογισμικού. Με αυτήν την ικανότητα ενσωματωμένη στο δίκτυο, οι εφαρμογές μπορούν να επωφεληθούν από μια πολύ μεγάλη ποσότητα μνήμης για βελτίωση συνολική απόδοση, χρήση συστήματος, αύξηση της απόδοσης χρήσης μνήμης και ενεργοποίηση νέων περιπτώσεων χρήσης. Το λογισμικό στους κόμβους δεξαμενών μνήμης (servers) επιτρέπει στους κόμβους να συνδέονται στο χώρο μνήμης για να συνεισφέρουν στη μνήμη και να αποθηκεύουν και να ανακτούν δεδομένα. Το λογισμικό διαχείρισης και οι τεχνολογίες της υπερπροσφοράς μνήμης διαχειρίζονται την κοινόχρηστη μνήμη, την εισαγωγή δεδομένων, τις πολιτικές απομάκρυνσης και παροχής, την ανάθεση δεδομένων σε συνεισφέροντες κόμβους και χειρίζεται αιτήματα από κόμβους πελατών. Η πρόσβαση στη μνήμη μπορεί να προσεγγιστεί σε επίπεδο εφαρμογής ή σε επίπεδο λειτουργικού συστήματος. Σε επίπεδο εφαρμογής, η πρόσβαση στην συγκεντρωτική μνήμη (memory pool) γίνεται μέσω API ή ως δικτυακού συστήματος αρχείων για τη δημιουργία προσωρινής κοινόχρηστης μνήμης υψηλής ταχύτητας. Σε επίπεδο λειτουργικού συστήματος, μια προσωρινή μνήμη σελίδας μπορεί να χρησιμοποιήσει το χώρο συγκέντρωσης ως έναν πολύ μεγάλο πόρο μνήμης που είναι πολύ πιο γρήγορος από τον τοπικό ή δικτυακό χώρο αποθήκευσης.

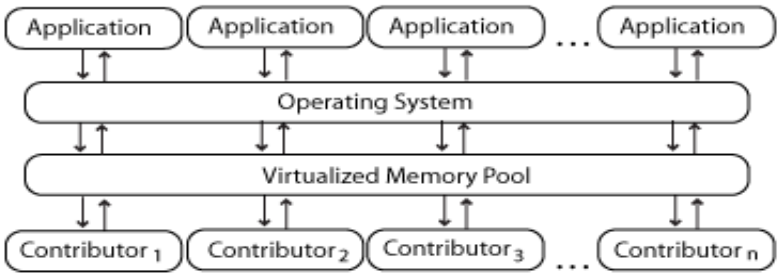
Οι υλοποιήσεις εικονικοποίησης μνήμης διακρίνονται από τα κοινόχρηστα συστήματα μνήμης. Τα συστήματα κοινόχρηστης μνήμης δεν επιτρέπουν την αφαίρεση πόρων μνήμης, απαιτώντας έτσι υλοποίηση με μία μόνο παρουσία λειτουργικού συστήματος.

Ενσωμάτωση επιπέδου εφαρμογής: Σε αυτήν την περίπτωση, οι εφαρμογές που εκτελούνται σε συνδεδεμένους υπολογιστές συνδέονται απευθείας στην ομάδα μνήμης μέσω ενός API ή του συστήματος αρχείων.



Εικόνα 15: Ενσωμάτωση επιπέδου εφαρμογής.

Ενσωμάτωση επιπέδου λειτουργικού συστήματος: Σε αυτήν την περίπτωση, το λειτουργικό σύστημα συνδέεται στο χώρο αποθήκευσης μνήμης και καθιστά τη συγκεντρωτική μνήμη διαθέσιμη σε εφαρμογές.



Εικόνα 16: Ενσωμάτωση επιπέδου λειτουργικού συστήματος

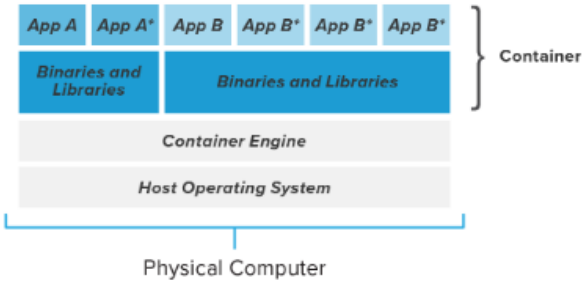
Η τεχνολογία εικονικοποίησης μνήμης ακολουθεί μετά από αρχιτεκτονικές διαχείρισης μνήμης και τεχνικές εικονικής μνήμης.

3. Εικονικοποίηση σε Επίπεδο Λειτουργικού Συστήματος

3.1 Η έννοια του Containerization και του Container

Το Containerization επιτρέπει στους προγραμματιστές να δημιουργούν και να αναπτύσσουν εφαρμογές γρηγορότερα και με μεγαλύτερη ασφάλεια. Με τις παραδοσιακές μεθόδους, ο κώδικας αναπτύσσεται σε ένα συγκεκριμένο υπολογιστικό περιβάλλον το οποίο, όταν μεταφέρεται σε μια νέα τοποθεσία, συχνά οδηγεί σε σφάλματα (bugs, errors). Για παράδειγμα, όταν ένας προγραμματιστής μεταφέρει κώδικα από επιτραπέζιο υπολογιστή σε εικονική μηχανή (VM) ή από Linux σε λειτουργικό σύστημα Windows. Το Containerization εξαλείφει αυτό το πρόβλημα ομαδοποιώντας τον κωδικό εφαρμογής μαζί με τα σχετικά αρχεία διαμόρφωσης, βιβλιοθήκες και εξαρτήσεις που απαιτούνται για την εκτέλεση του. Αυτό το ενιαίο πακέτο λογισμικού ή κοντέινερ αφαιρείται από το λειτουργικό σύστημα του κεντρικού υπολογιστή, και ως εκ τούτου, είναι μόνο του και γίνεται φορητό - ικανό να τρέχει σε οποιαδήποτε πλατφόρμα ή cloud, χωρίς προβλήματα. Η έννοια του containerization είναι δεκαετιών, αλλά η εμφάνιση του ανοιχτού κώδικα Docker Engine το 2013, ένα βιομηχανικό πρότυπο για κοντέινερ με απλά εργαλεία προγραμματιστή και μια καθολική προσέγγιση packaging, επιτάχυνε την υιοθέτηση αυτής της τεχνολογίας. Η ερευνητική εταιρεία Gartner προβλέπει ότι περισσότερο από το 50% των εταιρειών θα χρησιμοποιήσουν τεχνολογία κοντέινερ έως το τέλος του 2020. Από τα αποτελέσματα από μια έρευνα στα τέλη του 2017 που διενήργησε η IBM δείχνουν ότι η υιοθέτηση πραγματοποιείται ακόμη γρηγορότερα, αποκαλύπτοντας ότι το 59% των εταιριών που προχώρησαν στη υιοθέτηση της τεχνολογίας του κοντέινερ βελτίωσε την ποιότητα της εφαρμογής και μείωσε τα ελαττώματα. Με απλά λόγια, το κοντέινερ επιτρέπει στις εφαρμογές να γράφονται μία φορά και να εκτελούνται οπουδήποτε. Αυτή η φορητότητα είναι σημαντική όσον αφορά τη διαδικασία ανάπτυξης και τη συμβατότητα του προμηθευτή. Προσφέρει επίσης άλλα αξιοσημείωτα οφέλη, όπως απομόνωση σφαλμάτων, ευκολία διαχείρισης και ασφάλειας.

Το κοντέινερ (container) είναι μια τυπική μονάδα λογισμικού που συσκευάζει κώδικα και όλες τις εξαρτήσεις του, έτσι ώστε η εφαρμογή να εκτελείται γρήγορα και αξιόπιστα από το ένα υπολογιστικό περιβάλλον στο άλλο. Μια εικόνα κοντέινερ (Container Image) είναι ένα ελαφρύ, αυτόνομο, εκτελέσιμο πακέτο λογισμικού που περιλαμβάνει όλα όσα χρειάζονται για την εκτέλεση μιας εφαρμογής: κωδικός, χρόνος εκτέλεσης, εργαλεία συστήματος, βιβλιοθήκες συστήματος και ρυθμίσεις. Οι εικόνες κοντέινερ γίνονται κοντέινερ κατά το χρόνο εκτέλεσης και στην περίπτωση κοντέινερ Docker - οι εικόνες γίνονται κοντέινερ όταν εκτελούνται στο Docker Engine. Διαθέσιμο για εφαρμογές Linux και Windows, το λογισμικό με κοντέινερ θα λειτουργεί πάντα το ίδιο, ανεξάρτητα από την υποδομή. Τα κοντέινερ απομονώνουν το λογισμικό από το περιβάλλον του και διασφαλίζουν ότι λειτουργεί ομοιόμορφα παρά τις διαφορές, για παράδειγμα, μεταξύ ανάπτυξης και σταδιοποίησης.



3.2 Οφέλη από τη χρήση του Containerization

Η τεχνολογία του Containerization προσφέρει σημαντικά οφέλη για προγραμματιστές και ομάδες ανάπτυξης. Μεταξύ αυτών είναι τα ακόλουθα:

- Φορητότητα: Ένα κοντέινερ δημιουργεί ένα εκτελέσιμο πακέτο λογισμικού που αφαιρείται (δεν συνδέεται ή εξαρτάται από) το λειτουργικό σύστημα κεντρικού υπολογιστή, και ως εκ τούτου, είναι φορητό και μπορεί να λειτουργεί ομοιόμορφα και με συνέπεια σε οποιαδήποτε πλατφόρμα ή σύννεφο (cloud).
- Ευελιξία: Η πηγή ανοιχτού κώδικα Docker Engine για την εκτέλεση κοντέινερ ξεκίνησε το βιομηχανικό πρότυπο για κοντέινερ με απλά εργαλεία προγραμματιστή και μια καθολική προσέγγιση packaging που λειτουργεί τόσο σε λειτουργικά συστήματα Linux όσο και σε Windows. Το σύστημα εμπορευματοκιβωτίων έχει μετατοπιστεί σε μηχανές που διαχειρίζεται η Open Container Initiative (OCI). Οι προγραμματιστές λογισμικού μπορούν να συνεχίσουν να χρησιμοποιούν ευέλικτα εργαλεία και διαδικασίες DevOps για ταχεία ανάπτυξη και βελτίωση εφαρμογών.
- Ταχύτητα: Τα εμπορευματοκιβώτια αναφέρονται συχνά ως «ελαφριά», που σημαίνει ότι μοιράζονται τον πυρήνα του λειτουργικού συστήματος (OS) του μηχανήματος και δεν έχουν κολλήσει με αυτήν την επιπλέον επιβάρυνση. Αυτό όχι μόνο αυξάνει την αποδοτικότητα του διακομιστή, αλλά μειώνει επίσης το κόστος διακομιστή και αδειοδότησης, ενώ επιταχύνει τους χρόνους έναρξης, καθώς δεν υπάρχει λειτουργικό σύστημα για εκκίνηση.
- Απομόνωση βλαβών: Κάθε εφαρμογή σε κοντέινερ είναι απομονωμένη και λειτουργεί ανεξάρτητα από άλλες. Η βλάβη ενός κοντέινερ δεν επηρεάζει τη συνεχιζόμενη λειτουργία οποιωνδήποτε άλλων κοντέινερ. Οι ομάδες ανάπτυξης μπορούν να εντοπίσουν και να διορθώσουν τυχόν τεχνικά ζητήματα σε ένα κοντέινερ χωρίς διακοπή λειτουργίας σε άλλα κοντέινερ. Επίσης, η μηχανή του κοντέινερ μπορεί να αξιοποιήσει οποιεσδήποτε τεχνικές απομόνωσης ασφάλειας λειτουργικού συστήματος - όπως SELinux access control - για την απομόνωση σφαλμάτων μέσα στο ίδιο το κοντέινερ.
- Αποδοτικότητα: Το λογισμικό που εκτελείται σε περιβάλλοντα κοντέινερ μοιράζεται τον πυρήνα λειτουργικού συστήματος του μηχανήματος και τα επίπεδα εφαρμογών εντός ενός κοντέινερ μπορούν να κοινοποιηθούν σε κοντέινερ. Έτσι, τα κοντέινερ είναι εγγενώς μικρότερα σε χωρητικότητα από ένα VM και απαιτούν λιγότερο χρόνο εκκίνησης, επιτρέποντας πολύ περισσότερα κοντέινερ να λειτουργούν στην ίδια χωρητικότητα με ένα μόνο VM. Αυτό αυξάνει την αποδοτικότητα του διακομιστή, μειώνοντας το κόστος διακομιστή και αδειοδότησης.
- Ευκολία διαχείρισης: Μια πλατφόρμα ενορχήστρωσης κοντέινερ αυτοματοποιεί την εγκατάσταση, κλιμάκωση και διαχείριση φόρτου εργασίας και υπηρεσιών με

κοντέινερ. Οι πλατφόρμες ενορχήστρωσης κοντέινερ μπορούν να διευκολύνουν εργασίες διαχείρισης, όπως κλιμάκωση εφαρμογών σε κοντέινερ, ανάπτυξη νέων εκδόσεων εφαρμογών και παροχή παρακολούθησης, καταγραφής και εντοπισμού σφαλμάτων, μεταξύ άλλων λειτουργιών. Το Kubernetes, ίσως το πιο δημοφιλές σύστημα ενορχήστρωσης κοντέινερ, είναι μια τεχνολογία ανοιχτού κώδικα που αυτοματοποιεί τις λειτουργίες κοντέινερ Linux. Το Kubernetes λειτουργεί με πολλούς κινητήρες εμπορευματοκιβωτίων, όπως το Docker, αλλά λειτουργεί επίσης με οποιοδήποτε σύστημα κοντέινερ που συμμορφώνεται με τα πρότυπα Open Container Initiative (OCI) για μορφές εικόνας κοντέινερ και χρόνους εκτέλεσης.

- Ασφάλεια: Η απομόνωση των εφαρμογών ως κοντέινερ αποτρέπει εγγενώς την εισβολή κακόβουλου κώδικα σε άλλα κοντέινερ ή στο σύστημα κεντρικού υπολογιστή. Επιπλέον, τα δικαιώματα ασφαλείας μπορούν να οριστούν ώστε να αποκλείουν αυτόματα τα ανεπιθύμητα στοιχεία από την είσοδο σε κοντέινερ ή να περιορίζουν τις επικοινωνίες με περιττούς πόρους.

3.3 Σύγκριση εννοιών Virtualization και Containerization

Αφού εξετάσαμε καθεμία από τις τεχνολογίες ξεχωριστά, το ερώτημα είναι, ποια πρέπει να προτιμάται; Η απάντηση εξαρτάται από πολλά διαφορετικά σημεία. Ουσιαστικά, κάθε επιχείρηση ή εφαρμογή έχει διαφορετικές ανάγκες, απαιτήσεις και σκοπό. Η επιλογή του virtualization έναντι του containerization εξαρτάται από την επιχειρηματική ανάπτυξη, το επιχειρησιακό μοντέλο ή τον τρόπο σύνταξης και παραγωγής των εφαρμογών. Και οι δύο είναι τεχνολογίες λογισμικού που δημιουργούν αυτόνομα εικονικά πακέτα, αλλά για να επιλέξουμε αυτό που θα ταιριάζει καλύτερα στις ανάγκες του χρήστη, θα εξετάσουμε τα ακόλουθα σημεία:

- Ταχύτητα: Όσον αφορά την ταχύτητα, τα κοντέινερ προορίζονταν να μειώσουν σημαντικά το χρόνο που απαιτείται για την ανάπτυξη και την εκτέλεση μιας εφαρμογής. Το κοντέινερ ξεκινά αμέσως, αφού το λειτουργικό σύστημα είναι ήδη σε λειτουργία, επομένως, η εφαρμογή θα ξεκινήσει χωρίς καμία καθυστέρηση. Αυτή είναι μια εξαιρετική λύση για ένα περιβάλλον ανάπτυξης, καθώς εξοικονομεί χρόνο στον κύκλο δοκιμών εφαρμογών (application testing cycle). Από την άλλη πλευρά, οι εικονικές μηχανές πρέπει να ξεκινήσουν ολόκληρο το λειτουργικό σύστημα, ενέργεια που περιλαμβάνει την πλήρη διαδικασία εκκίνησης. Αυτό θα περιλαμβάνει επίσης την εκκίνηση των υπηρεσιών (services) και θα διαρκέσει πολύ περισσότερο από ό, τι για ένα κοντέινερ.
- Πόροι: Δεδομένου ότι οι εικονικοί διακομιστές (virtual servers) εκτελούν ξεχωριστά λειτουργικά συστήματα και κάθε κλήση συστήματος πρέπει να περάσει από το επίπεδο εικονικοποίησης, δημιουργείται ένα ορισμένο ποσό γενικής επιβάρυνσης, με αποτέλεσμα να χρησιμοποιούνται περισσότεροι πόροι. Αυτό ισχύει ιδιαίτερα για τη χρήση της μνήμης, καθώς οι εικονικές μηχανές καταναλώνουν μνήμη ακόμα και όταν δεν εκτελούν καμία διαδικασία χρήστη. Ωστόσο, η εικονικοποίηση της CPU είναι σχετικά φθηνή, επομένως τα γενικά έξοδα της CPU μιας εικονικής μηχανής μπορεί να είναι πολύ μικρότερα. Όταν πρόκειται για κοντέινερ, μπορούν να ξεκινήσουν αρκετά γρήγορα, έτσι ώστε η κατανάλωση μνήμης τους να μην αυξάνεται. Υπάρχουν επίσης πολύ λιγότερα γενικά έξοδα, καθώς χρησιμοποιούν το ίδιο λειτουργικό σύστημα χωρίς να περάσουν από έναν επόπτη (hypervisor).

- Ασφάλεια και απομόνωση: Όσον αφορά την ασφάλεια και την απομόνωση, η εικονικοποίηση κερδίζει, καθώς διατηρεί τις εικονικές μηχανές ξεχωριστές και απομονωμένες μεταξύ τους. Μία “μολυσμένη” εικονική μηχανή δεν θα επηρεάσει άλλη, και κάθε εικονική μηχανή μπορεί να ενσωματώσει τα δικά της πρωτόκολλα ασφαλείας, καθώς εκτελούνται σε ένα πλήρως απομονωμένο περιβάλλον. Ωστόσο, δεδομένου ότι τα κοντέινερ απομονώνουν μόνο δεδομένα και εφαρμογές σε επίπεδο διαδικασίας, παρέχουν λιγότερο ασφαλές περιβάλλον και εξαρτώνται από τα πρωτόκολλα ασφαλείας του συστήματος κεντρικού υπολογιστή.
- Φορητότητα και κοινή χρήση εφαρμογών: Δεδομένου ότι οι εικόνες κοντέινερ είναι πολύ μικρότερες από τις εικονικές μηχανές, είναι ευκολότερη η μεταφορά και υπάρχει εξοικονόμηση χώρου στο σύστημα αρχείων του κεντρικού υπολογιστή. Οι εικονικές μηχανές, από την άλλη πλευρά, πρέπει να έχουν ένα αντίγραφο ολόκληρου του λειτουργικού συστήματος, συμπεριλαμβανομένων του πυρήνα, των βιβλιοθηκών του συστήματος, των αρχείων διαμόρφωσης, όλων των καταλόγων που απαιτούνται από το λειτουργικό σύστημα και όλων των βοηθητικών προγραμμάτων. Αυτό αυξάνει δραματικά το μέγεθος της εικόνας και δεν είναι τόσο εύκολο ο διαμοιρασμός (share). Οι εικόνες κοντέινερ μπορούν να κοινοποιούνται με οποιονδήποτε τρόπο και υπάρχουν αρκετοί κόμβοι κοινής χρήσης εφαρμογών στο Διαδίκτυο. Οι εικόνες της εικονικής μηχανής δεν έχουν τέτοια κεντρικά κέντρα και, συνήθως, θα πρέπει να μεταφορτωθούν (upload) σε άλλον διακομιστή (server).
- Απαιτήσεις λειτουργικού συστήματος: Μια εικονική μηχανή χρησιμοποιείται καλύτερα στην περίπτωση που μια επιχείρηση χρειάζεται να εκτελεί πολλές εφαρμογές που απαιτούν την πλήρη λειτουργικότητα ενός αποκλειστικού λειτουργικού συστήματος. Ωστόσο, εάν οι περισσότερες από τις εφαρμογές έχουν τις ίδιες απαιτήσεις λειτουργικού συστήματος, τα κοντέινερ θα ήταν μια αρκετά πιο πρακτική λύση.
- Κύκλος ζωής εφαρμογής: Τα κοντέινερ είναι εξαιρετικά κατάλληλα για βραχυπρόθεσμες ανάγκες εφαρμογής, καθώς μπορούν να ρυθμιστούν γρήγορα, είναι φορητά και μπορούν να ξεκινήσουν πολύ γρήγορα. Ωστόσο, περιορίζονται από την έλλειψη ειδικού λειτουργικού συστήματος, πόρων επεξεργασίας και αποθήκευσης. Τα κοντέινερ πρέπει να χρησιμοποιούνται όταν η μεγαλύτερη προτεραιότητα είναι η μεγιστοποίηση του αριθμού των εφαρμογών που εκτελούνται σε ελάχιστο αριθμό διακομιστών. Ωστόσο, οι εικονικές μηχανές είναι πολύ πιο κατάλληλες για εφαρμογές που πρέπει να χρησιμοποιούνται για μεγάλο χρονικό διάστημα, καθώς λειτουργούν σε εικονικό περιβάλλον που είναι πιο στιβαρό και ευέλικτο.

Ωστόσο, είναι σημαντικό να σημειωθεί ότι υπάρχουν τρόποι συνδυασμού containerization και virtualization έτσι ώστε τα πλεονεκτήματα και των δύο τεχνολογιών να συνδυάζονται. Ένας τέτοιος συνδυασμός ονομάζεται υβριδική αρχιτεκτονική κοντέινερ (hybrid container architecture) και μπορεί να επιτευχθεί βάζοντας μια εικονική μηχανή μέσα σε ένα κοντέινερ ή ένα μόνο κοντέινερ μέσα σε μια εικονική μηχανή ή πολλαπλά κοντέινερ μέσα σε μια εικονική μηχανή. Με αυτόν τον τρόπο, μπορούμε να πετύχουμε την ασφάλεια και την απομόνωση μιας εικονικής μηχανής σε συνδυασμό με τη γρήγορη και ελαφριά ρύθμιση μιας εφαρμογής στο εσωτερικό ενός κοντέινερ.

Πίνακας 1:Virtualization vs Containerization

Virtualization	Containerization
More secure and fully isolated	Less secure and isolated at the process level
Heavyweight, high resource usage	Lightweight, less resource usage
Hardware-level virtualization	Operating system virtualization
Each virtual machine runs in its own operating system	All containers share the host operating system
Startup time in minutes and slow provisioning	Startup time in milliseconds and quicker provisioning

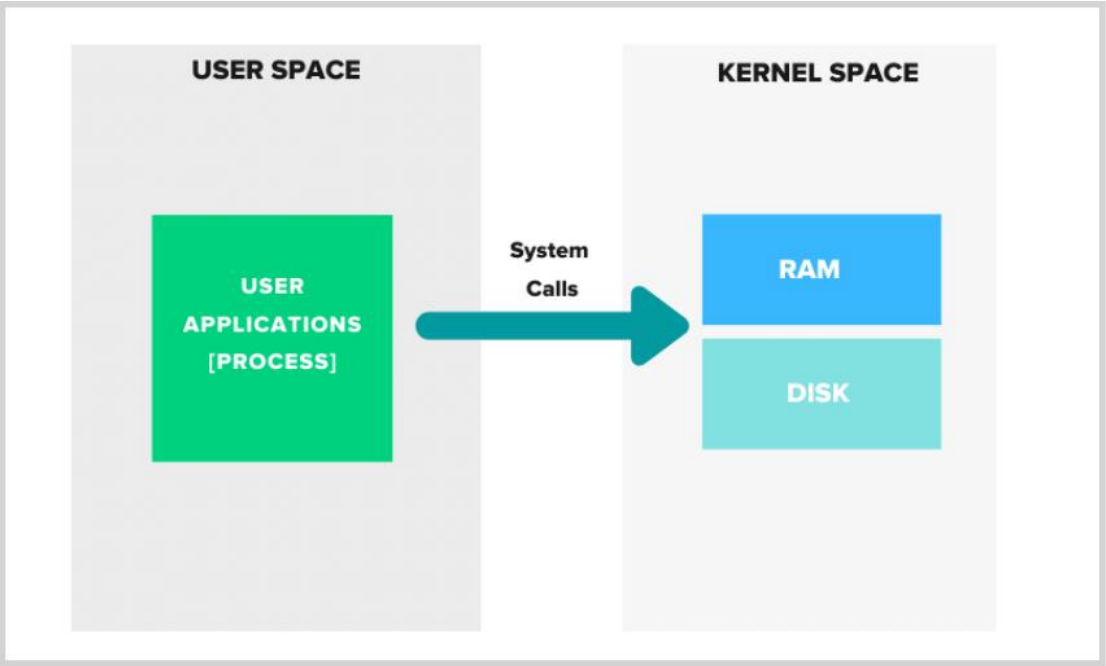
Συνοψίζοντας, μπορούμε να δούμε ότι κάθε τεχνολογία εξυπηρετεί έναν διαφορετικό σκοπό και ότι η επιλογή εξαρτάται σε μεγάλο βαθμό από τις ανάγκες του χρήστη και της εφαρμογής αλλά και από τη χωρητικότητα του διακομιστή. Δεδομένου ότι τόσο το virtualization όσο και το containerization έρχονται με σημαντικά πλεονεκτήματα και μειονεκτήματα, η επιλογή του ενός έναντι του άλλου πρέπει να γίνει προσεκτικά λαμβάνοντας υπόψη όλα αυτά τα σημεία.

3.4 Εισαγωγή στο εργαλείο Docker

Το Docker είναι μια ανοιχτή πλατφόρμα για ανάπτυξη, αποστολή και εκτέλεση εφαρμογών. Μας επιτρέπει να διαχωρίζουμε τις εφαρμογές (application) από τη δομή (infrastructure), ώστε να μπορούμε να παραδίδουμε το λογισμικό γρήγορα. Με το Docker, μπορούμε να διαμερίζουμε την δομή μας με τον ίδιο τρόπο που διαχειριζόμαστε τις εφαρμογές μας. Αξιοποιώντας τις μεθοδολογίες του Docker για αποστολή, δοκιμή και ανάπτυξη κώδικα, μπορούμε να μειώσουμε σημαντικά την καθυστέρηση μεταξύ σύνταξης κώδικα και εκτέλεσής του στην παραγωγή. Το Docker παρέχει τη δυνατότητα packaging και εκτέλεσης μιας εφαρμογής σε ένα απομονωμένο περιβάλλον που ονομάζεται κοντέινερ. Η απομόνωση και η ασφάλεια μας επιτρέπουν να εκτελούμε πολλά κοντέινερ ταυτόχρονα σε έναν συγκεκριμένο κεντρικό υπολογιστή. Τα κοντέινερ είναι ελαφριά και περιέχουν όλα όσα χρειάζονται για την εκτέλεση της εφαρμογής. Το Docker βελτιστοποιεί τον κύκλο ζωής ανάπτυξης επιτρέποντας στους προγραμματιστές να εργάζονται σε τυποποιημένα περιβάλλοντα χρησιμοποιώντας τοπικά κοντέινερ που παρέχουν τις εφαρμογές και τις υπηρεσίες. Τα κοντέινερ είναι ιδανικά για συνεχή ολοκλήρωση (continuous integration, CI) και ροές εργασίας συνεχούς παράδοσης (continuous delivery, CD). Το Docker έχει γίνει το πρότυπο όταν πρόκειται για εφαρμογές που βασίζονται σε κοντέινερ. Από εφαρμογές μικρής κλίμακας έως εταιρικές εφαρμογές μεγάλης κλίμακας, το Docker χρησιμεύει ως βάση για ενορχήστρωση με βάση κοντέινερ. Το Docker κέρδισε τόση δημοτικότητα και υιοθέτηση στην κοινότητα DevOps σε σύντομο χρονικό διάστημα, λόγω του τρόπου με τον οποίο αναπτύχθηκε έτσι ώστε να παρέχει φορητότητα και καθώς έχει σχεδιαστεί για σύγχρονη αρχιτεκτονική μικροϋπηρεσιών. Η Google χρησιμοποιεί τη δική της τεχνολογία κοντέινερ στην υποδομή της εδώ και χρόνια. Η έννοια των κοντέινερ ξεκίνησε από τη δεκαετία του 2000. Στην πραγματικότητα, οι ρίζες ξεκινούν το 1979 όπου είχαμε chroot, μια έννοια της αλλαγής του ριζικού καταλόγου μιας διαδικασίας.

3.4.1 Αναφορά στο Linux container(LXC)

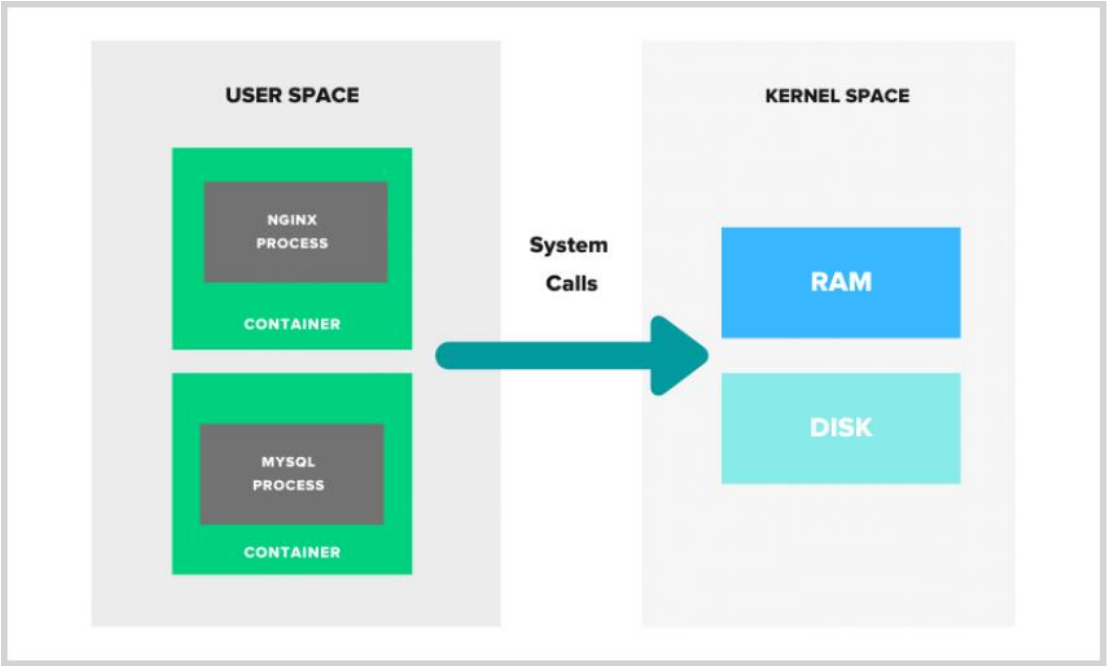
Πριν ξεκινήσουμε απευθείας στις έννοιες του Docker, πρώτα, πρέπει να καταλάβουμε τί είναι το Linux Container. Σε ένα τυπικό εικονικοποιημένο περιβάλλον, μία ή περισσότερες εικονικές μηχανές εκτελούνται πάνω από έναν φυσικό διακομιστή χρησιμοποιώντας έναν επόπτη όπως το Xen, το Hyper-V κ.λπ. Τα containers, από την άλλη πλευρά, τρέχουν πάνω από τον πυρήνα των λειτουργικών συστημάτων. Μπορούμε να το ονομάσουμε ως εικονικοποίηση σε επίπεδο λειτουργικού συστήματος. Πριν μπούμε στις βασικές έννοιες κοντέινερ, πρέπει να κατανοήσουμε τις δύο βασικές Linux έννοιες.



Εικόνα 17:User space Kernal space

- Userspace: Όλος ο κώδικας που απαιτείται για την εκτέλεση προγραμμάτων χρηστών (εφαρμογές, διαδικασία) ονομάζεται userpace. Όταν ξεκινάτε μια ενέργεια προγράμματος, για παράδειγμα, για να δημιουργήσετε ένα αρχείο, η διαδικασία στο χώρο χρήστη πραγματοποιεί μια κλήση συστήματος στο Kernel space.
- Kernel Space: Αυτή είναι η καρδιά του λειτουργικού συστήματος, όπου έχουμε τον kernel code που αλληλοεπιδρά με το υλικό του συστήματος, την αποθήκευση κ.λπ.

Όταν ξεκινάμε μια εφαρμογή, για παράδειγμα, έναν Nginx web server, ξεκινάμε στην πραγματικότητα μια «διαδικασία», process. Η ίδια η διαδικασία είναι μια αυτόνομη οδηγία με περιορισμένη απομόνωση. Τί θα γινόταν αν μπορούμε να απομονώσουμε τη διαδικασία μόνο με αρχεία και ρυθμίσεις που απαιτούνται για την εκτέλεση και τη λειτουργία της; Αυτό ακριβώς κάνει ένα container. Ένα κοντέινερ είναι βασικά μια διαδικασία με αρκετή απομόνωση των στοιχείων χώρου χρήστη έτσι ώστε να δίνει την αίσθηση ενός ξεχωριστού λειτουργικού συστήματος.



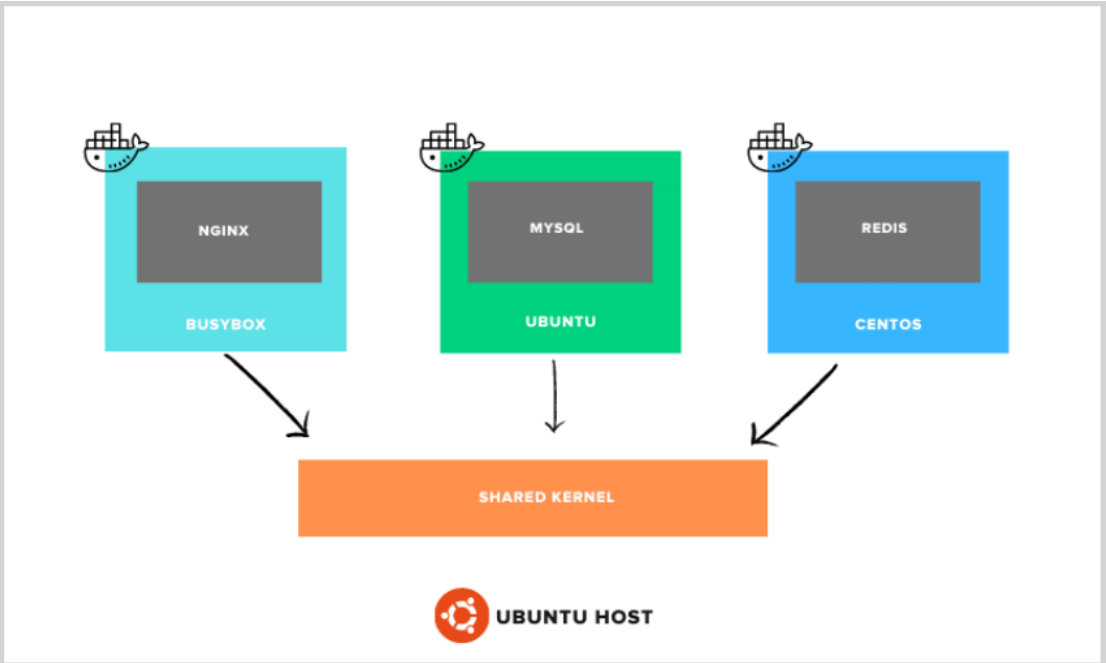
Η διαδικασία γονικού κοντέινερ μπορεί να έχει θυγατρική διαδικασία. Έτσι μπορούμε να πούμε πως, ένα κοντέινερ είναι επίσης και μια ομάδα διαδικασιών. Για παράδειγμα, όταν ξεκινάμε μια υπηρεσία Nginx, ξεκινά μια γονική διαδικασία Nginx. Η γονική διαδικασία εκτείνεται έπειτα στις θυγατρικές της διαδικασίες, όπως cache manager, cache loader και workers.

Image Src: https://www.nginx.com

Έτσι, όταν ξεκινά ένα κοντέινερ Nginx, ξεκινά μια κύρια διαδικασία Nginx σε απομονωμένο περιβάλλον.

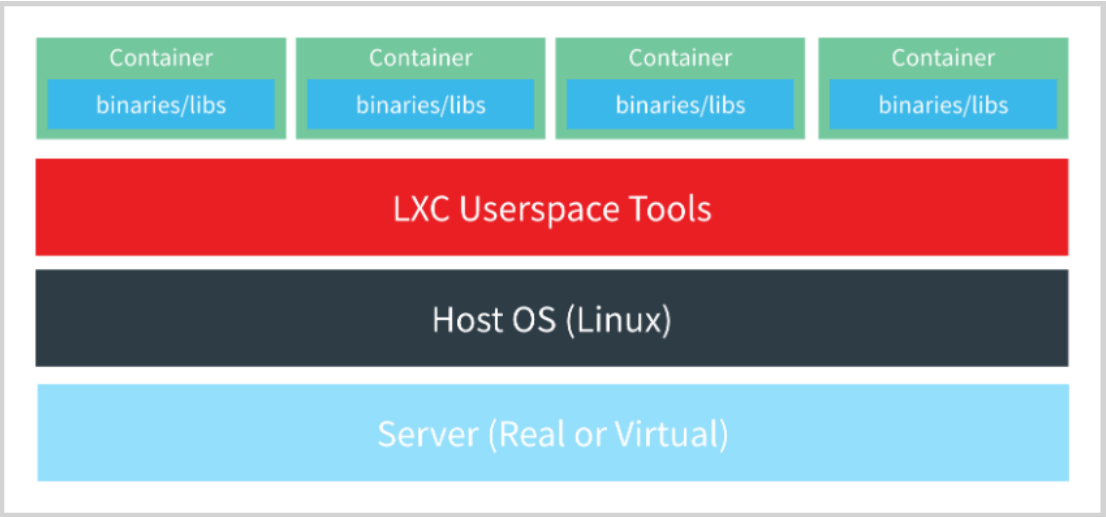
Κάθε κοντέινερ έχει τον απομονωμένο χώρο χρήστη και μπορούμε να εκτελούμε πολλά κοντέινερ σε έναν κεντρικό υπολογιστή. Αυτό σημαίνει ότι ένα κοντέινερ έχει ολόκληρο το

λειτουργικό σύστημα; Όχι. Σε αντίθεση με ένα VM με τον δικό του πυρήνα, ένα κοντέινερ περιέχει μόνο τα απαιτούμενα αρχεία που σχετίζονται με μια συγκεκριμένη διανομή και χρησιμοποιεί τον κοινόχρηστο πυρήνα κεντρικού υπολογιστή. Το πιο σημαντικό είναι ότι μπορούμε να εκτελέσουμε διαφορετικά κοντέινερ που βασίζονται σε διανομές Linux (Linux distros based containers) σε έναν κεντρικό υπολογιστή που μοιράζεται τον ίδιο χώρο στον πυρήνα (kernel space).



Για παράδειγμα, μπορούμε να εκτελέσουμε ένα RHEL, CentOS, ένα κοντέινερ βασισμένο σε SUSE σε διακομιστή Ubuntu. Είναι πιθανό επειδή για όλες τις διανομές Linux (Linux distros) και ο χώρος του πυρήνα (kernel space) είναι ο ίδιος αλλά μόνο ο χώρος χρηστών (userspace) είναι διαφορετικός.

3.4.2 Αναφορά στις έννοιες Linux Containers και Namespaces



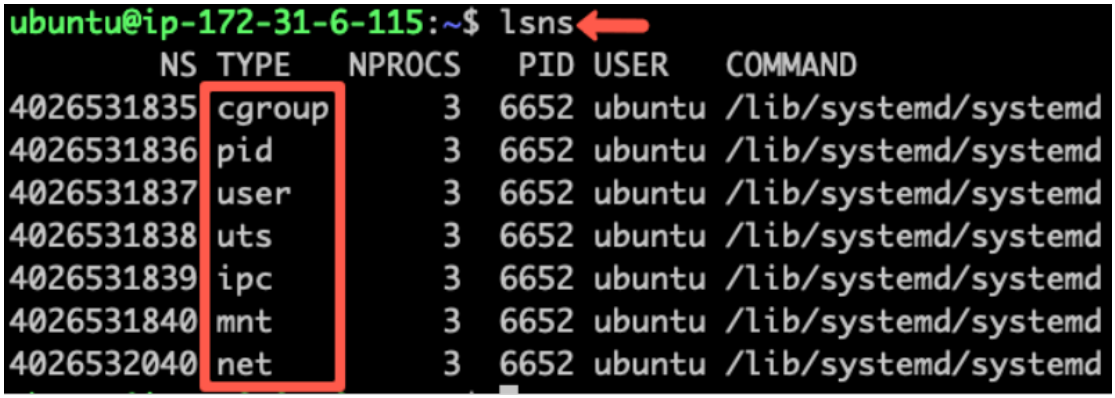
Για να απομονώσουμε κοντέινερ με τη δική τους CPU, μνήμη, διεύθυνση IP, σημεία προσάρτησης, διαδικασίες, χρειαζόμαστε δύο λειτουργίες πυρήνα Linux που ονομάζονται χώροι ονομάτων (namespaces) και ομάδες ελέγχου (control groups).

Ένα κοντέινερ το μόνο που απαιτεί για την άριστη λειτουργία του είναι ένα απομονωμένο περιβάλλον ώστε να γίνει η εκτέλεση μιας υπηρεσίας (Process). Για να επιτευχθεί αυτό το επίπεδο απομόνωσης, ένα κοντέινερ πρέπει να έχει το δικό του σύστημα αρχείων, διεύθυνση IP, σημεία προσάρτησης, αναγνωριστικά διεργασίας κ.λπ. Μπορούμε να το επιτύχετε χρησιμοποιώντας τους Χώρους ονομάτων Linux. Τα Namespaces είναι υπεύθυνα για τα κοντέινερ να έχουν τα δικά τους σημεία προσάρτησης, χρήστη, διεύθυνση IP, διαχείριση διεργασιών κ.λπ. Ουσιαστικά θέτουν τα όρια στα κοντέινερ.

Τα “key namespaces” στο Linux:

- pid namespace: Υπεύθυνος για την απομόνωση της διαδικασίας (PID: Process ID).
- net namespace: Διαχειρίζεται διασυνδέσεις δικτύου (NET: Networking).
- ipc namespace: Διαχειρίζεται την πρόσβαση σε πόρους IPC (IPC: InterProcess Communication).
- mnt namespace: Υπεύθυνος για τη διαχείριση των σημείων προσάρτησης του συστήματος αρχείων (MNT: Mount).
- uts namespace: Απομόνωση πυρήνα και αναγνωριστικών έκδοσης. (UTS: Unix Timesharing System).
- usr namespace: Απομόνωση αναγνωριστικών χρήστη. Με απλά λόγια, απομονώνει τα αναγνωριστικά χρήστη μεταξύ του κεντρικού υπολογιστή και του κοντέινερ.
- Cgroup namespace: Απομόνωση των πληροφοριών της ομάδας ελέγχου από τη διαδικασία κοντέινερ.

Χρησιμοποιώντας αυτούς τους χώρους ονομάτων ένα κοντέινερ μπορεί να έχει τις δικές του διεπαφές δικτύου, διεύθυνση IP κ.λπ. Κάθε κοντέινερ θα έχει το δικό του χώρο ονομάτων και οι διαδικασίες που εκτελούνται εντός αυτού του χώρου ονομάτων δεν θα έχουν προνόμια εκτός του χώρου ονομάτων του. Είναι ενδιαφέρον ότι μπορούμε να παραθέσουμε τους χώρους ονομάτων σε μια μηχανή Linux χρησιμοποιώντας την εντολή “lsns”.



3.4.3 Αναφορά στα Linux Control Groups

Όταν ξεκινάμε μια υπηρεσία, δεν καθορίζουμε κανένα όριο μνήμης ή όριο χρήσης τη CPU. Το αφήνουμε στον πυρήνα να δώσει προτεραιότητα και να διαθέσει πόρους για τις υπηρεσίες. Ωστόσο, μπορούμε να ορίσουμε ρητά όρια χρήσης της CPU και όρια μνήμης για τις υπηρεσίες μας χρησιμοποιώντας μια λειτουργία πυρήνα Linux που ονομάζεται CGroups. Δεν πρόκειται για μια απλή προσέγγιση, πρέπει να κάνουμε κάποιες επιπλέον ρυθμίσεις και τροποποιήσεις για να λειτουργήσει.. Δεδομένου ότι μπορούμε να εκτελέσουμε πολλά κοντέινερ μέσα σε έναν κεντρικό υπολογιστή, πρέπει να υπάρχει ένας μηχανισμός για τον περιορισμό της χρήσης πόρων, της πρόσβασης σε συσκευές κτλπ. Η διαχείριση των πόρων από ένα κοντέινερ γίνεται από ομάδες ελέγχου Linux (Linux control groups). Μπορούμε να περιορίσουμε τους πόρους που χρησιμοποιεί η CPU, τη μνήμης, το δικτύου και IO από ομάδες ελέγχου Linux κοντέινερ.

Τι συμβαίνει λοιπόν εάν δεν περιορίσω τον πόρο CPU και μνήμης ενός κοντέινερ; Ένα μεμονωμένο κοντέινερ ενδέχεται να καταλήξει να χρησιμοποιεί όλους τους πόρους του κεντρικού υπολογιστή αφήνοντας άλλα κοντέινερ να καταρρεύσουν λόγω μη διαθεσιμότητας πόρων. Εργαλεία όπως το docker αφαιρεί όλες τις σύνθετες διαμορφώσεις backend και μας επιτρέπει να καθορίσουμε αυτά τα όρια πόρων με απλές παραμέτρους.

3.5 Αρχιτεκτονική του εργαλείου Docker

Στις ακόλουθες ενότητες, θα εξετάσουμε την αρχιτεκτονική του Docker και τα σχετικά στοιχεία της. Θα εξετάσουμε επίσης πώς λειτουργεί κάθε στοιχείο για να κάνει το Docker να λειτουργεί. Η αρχιτεκτονική του Docker χρησιμοποιεί ένα μοντέλο διακομιστή-πελάτη και περιλαμβάνει τα στοιχεία Docker Client, Docker Host, Network and Storage και το Docker Registry / Hub. Η αρχιτεκτονική του Docker έχει αλλάξει μερικές φορές από την ίδρυσή του. Εδώ είναι μερικές αξιοσημείωτες αρχιτεκτονικές αλλαγές που συνέβησαν για το Docker:

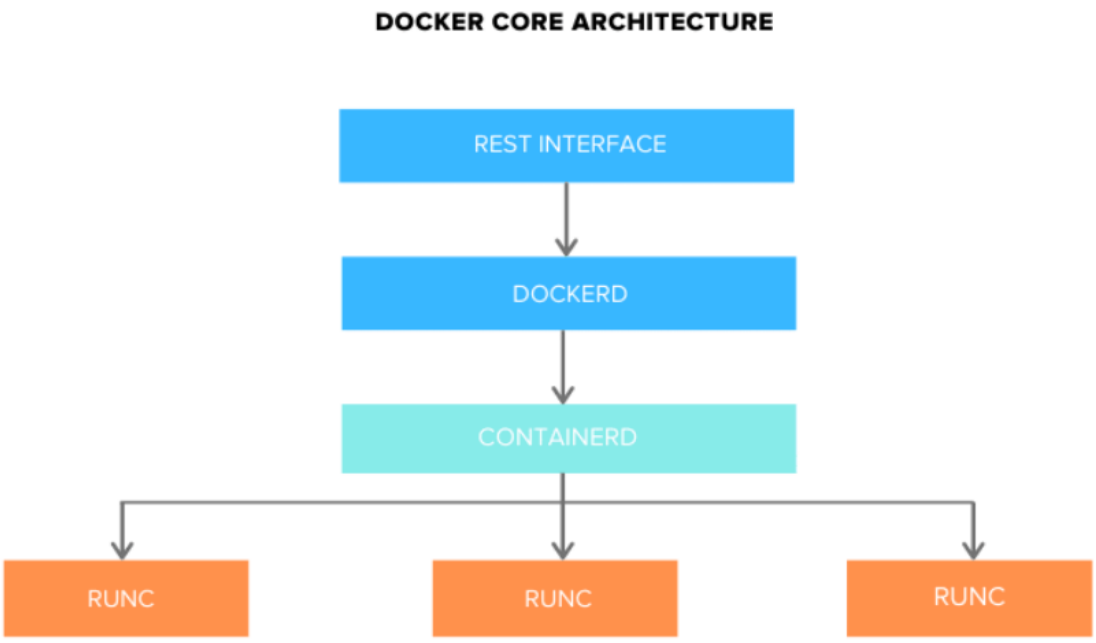
- Η Docker «άλλαξε» από το LXC στο libcontainer το 2014.
- runc - ένα CLI για containers που ακολουθούν όλες τις προδιαγραφές OCI.
- containerd - Η Docker διαχώρισε το στοιχείο διαχείρισης κοντέινερ με το containerd το 2016.

Το Open Container Initiative (OCI) είναι ένα ανοιχτό βιομηχανικό πρότυπο για το χρόνο εκτέλεσης των κοντέινερ και τις προδιαγραφές.

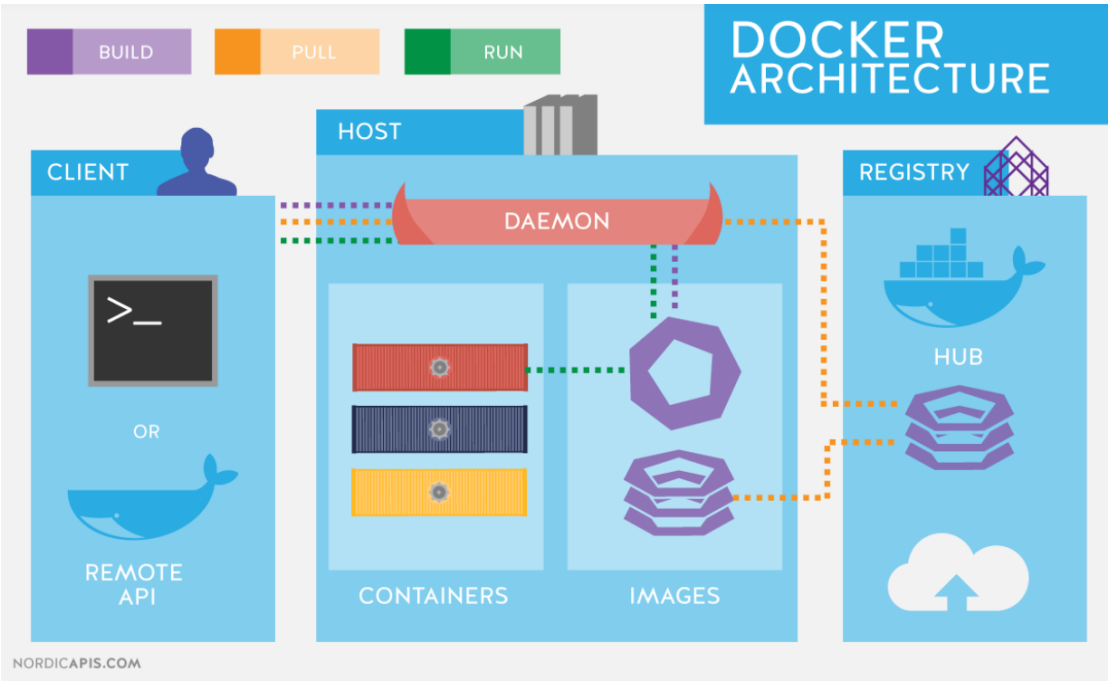
Όταν ξεκίνησε αρχικά το docker, είχε μια μονολιθική αρχιτεκτονική. Τώρα χωρίζεται σε ακόλουθα τρία διαφορετικά στοιχεία:

1. Docker Engine (dockerd).
2. Docker-containerd (containerd)
3. Docker-runc (runc)

Το Docker και άλλοι μεγάλοι οργανισμοί αποφάσισαν να συνεισφέρουν σε ένα κοινό επίπεδο χρόνου εκτέλεσης και διαχείρισης κοντέινερ. Ως εκ τούτου, το containerd και το runc αποτελούν πλέον μέρος του Cloud Native Foundation με συνεισφέροντες από όλους τους οργανισμούς. Κατά την εγκατάσταση του Docker, εγκαθίστανται όλα αυτά τα στοιχεία. Δεν χρειάζεται να τα εγκαταστήσουμε ξεχωριστά.



Εικόνα : Docker Core Architecture.



Εικόνα:Docker Architecture.

Ο Docker Client επιτρέπει στους χρήστες να αλληλοεπιδρούν με το Docker. Ο Docker Client μπορεί να βρίσκεται στον ίδιο κεντρικό υπολογιστή με τον Daemon ή να συνδεθεί με έναν Daemon σε έναν απομακρυσμένο κεντρικό υπολογιστή. Ένας Docker Client μπορεί να

επικοινωνήσει με περισσότερους από έναν daemon. Επίσης παρέχει μια διεπαφή γραμμής εντολών (CLI) που μας επιτρέπει να δίνουμε εντολές κατασκευής, εκτέλεσης και διακοπής εντολών σε έναν daemon Docker.

Ο κύριος σκοπός του Docker Client είναι να παρέχει ένα μέσο για να κατευθύνει το «τράβηγμα» των εικόνων από ένα μητρώο (registry) και να το τρέχει σε έναν κεντρικό υπολογιστή Docker. Οι κοινές εντολές που δίνονται από έναν πελάτη είναι:

- Docker build
- Docker pull
- Docker run

Τα Docker Registries είναι υπηρεσίες που παρέχουν τοποθεσίες από τις οποίες μπορούμε να αποθηκεύσουμε και να κατεβάσουμε εικόνες Docker. Με άλλα λόγια, ένα Docker Registry περιέχει Docker repositories που φιλοξενούν μία ή περισσότερες εικόνες Docker. Τα δημόσια registries περιλαμβάνουν το Docker Hub και το Docker Cloud. Μπορούν επίσης να χρησιμοποιηθούν ιδιωτικά registries.

Ο Docker Host παρέχει ένα πλήρες περιβάλλον για την εκτέλεση (execute και run) εφαρμογών. Αποτελείται από τον Docker Daemon , Images, containers, networks και Storage (αποθήκευση). Όπως αναφέρθηκε προηγουμένως, ο δαίμονας είναι υπεύθυνος για όλες τις ενέργειες που σχετίζονται με κοντέινερ και λαμβάνει εντολές μέσω του CLI ή του REST API. Μπορεί επίσης να επικοινωνήσει με άλλους δαίμονες για τη διαχείριση των υπηρεσιών του. Ο δαίμονας Docker τραβά και δημιουργεί εικόνες κοντέινερ όπως ζητήθηκε από τον πελάτη. Μόλις τραβήξει μια ζητούμενη εικόνα, δημιουργεί ένα λειτουργικό μοντέλο για το κοντέινερ χρησιμοποιώντας ένα σύνολο οδηγιών γνωστών ως αρχείο build. Το αρχείο build μπορεί επίσης να περιλαμβάνει οδηγίες για τον δαίμονα να προ φορτώνει άλλα στοιχεία πριν από την εκτέλεση του κοντέινερ ή οδηγίες για αποστολή στην τοπική γραμμή εντολών μετά την κατασκευή του κοντέινερ.

Οι εικόνες (Images) είναι ένα δυαδικό πρότυπο μόνο για ανάγνωση που χρησιμοποιείται για τη δημιουργία κοντέινερ. Οι εικόνες περιέχουν επίσης μεταδεδομένα που περιγράφουν τις δυνατότητες και τις ανάγκες του κοντέινερ. Οι εικόνες χρησιμοποιούνται για την αποθήκευση και αποστολή εφαρμογών. Μια εικόνα μπορεί να χρησιμοποιηθεί από μόνη της για την κατασκευή ενός κοντέινερ ή προσαρμοσμένη για την προσθήκη πρόσθετων στοιχείων για την επέκταση της τρέχουσας διαμόρφωσης. Οι εικόνες κοντέινερ μπορούν να κοινοποιηθούν σε ομάδες εντός μιας επιχείρησης που χρησιμοποιούν ένα ιδιωτικό μητρώο κοντέινερ ή να κοινοποιηθούν στον κόσμο χρησιμοποιώντας ένα δημόσιο μητρώο όπως το Docker Hub. Οι εικόνες αποτελούν βασικό μέρος του Docker καθώς επιτρέπουν τη συνεργασία μεταξύ προγραμματιστών με έναν τρόπο που δεν ήταν δυνατό πριν.

Τα κοντέινερ είναι ενθυλακωμένα περιβάλλοντα στα οποία εκτελείτε εφαρμογές. Το κοντέινερ ορίζεται από την εικόνα και τυχόν πρόσθετες επιλογές διαμόρφωσης που παρέχονται κατά την εκκίνηση του κοντέινερ, και δεν περιορίζονται στις συνδέσεις δικτύου και στις επιλογές αποθήκευσης. Τα κοντέινερ έχουν πρόσβαση μόνο σε πόρους που ορίζονται στην εικόνα, εκτός εάν ορίζεται πρόσθετη πρόσβαση κατά την κατασκευή της εικόνας σε κοντέινερ. Μπορούμε επίσης να δημιουργήσουμε μια νέα εικόνα με βάση την τρέχουσα κατάσταση ενός κοντέινερ. Δεδομένου ότι τα κοντέινερ είναι πολύ μικρότερα από τα VM, μπορούν να περιστραφούν σε λίγα δευτερόλεπτα και να έχουν πολύ καλύτερη πυκνότητα διακομιστή (server density).

Μπορούμε να αποθηκεύσουμε δεδομένα εντός του εγγράψιμου επιπέδου ενός κοντέινερ, αλλά απαιτείται πρόγραμμα οδήγησης αποθήκευσης. Όντας μη-ανθεκτικό (non persistent), χάνεται κάθε φορά που το container δεν λειτουργεί. Επιπλέον, δεν είναι εύκολο να μεταφέρουμε αυτά τα δεδομένα. Όσον αφορά τη μόνιμη αποθήκευση, το Docker προσφέρει τέσσερις επιλογές:

1. Data Volumes: Παρέχουν τη δυνατότητα δημιουργίας μόνιμου χώρου αποθήκευσης, με τη δυνατότητα μετονομασίας τόμων, λίστας τόμων ή λίστα με το κοντέινερ που σχετίζεται με τον τόμο. Οι όγκοι δεδομένων τοποθετούνται στο σύστημα αρχείων κεντρικού υπολογιστή, έξω από το μηχανισμό αντιγραφής σε εγγραφή και είναι αρκετά αποδοτικοί.
2. Το Data Volume Container είναι μια εναλλακτική προσέγγιση όπου ένα ειδικό κοντέινερ φιλοξενεί έναν τόμο και για να τοποθετήσει αυτόν τον τόμο σε άλλα δοχεία. Σε αυτήν την περίπτωση, το volume container είναι ανεξάρτητο από το κοντέινερ εφαρμογής και επομένως μπορεί να μοιραστεί σε περισσότερα από ένα container.
3. Directory Mounts: Μια άλλη επιλογή είναι να προσαρτήσουμε τον τοπικό κατάλογο ενός κεντρικού υπολογιστή σε ένα κοντέινερ. Στις προαναφερθείσες περιπτώσεις, οι τόμοι θα πρέπει να βρίσκονται εντός του φακέλου τόμων Docker, ενώ όταν πρόκειται για Directory Mounts, οποιοσδήποτε κατάλογος στον κεντρικό υπολογιστή μπορεί να χρησιμοποιηθεί ως πηγή για τον τόμο.
4. Storage Plugins: Τα Storage Plugins παρέχουν τη δυνατότητα σύνδεσης σε εξωτερικές πλατφόρμες αποθήκευσης. Αυτά τα πρόσθετα αντιστοιχούν στον χώρο αποθήκευσης από τον κεντρικό υπολογιστή σε μια εξωτερική πηγή, όπως ένας πίνακας αποθήκευσης ή μια συσκευή. Μπορούμε να βρούμε μια λίστα προσθηκών αποθήκευσης στη σελίδα προσθήκης του Docker. Υπάρχουν πρόσθετα αποθήκευσης από διάφορες εταιρείες για την αυτοματοποίηση της διαδικασίας παροχής αποθήκευσης. Για παράδειγμα, HPE 3PAR, EMC (ScaleIO, XtremIO, VMAX, Isilon), NetApp. Υπάρχουν επίσης plugins που υποστηρίζουν δημόσιους παρόχους cloud όπως: Azure File Storage, Google Compute Platform.

3.6 Επίπεδο δικτύωσης του Docker

Προκειμένου τα Docker containers να επικοινωνούν μεταξύ τους και με τον εξωτερικό κόσμο μέσω ενός κεντρικού υπολογιστή, πρέπει να υπάρχει ένα επίπεδο δικτύωσης. Το Docker υποστηρίζει διαφορετικούς τύπους δικτύων, κάθε ένα κατάλληλο για συγκεκριμένες περιπτώσεις χρήσης. Για παράδειγμα, η κατασκευή μιας εφαρμογής που εκτελείται σε ένα κοντέινερ Docker θα έχει διαφορετική ρύθμιση δικτύου σε σύγκριση με μια εφαρμογή ιστού με ένα σύμπλεγμα με βάση δεδομένων, εξισορροπητές εφαρμογών και φορτίων που καλύπτουν πολλά κοντέινερ που πρέπει να επικοινωνούν μεταξύ τους. Επιπλέον, οι πελάτες από τον εξωτερικό κόσμο θα πρέπει να έχουν πρόσβαση στο κοντέινερ εφαρμογών ιστού. Όταν εγκαθίσταται το Docker, δημιουργείται ένα προεπιλεγμένο δίκτυο γέφυρας με το όνομα docker0. Κάθε νέο κοντέινερ Docker συνδέεται αυτόματα σε αυτό το δίκτυο, εκτός εάν έχει καθοριστεί προσαρμοσμένο δίκτυο. Εκτός από το docker0, δύο άλλα δίκτυα δημιουργούνται αυτόματα από το Docker: host (χωρίς απομόνωση μεταξύ κεντρικού υπολογιστή και κοντέινερ σε αυτό το δίκτυο, στον εξωτερικό κόσμο βρίσκονται στο ίδιο

δίκτυο) και το ‘none’.(τα συνδεδεμένα κοντέινερ λειτουργούν σε στοίβα δικτύου για συγκεκριμένα κοντέινερ).

Το Docker έρχεται με προγράμματα οδήγησης δικτύου προσανατολισμένα σε διαφορετικές περιπτώσεις χρήσης. Οι πιο συνηθισμένοι τύποι δικτύου είναι: Bridge, Overlay και macvlan Network.

1. Bridge Network: Η δικτύωση Bridge είναι ο πιο κοινός τύπος δικτύου. Περιορίζεται σε κοντέινερ σε έναν μόνο κεντρικό υπολογιστή που λειτουργεί με τον κινητήρα Docker. Τα δίκτυα Bridge είναι εύκολο να δημιουργηθούν, να διαχειριστούν και να αντιμετωπιστούν προβλήματα. Παρόμοιο με το προεπιλεγμένο δίκτυο γέφυρας, ένα δίκτυο Bridge που καθορίζεται από το χρήστη διαφέρει στο ότι δεν υπάρχει ανάγκη προώθησης θύρας για κοντέινερ εντός του δικτύου ώστε να επικοινωνούν μεταξύ τους. Η άλλη διαφορά είναι ότι έχει πλήρη υποστήριξη για αυτόματη ανάνηψη δικτύου. Προκειμένου τα containers στο bridge network να επικοινωνούν ή να είναι προσβάσιμα από τον εξωτερικό κόσμο, πρέπει να ρυθμιστεί η χαρτογράφηση των θυρών (port mapping). Για παράδειγμα, θεωρήστε ότι μπορείτε να έχετε ένα κοντέινερ Docker που εκτελεί μια υπηρεσία ιστού στη θύρα 80. Επειδή αυτό το κοντέινερ είναι συνδεδεμένο στο δίκτυο γέφυρας σε ένα ιδιωτικό υποδίκτυο, μια θύρα στο σύστημα κεντρικού υπολογιστή, όπως το 8000, πρέπει να αντιστοιχιστεί στη θύρα 80 στο κοντέινερ για εξωτερική κίνηση για να φτάσει στην υπηρεσία διαδικτύου. Για να δημιουργήσετε ένα δίκτυο γέφυρας με το όνομα my-bridge-net, περνάμε το όρισμα bridge με τη -d (driver) όπως φαίνεται παρακάτω:
   ```
   $ docker network create -d bridge my-bridge-net
   ```
2. Overlay Network: Ένα δίκτυο επικάλυψης χρησιμοποιείται όταν χρειαζόμαστε κοντέινερ σε ξεχωριστούς κεντρικούς υπολογιστές για να μπορούμε να επικοινωνούμε μεταξύ τους, όπως στην περίπτωση ενός κατανεμημένου δικτύου. Ωστόσο, μια προειδοποίηση είναι ότι η λειτουργία σμήνους πρέπει να είναι ενεργοποιημένη για ένα σύμπλεγμα κινητήρων Docker, γνωστό ως σμήνος (swarm), για να μπορεί να συμμετέχει στην ίδια ομάδα.
3. Macvlan Network: Όταν χρησιμοποιείτε δίκτυα Bridge και Overlay, μια γέφυρα βρίσκεται μεταξύ του κοντέινερ και του κεντρικού υπολογιστή. Ένα δίκτυο Macvlan καταργεί αυτήν τη γέφυρα, παρέχοντας το πλεονέκτημα της έκθεσης πόρων κοντέινερ σε εξωτερικά δίκτυα χωρίς να ασχολείται με port forwarding. Αυτό πραγματοποιείται χρησιμοποιώντας διευθύνσεις MAC αντί για διευθύνσεις IP. Να σημειωθεί ότι το macvlan πρέπει να ρυθμιστεί ανά κεντρικό υπολογιστή και διαθέτει υποστήριξη για φυσικό NIC, υπο-διεπαφή (sub-interface) , διεπαφές συνδεδεμένες με δίκτυο, ακόμη και ομαδοποιημένες διεπαφές. Η επισκεψιμότητα φιλτράρεται ρητά από τις μονάδες πυρήνα κεντρικού υπολογιστή (host kernel modules) για απομόνωση και ασφάλεια. Για να δημιουργήσουμε ένα δίκτυο macvlan που ονομάζεται macvlan -net, θα πρέπει να δώσουμε μια παράμετρο --gateway για να καθορίσουμε τη διεύθυνση IP της πύλης για το υποδίκτυο και μια παράμετρο -o για να ορίσουμε συγκεκριμένες επιλογές προγράμματος

οδήγησης. Σε αυτό το παράδειγμα, η μητρική διεπαφή έχει οριστεί σε eth0 interface στον κεντρικό υπολογιστή:

```
$ docker network create -d macvlan \
  --subnet=192.168.40.0/24 \
  --gateway=192.168.40.1 \
  -o parent=eth0 my-macvlan-net
```

3.7 Πόρτες που κοινοποιούνται και DNS υπηρεσίες.

Από προεπιλογή, όταν δημιουργούμε ή εκτελούμε ένα κοντέινερ χρησιμοποιώντας `docker create` ή `docker run`, δεν δημοσιεύεται καμία από τις θύρες του στον εξωτερικό κόσμο. Για να κάνουμε μια θύρα διαθέσιμη σε υπηρεσίες εκτός του Docker ή σε container Docker που δεν είναι συνδεδεμένα στο δίκτυο του container, χρησιμοποιούμε `--publish` ή `-p` flags. Αυτό δημιουργεί έναν κανόνα τείχους προστασίας που αντιστοιχίζει μια θύρα κοντέινερ σε μια θύρα του κεντρικού υπολογιστή Docker στον εξωτερικό κόσμο. Να μερικά παραδείγματα.

3.8 Πώς επιτυγχάνεται η επικοινωνία μεταξύ των containers

Διαφορετικά δίκτυα παρέχουν διαφορετικά πρότυπα επικοινωνίας (για παράδειγμα μόνο με διεύθυνση IP ή με όνομα κοντέινερ) μεταξύ των κοντέινερ ανάλογα με τον τύπο του δικτύου και το αν είναι προεπιλεγμένο Docker ή δίκτυο που καθορίζεται από τον χρήστη.

- Εύρεση container στο δίκτυο docker0 (DNS resolution)

  Το Docker θα εκχωρήσει ένα όνομα και ένα όνομα κεντρικού υπολογιστή (name, hostname) σε κάθε container που έχει δημιουργηθεί στο προεπιλεγμένο δίκτυο docker0, εκτός εάν έχει οριστεί διαφορετικό όνομα/όνομα κεντρικού υπολογιστή από τον χρήστη. Στη συνέχεια, το Docker διατηρεί μια αντιστοίχιση κάθε ονόματος/ονόματος κεντρικού υπολογιστή με τη διεύθυνση IP του container. Αυτή η αντιστοίχιση επιτρέπει το ping σε κάθε κοντέινερ με το όνομα σε αντίθεση με τη διεύθυνση IP. Παράδειγμα που ξεκινά ένα κοντέινερ Docker με προσαρμοσμένο όνομα, όνομα κεντρικού υπολογιστή και διακομιστή DNS:

  ```
  $ docker run --name test-container -it \
  --hostname=test-con.example.com \
  --dns=8.8.8.8 \
  ubuntu /bin/bash
  ```

  Σε αυτό το παράδειγμα, οι διεργασίες που εκτελούνται εντός του δοκιμαστικού κοντέινερ, όταν έρχονται αντιμέτωπες με ένα όνομα κεντρικού υπολογιστή που δεν βρίσκεται στο /etc /hosts, θα συνδεθούν στη διεύθυνση 8.8.8.8 στη θύρα 53 αναμένοντας μια υπηρεσία DNS.

- Απευθείας σύνδεση container

  Είναι δυνατή η απευθείας σύνδεση ενός κοντέινερ με ένα άλλο χρησιμοποιώντας την επιλογή - -link κατά την εκκίνηση ενός κοντέινερ. Αυτό επιτρέπει στα containers να ανακαλύπτουν το ένα το άλλο και να μεταφέρουν με ασφάλεια πληροφορίες.

Ωστόσο, το Docker έχει καταργήσει αυτήν τη δυνατότητα και συνιστά τη δημιουργία δικτύων που καθορίζονται από τον χρήστη. Έστω ότι διαθέτουμε ένα container mydb που εκτελεί μια υπηρεσία βάσης δεδομένων. Στη συνέχεια, μπορούμε να δημιουργήσουμε ένα application container με όνομα myweb και να το συνδέσουμε απευθείας με το mydb:

```
$ docker run --name myweb --link mydb:mydb -d -P myapp python app.py
```

3.9 Πώς επιτυγχάνεται η επικοινωνία των container με τον έξω κόσμο

Υπάρχουν διάφοροι τρόποι με τους οποίους τα Docker μπορούν να επικοινωνούν με τον έξω κόσμο, όπως περιγράφεται λεπτομερώς παρακάτω.

- Κοινοποιώντας τις θύρες τους (Ports) και προωθώντας τη κίνηση

Στις περισσότερες περιπτώσεις, τα δίκτυα Docker χρησιμοποιούν υποδίκτυα χωρίς πρόσβαση από τον έξω κόσμο. Για να επιτρέψουμε στα αιτήματα από το Διαδίκτυο να φτάσουν στο container, θα πρέπει να αντιστοιχίσουμε τις θύρες container σε θύρες του κεντρικού υπολογιστή του container. Για παράδειγμα, ένα αίτημα για όνομα κεντρικού υπολογιστή: 8000 θα προωθηθεί σε οποιαδήποτε υπηρεσία εκτελείται εντός του κοντέινερ στη θύρα 80, εάν είχε προηγουμένως οριστεί αντιστοίχιση από τη θύρα κεντρικού υπολογιστή 8000 στη θύρα κοντέινερ 80.

- Containers συνδεδεμένα σε πολλαπλά δίκτυα.

Οι λεπτομερείς πολιτικές δικτύου για συνδεσιμότητα και απομόνωση μπορούν να επιτευχθούν με τη σύνδεση container σε πολλά δίκτυα. Από προεπιλογή, κάθε container θα συνδέεται σε ένα μόνο δίκτυο. Μπορούν να συνδεθούν περισσότερα δίκτυα σε ένα κοντέινερ δημιουργώντας πρώτα το δίκτυο με την εντολή `docker create` (αντί του `docker run`) και στη συνέχεια εκτελώντας την εντολή `docker network connect`

Για παράδειγμα:

```
$ docker network create net1
```
δημιουργία bridge δικτύου με όνομα net1

```
$ docker network create net2
```
δημιουργία bridge δικτύου με όνομα net1

```
$ docker create -it --net net1 --name cont1 busybox sh
```

δημιουργία container με όνομα cont1 attached στο δίκτυο net1

```
$ docker network connect net2 cont1
```
attaches container cont1 στο δίκτυο net2

Με αυτόν το τρόπο το container είναι τώρα συνδεδεμένο σε δύο διαφορετικά δίκτυα ταυτόχρονα.

3.10 Docker Registry – Αποθήκευση και διανομή εικόνων Docker

Ένα Docker Registry είναι ένα σύστημα αποθήκευσης και διανομής εικόνων Docker (Docker Images). Η ίδια εικόνα μπορεί να έχει πολλές διαφορετικές εκδόσεις, που προσδιορίζονται από τις ετικέτες τους. Ένα Docker Registry οργανώνεται σε αποθήκες Docker, όπου ένα αποθετήριο περιέχει όλες τις εκδόσεις μιας συγκεκριμένης εικόνας. Το μητρώο επιτρέπει στους χρήστες του Docker να τραβούν εικόνες τοπικά, καθώς και να προωθούν νέες εικόνες στο μητρώο (παρέχονται επαρκή δικαιώματα πρόσβασης, όταν υπάρχουν). Από προεπιλογή, το Docker Engine αλληλοεπιδρά με το DockerHub, το δημόσιο δείγμα μητρώου του Docker. Ωστόσο, είναι δυνατό να εκτελέσετε εκ των προτέρων το μητρώο/διανομή Docker ανοιχτού κώδικα, καθώς και μια εμπορικά υποστηριζόμενη έκδοση που ονομάζεται Docker Trusted Registry. Υπάρχουν και άλλα δημόσια registries διαθέσιμα στο διαδίκτυο. Για να τραβήξουμε μια εικόνα από ένα μητρώο εσωτερικής εγκατάστασης, μπορούμε να εκτελέσουμε μια εντολή παρόμοια με:

```
docker pull my-registry:9000/foo/bar:2.1
```

όπου τραβάτε την έκδοση της εικόνας `foo/bar` με ετικέτα `2.1` από το μητρώο μας επί της εγκατάστασης που βρίσκεται στον τομέα `my-registry`, θύρα `9000`.

Εάν χρησιμοποιήσουμε το DockerHub αντ 'αυτού και το 2.1 ήταν επίσης η τελευταία έκδοση, μπορούμε να εκτελέσουμε αυτήν την εντολή για να τραβήξουμε την ίδια εικόνα τοπικά:

```
docker pull foo/bar
```

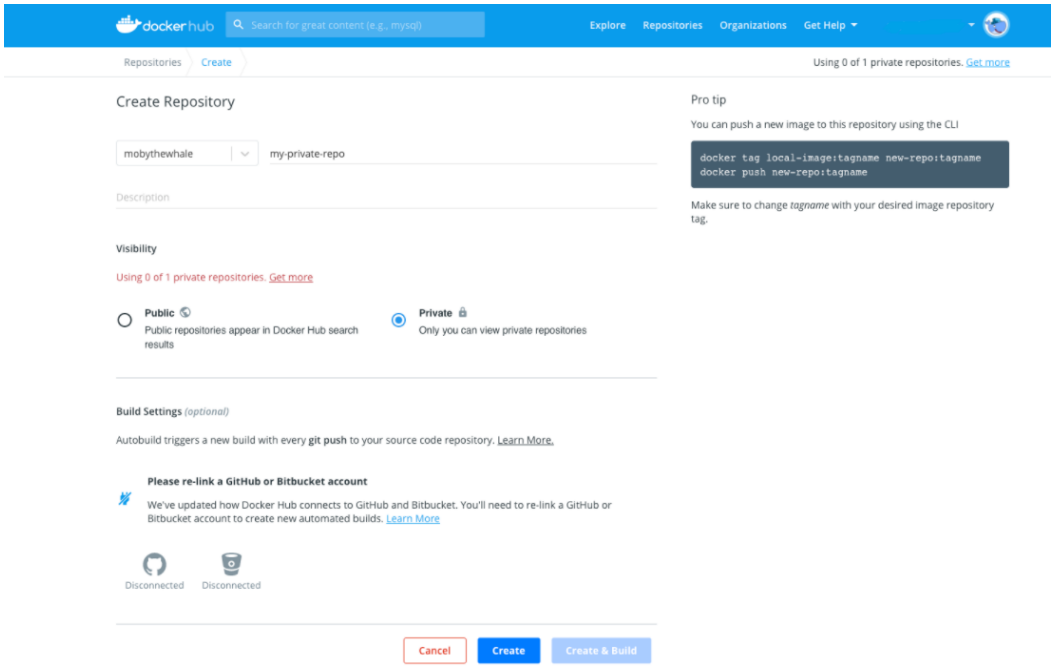
3.11 DockerHub - Αποθετήριο εικόνων Container

Το Docker Hub είναι μια υπηρεσία που παρέχεται από το Docker για εύρεση και κοινή χρήση εικόνων κοντέινερ. Είναι το μεγαλύτερο αποθετήριο εικόνων κοντέινερ στον κόσμο με μια σειρά πηγών περιεχομένου, συμπεριλαμβανομένων προγραμματιστών κοινότητας κοντέινερ, έργων ανοιχτού κώδικα και ανεξάρτητων προμηθευτών λογισμικού (ISV) που δημιουργούν και διανέμουν τον κώδικά τους σε κοντέινερ. Το Docker Hub παρέχει τα ακόλουθα κύρια χαρακτηριστικά:

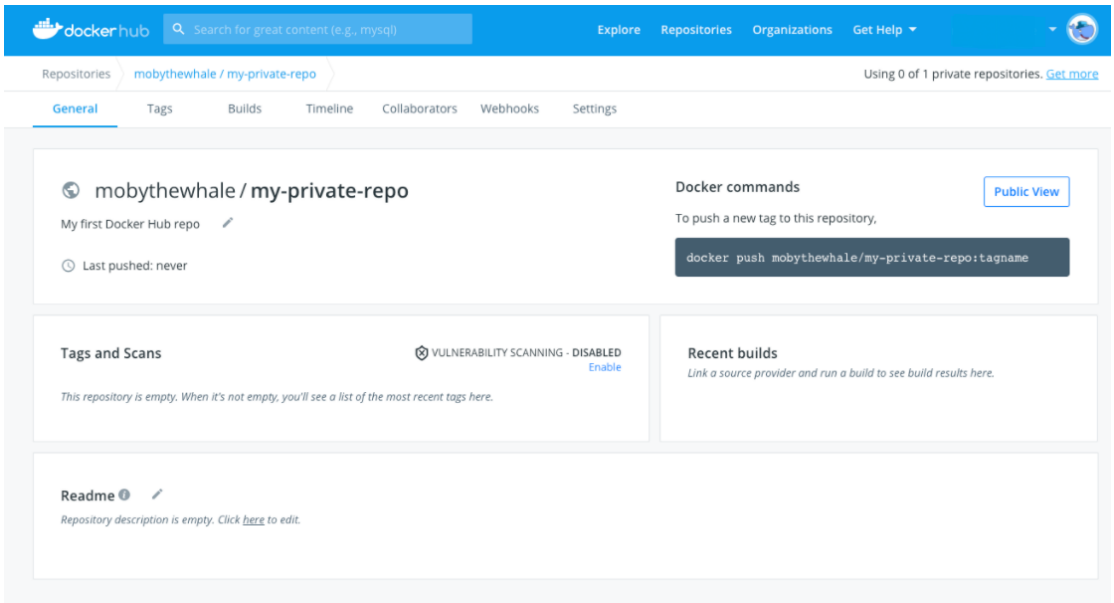
- Repositories: Ανεβάστε ή/και τραβήξτε εικόνες κοντέινερ.
- Teams & Organizations: Διαχειριστείτε την πρόσβαση σε ιδιωτικά αποθετήρια εικόνων κοντέινερ.
- Docker Official Images: Τραβήξτε και χρησιμοποιήστε εικόνες κοντέινερ υψηλής ποιότητας που παρέχονται από το Docker.
- Docker Verified Publisher Images: Τραβήξτε και χρησιμοποιήστε εικόνες κοντέινερ υψηλής ποιότητας που παρέχονται από εξωτερικούς προμηθευτές.
- Builds: Δημιουργήστε αυτόματα εικόνες κοντέινερ από το GitHub και το Bitbucket και σπρώξτε τις στο Docker Hub.
- Webhooks: Ενεργοποιήστε ενέργειες μετά από μια επιτυχημένη ώθηση σε ένα αποθετήριο για την ενσωμάτωση του Docker Hub με άλλες υπηρεσίες.

Η παρακάτω ενότητα περιέχει αναλυτικές οδηγίες σχετικά με το πώς να ξεκινήσετε εύκολα με το Docker Hub:

1. Εγγραφείτε στο Docker.
   Το Docker ID σας δίνει πρόσβαση στα αποθετήρια του Docker Hub και σας επιτρέπει να εξερευνήσετε εικόνες που είναι διαθέσιμες από την κοινότητα και τους επαληθευμένους εκδότες. Θα χρειαστείτε επίσης ένα αναγνωριστικό Docker για να μοιραστείτε εικόνες στο Docker Hub.
2. Δημιουργήστε το πρώτο σας repository.
   - Συνδεθείτε στο Docker Hub.
   - Κάντε κλικ στην επιλογή " Create Repository" στη σελίδα υποδοχής του Docker Hub:
   - Ονομάστε το <your-username>/my-private-repo.
   - Ορίστε την ορατότητα (Visibility) σε Ιδιωτικό.



- Click Create



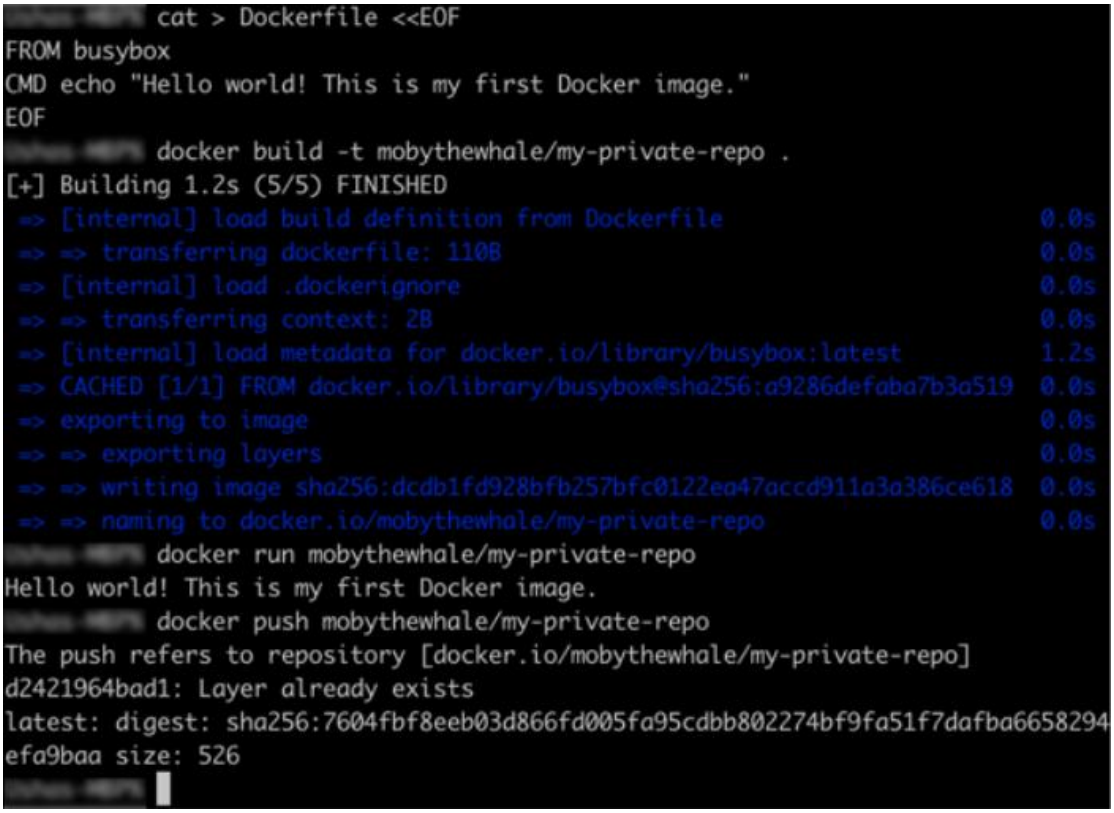
3. Κατεβάστε και εγκαταστήστε το Docker Desktop.
   Θα χρειαστεί να κατεβάσουμε το Docker Desktop για να δημιουργήσουμε και να προωθήσουμε μια εικόνα κοντέινερ στο Docker Hub. Κατεβάστε και εγκαταστήστε το Docker Desktop. Εάν χρησιμοποιείτε Linux, κάντε λήψη του Docker Engine. Συνδεθείτε στην εφαρμογή Docker Desktop χρησιμοποιώντας το αναγνωριστικό Docker ID που δημιουργήσατε.
4. Δημιουργήστε και "σπρώξτε" μια εικόνα κοντέινερ στο Docker Hub από τον υπολογιστή σας.
   Ξεκινήστε δημιουργώντας ένα Dockerfile για να καθορίσετε την εφαρμογή σας όπως φαίνεται παρακάτω:

```
# syntax=docker/dockerfile:1
FROM busybox
CMD echo "Hello world! This is my first Docker image."
```

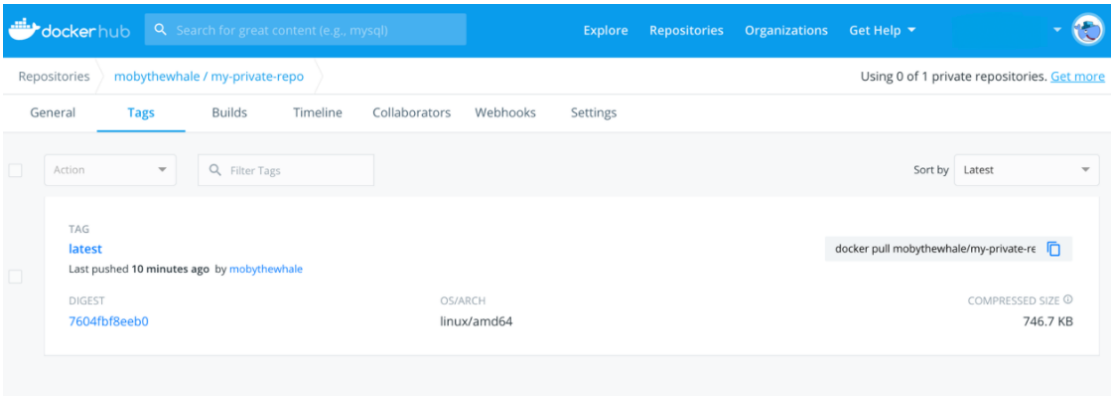
- Εκτελούμε `docker build -t <your_username>/my-private-repo .` για να χτίσουμε την Docker εικόνα μας.

- Εκτελούμε `docker run <your_username>/my-private-repo` για να ελέγξουμε την εικόνα μας τοπικά.

- Run `$docker push <your_username>/my-private-repo` για να ανεβάσουμε την εικόνα μας στο DockerHub. Θα πρέπει να έχουμε το εξής output:



5. Το Repository σας στο Docker Hub θα πρέπει τώρα να εμφανίζει μια νέα πιο πρόσφατη ετικέτα στην περιοχή Tags.


docker hub
Search for great content (e.g., mysql)
Explore
Repositories
Organizations
Get Help
Repositories
mobythewhale / my-private-repo
Using 0 of 1 private repositories. Get more
General
Tags
Builds
Timeline
Collaborators
Webhooks
Settings
Action
Filter Tags
Sort by
Latest
TAG
latest
Last pushed 10 minutes ago by mobythewhale
docker pull mobythewhale/my-private-re
DIGEST
7604fbf8eeb0
OS/ARCH
linux/amd64
COMPRESSED SIZE
746.7 KB

4. Docker Swarm – Λειτουργία σμήνους

Η λειτουργία σμήνους μας επιτρέπει να διαχειριζόμαστε ένα σύμπλεγμα Docker Engines, εγγενώς στην πλατφόρμα Docker. Μπορούμε να χρησιμοποιούμε το Docker CLI για να δημιουργήσουμε ένα σμήνος, να αναπτύξουμε υπηρεσίες εφαρμογών σε ένα σμήνος και να διαχειριστούμε τη συμπεριφορά σμήνους. Η λειτουργία Swarm παρέχει στους προγραμματιστές και διαχειριστές πληροφορικής τις εξής δυνατότητες:

- Αποκεντρωμένος σχεδιασμός (Decentralized design): Αντί να χειρίζεται τη διαφοροποίηση μεταξύ των ρόλων των κόμβων κατά το χρόνο ανάπτυξης, το Docker Engine χειρίζεται οποιαδήποτε εξειδίκευση κατά τη διάρκεια του χρόνου εκτέλεσης. Μας δίνετε η δυνατότητα να αναπτύξουμε και τα δύο είδη κόμβων, διαχειριστές και εργαζόμενους, χρησιμοποιώντας το Docker Engine. Αυτό σημαίνει ότι μπορούμε να δημιουργήσουμε ένα ολόκληρο σμήνος από μια εικόνα δίσκου.
- Διαχείριση συμπλεγμάτων (cluster management) ενσωματωμένη στο Docker Engine: Χρησιμοποιούμε το Docker Engine CLI για να δημιουργήσουμε ένα σμήνος Docker Engines όπου μπορούμε να αναπτύξουμε υπηρεσίες εφαρμογών. Δεν χρειαζόμαστε επιπλέον λογισμικό ενορχήστρωσης για να δημιουργήσουμε ή να διαχειριστούμε ένα σμήνος.
- Μοντέλο δηλωτικής υπηρεσίας (Declarative service model): Το Docker Engine χρησιμοποιεί μια δηλωτική προσέγγιση για να μας επιτρέψει να ορίσουμε την επιθυμητή κατάσταση των διαφόρων υπηρεσιών στη στοίβα εφαρμογών μας.
- Κλιμάκωση (Scaling): Για κάθε υπηρεσία, μπορούμε να δηλώσουμε τον αριθμό των εργασιών που θέλουμε να εκτελέσουμε. Όταν αυξάνετε ή μειώνετε αυτός ο αριθμός, ο διαχειριστής σμήνους προσαρμόζεται αυτόματα προσθέτοντας ή αφαιρώντας εργασίες για να διατηρήσει την επιθυμητή κατάσταση.
- Επιθυμητή «συμφιλίωση» κατάστασης: Ο κόμβος διαχειριστή του σμήνους παρακολουθεί συνεχώς την κατάσταση συμπλέγματος και συμβιβάζει τυχόν διαφορές μεταξύ της πραγματικής κατάστασης και της επιθυμητής κατάστασης που ορίσαμε. Για παράδειγμα, εάν ρυθμίσουμε μια υπηρεσία για την εκτέλεση 10 αντιγράφων ενός κοντέινερ και μια μηχανή εργαζομένων που φιλοξενεί δύο από αυτά τα αντίγραφα καταρρεύσει, ο διαχειριστής δημιουργεί δύο νέα αντίγραφα για να αντικαταστήσει τα αντίγραφα που συνετρίβησαν. Ο διαχειριστής σμήνους αναθέτει τα νέα αντίγραφα σε εργαζόμενους που είναι σε λειτουργία και διαθέσιμα.
- Δίκτυο πολλαπλών κεντρικών υπολογιστών (Multi-host networking): Μπορούμε να καθορίσουμε ένα δίκτυο επικάλυψης για τις υπηρεσίες μας. Ο διαχειριστής σμήνους εκχωρεί αυτόματα διευθύνσεις στα κοντέινερ στο δίκτυο επικάλυψης (overlay network) όταν προετοιμάζει ή ενημερώνει την εφαρμογή.
- Ανακάλυψη υπηρεσίας (Service discovery): Οι κόμβοι διαχειριστή σμήνους εκχωρούν σε κάθε υπηρεσία στο σμήνος ένα μοναδικό όνομα DNS και ισορροπίες φορτίου που εκτελούν κοντέινερ. Μπορούμε να αναζητήσουμε κάθε κοντέινερ που τρέχει στο σμήνος μέσω ενός διακομιστή DNS που είναι ενσωματωμένος στο σμήνος.
- Εξισορρόπηση φορτίου (Load Balancing): Μπορούμε να εκθέσουμε τις θύρες για υπηρεσίες σε εξωτερικό εξισορροπητή φορτίου. Εσωτερικά, το σμήνος μας επιτρέπει να καθορίσουμε τον τρόπο διανομής των υπηρεσιών μεταξύ των κόμβων.

- Ασφάλεια από προεπιλογή (Secure by default): Κάθε κόμβος στο σμήνος επιβάλλει αμοιβαίο έλεγχο ταυτότητας και κρυπτογράφηση TLS για την εξασφάλιση επικοινωνιών μεταξύ αυτού και όλων των άλλων κόμβων. Έχουμε την επιλογή να χρησιμοποιήσουμε αυτο-υπογεγραμμένα πιστοποιητικά διαδρομής ή πιστοποιητικά από μια προσαρμοσμένη διαδρομή CA.
- Κυλιόμενες ενημερώσεις: Μπορούμε να εφαρμόσουμε σταδιακά ενημερώσεις υπηρεσιών στους κόμβους. Ο διαχειριστής σμήνους μας επιτρέπει να ελέγχουμε την καθυστέρηση μεταξύ της ανάπτυξης της υπηρεσίας σε διαφορετικά σύνολα κόμβων. Εάν κάτι πάει στραβά, μπορούμε να επιστρέψουμε σε μια προηγούμενη έκδοση της υπηρεσίας.

4.1 Ενορχήστρωση

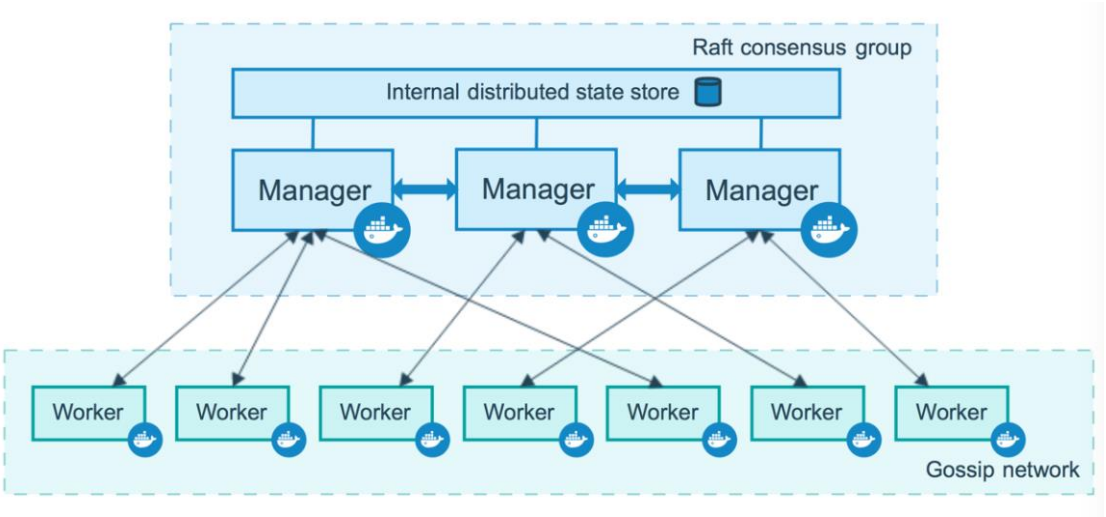
Η φορητότητα και η αναπαραγωγιμότητα μιας containerized διαδικασίας (process) μας παρέχει την ευκαιρία να μετακινούμε και να κλιμακώνουμε τις εφαρμογές μας είτε σε containers, είτε σε cloud είτε σε κέντρα δεδομένων (datacenters). Τα container εγγυώνται αποτελεσματικά ότι αυτές οι εφαρμογές εκτελούνται με τον ίδιο τρόπο οπουδήποτε, επιτρέποντάς μας να εκμεταλλευτούμε γρήγορα και εύκολα όλα αυτά τα περιβάλλοντα. Επιπλέον, καθώς αυξάνουμε τις εφαρμογές μας, χρειαζόμαστε κάποια εργαλεία για να αυτοματοποιήσουμε τη συντήρηση αυτών των εφαρμογών, να ενεργοποιήσουμε αυτόματα την αντικατάσταση των αποτυχημένων κοντέινερ και να διαχειριστούμε την κυκλοφορία των ενημερώσεων και των επαναδιαμορφώσεων αυτών των κοντέινερ κατά τη διάρκεια του κύκλου ζωής τους. Τα εργαλεία για τη διαχείριση, την κλιμάκωση και τη διατήρηση εφαρμογών σε κοντέινερ ονομάζονται ενορχηστρωτές, και τα πιο συνηθισμένα παραδείγματα αυτών είναι οι Kubernetes, το Docker Swarm και το Apache Mesos. Η ανάπτυξη του περιβάλλοντος αυτών ενορχηστρωτών παρέχεται από το Docker Desktop.

Σε αυτό το σημείο, θα κάνουμε μια αναφορά στο Kubernetes. Το Kubernetes είναι ένα εργαλείο ενορχήστρωσης κοντέινερ ανοιχτού κώδικα που αναπτύχθηκε και σχεδιάστηκε αρχικά από μηχανικούς της Google. Η Google δώρισε το έργο Kubernetes στο νεοσύστατο Cloud Native Computing Foundation το 2015.

Η ενορχήστρωση Kubernetes μας επιτρέπει να δημιουργήσουμε υπηρεσίες εφαρμογών που εκτείνονται σε πολλαπλά κοντέινερ, να προγραμματίζουμε κοντέινερ σε ένα σύμπλεγμα, να κλιμακώνουμε αυτά τα κοντέινερ και να διαχειριζόμαστε την υγεία τους με την πάροδο του χρόνου. Το Kubernetes εξαλείφει πολλές από τις χειροκίνητες διαδικασίες που εμπλέκονται στην ανάπτυξη και την κλιμάκωση εφαρμογών που περιέχονται σε κοντέινερ. Μπορούμε να ομαδοποιήσουμε ομάδες από hosts (cluster together groups of hosts), είτε πρόκειται περί φυσικών είτε εικονικών μηχανών, που εκτελούν κοντέινερ Linux και το Kubernetes μας δίνει την πλατφόρμα για να διαχειριζόμαστε εύκολα και αποτελεσματικά αυτά τα συμπλέγματα. Αυτές οι ομάδες μπορούν να εκτείνονται είτε σε κεντρικούς υπολογιστές, είτε σε δημόσια, ιδιωτικά ή υβριδικά clouds. Για το λόγο αυτό, το Kubernetes είναι μια ιδανική πλατφόρμα για τη φιλοξενία εφαρμογών που προέρχονται από το cloud και απαιτούν γρήγορη κλιμάκωση (rapid scaling). Επιπρόσθετα, το Kubernetes βοηθά στη φορητότητα του φόρτου εργασίας και την εξισορρόπηση φορτίου, επιτρέποντάς μας να μετακινούμε εφαρμογές χωρίς να τις επανασχεδιάζουμε.

4.2 Ρόλοι- Λειτουργία των κόμβων

Το Docker Engine 1.12 εισάγει τη λειτουργία σμήνους που μας επιτρέπει να δημιουργήσουμε ένα σύμπλεγμα από έναν ή περισσότερους κινητήρες Docker που ονομάζεται σμήνος. Ένα σμήνος αποτελείται από έναν ή περισσότερους κόμβους: φυσικές ή εικονικές μηχανές που λειτουργούν με το Docker Engine 1.12 ή νεότερο. Υπάρχουν δύο τύποι κόμβων: διαχειριστές (managers) και εργαζόμενοι (workers).



4.2.1 Κόμβοι Manager

Οι κόμβοι διαχείρισης χειρίζονται εργασίες διαχείρισης συμπλέγματος:

- διατήρηση της κατάστασης συμπλέγματος
- υπηρεσίες προγραμματισμού
- εξυπηρέτηση καταληκτικών σημείων HTTP API

Χρησιμοποιώντας μια εφαρμογή Raft, οι διαχειριστές διατηρούν μια συνεπή εσωτερική κατάσταση ολόκληρου του σμήνους και όλων των υπηρεσιών που εκτελούνται σε αυτό. Για σκοπούς δοκιμής, μπορούμε να εκτελέσουμε ένα σμήνος με έναν μόνο διαχειριστή. Εάν ο διαχειριστής σε ένα σμήνος αποτύχει, οι υπηρεσίες μας συνεχίζουν να εκτελούνται, αλλά πρέπει να δημιουργήσουμε ένα νέο σύμπλεγμα για να τις ανακτήσουμε. Για να επωφεληθούμε από τις δυνατότητες ανοχής σε σφάλματα της λειτουργίας σμήνους, το Docker συνιστά να εφαρμόζετε έναν αριθμό κόμβων σύμφωνα με τις απαιτήσεις υψηλής διαθεσιμότητας του οργανισμού σας. Παραδείγματος χάρη, όταν διαθέτουμε πολλούς διαχειριστές, μπορούμε να ανακάμψουμε από την αποτυχία ενός κόμβου διαχειριστή χωρίς διακοπή λειτουργίας. Χαρακτηριστικά:

- Ένα σμήνος τριών διαχειριστών ανέχεται τη μέγιστη απώλεια ενός διαχειριστή.
- Ένα σμήνος πέντε διαχειριστών ανέχεται τη μέγιστη ταυτόχρονη απώλεια δύο κόμβων διαχειριστή.
- Ένα σύμπλεγμα N διαχειριστών ανέχεται την απώλεια το πολύ (N-1)/2 διαχειριστών.

Από το Docker συνιστάται έως και επτά κόμβους διαχειριστή για ένα σμήνος.

4.2.2 Κόμβοι workers

Οι κόμβοι εργαζομένων είναι επίσης περιπτώσεις του Docker Engine των οποίων ο μοναδικός σκοπός είναι η εκτέλεση κοντέινερ. Οι κόμβοι εργαζομένων δεν συμμετέχουν στην κατανεμημένη κατάσταση Raft, δεν λαμβάνουν αποφάσεις προγραμματισμού ή δεν εξυπηρετούν το HTM API σε λειτουργία σμήνους. Μπορείτε να δημιουργήσετε ένα σμήνος ενός κόμβου διαχειριστή, αλλά δεν μπορείτε να έχετε έναν κόμβο εργαζομένου χωρίς τουλάχιστον έναν κόμβο διαχειριστή. Από προεπιλογή, όλοι οι διαχειριστές είναι επίσης εργαζόμενοι. Σε ένα σύμπλεγμα κόμβων διαχειριστή, μπορείτε να εκτελέσετε εντολές όπως `$docker service create` και ο προγραμματιστής τοποθετεί όλες τις εργασίες στην τοπική μηχανή. Για να αποτρέψετε τον προγραμματιστή να τοποθετήσει εργασίες σε έναν κόμβο διαχειριστή σε ένα σμήνος πολλαπλών κόμβων, ορίστε τη διαθεσιμότητα για τον κόμβο διαχειριστή στο `Drain`. Ο χρονοπρογραμματιστής διακόπτει τις εργασίες σε κόμβους στη λειτουργία `Drain` και προγραμματίζει τις εργασίες σε έναν `Active` κόμβο.

4.2.3 Αλλαγή στους ρόλους των κόμβων

Μπορούμε να αναβαθμίσουμε έναν κόμβο εργαζομένου ως διαχειριστή εκτελώντας `$docker node promote`. Για παράδειγμα, μπορεί να θέλουμε να αναβαθμίσουμε έναν κόμβο εργαζόμενου όταν μεταφέρουμε έναν κόμβο διαχειριστή εκτός σύνδεσης για συντήρηση. Επίσης δίνεται η δυνατότητα να υποβαθμίσουμε έναν κόμβο εκτελώντας `$docker node demote`

4.3 Δημιουργία σμήνους

Το Docker Engine λειτουργεί από προεπιλογή με απενεργοποιημένη τη λειτουργία σμήνους. Για να εκτελέσουμε το Docker σε κατάσταση σμήνους, μπορούμε είτε να δημιουργήσουμε ένα νέο σμήνος είτε να βάλουμε το κοντέινερ σε ένα υπάρχον σμήνος.

- Για να δημιουργήσετε ένα σμήνος-εκτελούμε την εντολή `docker swarm init`, η οποία δημιουργεί ένα σμήνος ενός κόμβου στην τρέχουσα μηχανή Docker. Ο τρέχων κόμβος γίνεται ο κόμβος διαχειριστή του νέου σμήνους.
- Για να συμμετάσχουμε σε ένα σμήνος - η έξοδος για την εντολή `docker swarm init` μας λέει ποια εντολή πρέπει να εκτελέσουμε σε άλλα Docker container για να τους επιτρέψουμε να ενταχθούν στο σμήνος μας ως κόμβοι εργαζομένων (workers), συμπεριλαμβανομένου ενός "διακριτικού σύνδεσης". Για παράδειγμα, για να προσθέσετε έναν εργαζόμενο σε αυτό το σμήνος, εκτελούμε την ακόλουθη εντολή:
  ```
  docker swarm join \
  --token                                                SWMTKN-1-
  49nj1cmql0jkz5s954yi3oex3nedyz0fb0xx14ie39trti4wxv-
  8vxv8rssmk743ojnwacrr2e7c \
  192.168.99.100:2377
  ```

Υπάρχει ένα διαφορετικό διακριτικό σύνδεσης για τους κόμβους εργαζομένων και τους κόμβους διαχειριστή. Το διακριτικό χρησιμοποιείται μόνο τη στιγμή που ένα δοχείο ενώνεται με το σμήνος. Τα διακριτικά διαχειριστή πρέπει να προστατεύονται έντονα, επειδή οποιαδήποτε πρόσβαση στο διακριτικό διαχειριστή παρέχει έλεγχο σε ολόκληρο το σμήνος. Μπορούμε να εκτελέσουμε `swarm join-token --rotate` ανά πάσα στιγμή για να ακυρώσουμε το παλαιότερο διακριτικό και να δημιουργήσουμε ένα νέο, για λόγους

ασφαλείας. Οι κόμβοι σμήνους μπορούν να έχουν πρόσβαση στο SwarmKit API (που παρέχει λειτουργίες όπως η ανακάλυψη κόμβων και ο προγραμματισμός εργασιών) και η επικάλυψη δικτύων, χρησιμοποιώντας μια "διεύθυνση διαφήμισης" που καθορίζετε για τον κόμβο διαχειριστή. Εάν δεν καθορίσετε μια διεύθυνση και υπάρχει ένα μόνο IP για το σύστημα, το Docker ακούει από προεπιλογή στη θύρα 2377. Το SwarmKit είναι μια εργαλειοθήκη για την ενορχήστρωση κατανεμημένων συστημάτων, συμπεριλαμβανομένης της ανακάλυψης κόμβων και του προγραμματισμού εργασιών.

Πριν ξεκινήσουμε με τη λειτουργία swarm θα πρέπει να διασφαλίσουμε τα εξής:

- τρεις οικοδεσπότες Linux που μπορούν να επικοινωνούν μέσω δικτύου, με εγκατεστημένο το Docker. Αυτοί μπορεί να είναι φυσικά μηχανήματα, εικονικές μηχανές, παρουσίες Amazon EC2 ή να φιλοξενούνται με κάποιον άλλο τρόπο. Μπορούμε ακόμη να χρησιμοποιήσουμε το Docker Machine από έναν κεντρικό υπολογιστή Linux, Mac ή Windows.
- τη διεύθυνση IP του μηχανήματος διαχείρισης. Η διεύθυνση IP πρέπει να εκχωρηθεί σε μια διεπαφή δικτύου διαθέσιμη στο λειτουργικό σύστημα κεντρικού υπολογιστή. Όλοι οι κόμβοι στο σμήνος πρέπει να συνδεθούν με τον διαχειριστή στη διεύθυνση IP. Επειδή άλλοι κόμβοι επικοινωνούν με τον κόμβο διαχειριστή στη διεύθυνση IP του, θα πρέπει να χρησιμοποιήσετε μια σταθερή διεύθυνση IP. Μπορούμε να εκτελέσουμε `ifconfig` σε Linux ή macOS για να δούμε μια λίστα με τις διαθέσιμες διεπαφές δικτύου. Εάν χρησιμοποιείτε Docker Machine, μπορείτε να λάβετε το IP διαχειριστή είτε με `docker-machine ls` είτε με `docker-machine ip <MACHINE-NAME>` -για παράδειγμα, `docker-machine ip manager1.`
  Στη συγκεκριμένη περίπτωση θα χρησιμοποιήσουμε manager1: 192.168.99.100.
- ανοιχτές θύρες μεταξύ των κεντρικών υπολογιστών. Οι ακόλουθες θύρες πρέπει να είναι διαθέσιμες. Σε ορισμένα συστήματα, αυτές οι θύρες είναι ανοιχτές από προεπιλογή.
    - Θύρα TCP 2377 για επικοινωνίες διαχείρισης συμπλέγματος
    - Θύρα TCP και UDP 7946 για επικοινωνία μεταξύ κόμβων
    - Θύρα UDP 4789 για επικάλυψη κυκλοφορίας δικτύου

  Εάν σκοπεύετε να δημιουργήσετε ένα δίκτυο επικάλυψης με κρυπτογράφηση (-opt κρυπτογραφημένο), πρέπει επίσης να διασφαλίσετε ότι επιτρέπεται η κυκλοφορία του πρωτοκόλλου IP 50 (ESP).

Αφού ολοκληρώσουμε τα παραπάνω βήματα, είμαστε έτοιμοι να δημιουργήσουμε ένα σμήνος. Βεβαιωθείτε ότι ο Docker Engine daemon έχει ξεκινήσει στις κεντρικές μηχανές.

Ανοίξτε ένα τερματικό και ssh στο μηχάνημα όπου θέλετε να εκτελέσετε τον κόμβο διαχειριστή. Χρησιμοποιούμε σε αυτό το παράδειγμα ένα μηχάνημα που ονομάζεται manager1. Εάν χρησιμοποιείτε το Docker Machine, μπορείτε να συνδεθείτε σε αυτό μέσω SSH χρησιμοποιώντας την ακόλουθη εντολή:

```
$ docker-machine ssh manager1
```

Για να φτιάξουμε ένα νέο Swarm στο manager1 μηχάνημα εκτελούμε:

```
$ docker swarm init --advertise-addr 192.168.99.100
Swarm initialized: current node (dxn1zf6l61qsb1josjja83ngz) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join \
    --token SWMTKN-1-49nj1cmql0jkz5s954yi3oex3nedyz0fb0xx14ie39trti4wxv-8vxv8rssmk743ojnwacrr2e7c \
    192.168.99.100:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

Το flag `--advertise-addr` ρυθμίζει τον κόμβο διαχειριστή για να δημοσιεύσει τη διεύθυνσή του ως 192.168.99.100. Οι άλλοι κόμβοι στο σμήνος πρέπει να έχουν πρόσβαση στον διαχειριστή στη διεύθυνση IP. Η έξοδος περιλαμβάνει τις εντολές για να ενώσετε νέους κόμβους στο σμήνος. Οι κόμβοι θα ενταχθούν ως διαχειριστές ή εργαζόμενοι ανάλογα με την τιμή για τη σημαία -token.

Εκτελώντας `$docker info` μας δίνεται η δυνατότητα να ελέγξουμε την τρέχουσα κατάσταση του σμήνους:

```
$ docker info

Containers: 2
Running: 0
Paused: 0
Stopped: 2
  ...snip...
Swarm: active
  NodeID: dxn1zf6l61qsb1josjja83ngz
  Is Manager: true
  Managers: 1
  Nodes: 1
  ...snip...
```

Εκτελώντας `$docker node ls` μας εμφανίζονται πληροφορίες σχετικά με τους κόμβους.

```
$ docker node ls

ID                           HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
dxn1zf6l61qsb1josjja83ngz *  manager1  Ready   Active        Leader
```

4.4 Προσθήκη κόμβων στο σμήνος

Μόλις δημιουργηθεί ένα σμήνος με έναν κόμβο διαχειριστή, είμαστε έτοιμοι να προσθέσουμε κόμβους εργαζομένων (workers).

Ανοίγουμε ένα τερματικό και κάνουμε ssh στο μηχάνημα όπου θέλουμε να εκτελέσουμε έναν κόμβο εργαζομένου. Θα το ονομάσουμε worker1. Εκτελούμε την εντολή που

παράγεται από την έξοδο του `$docker swarm init` από το παραπάνω βήμα στη δημιουργία σμήνους για να δημιουργήσουμε έναν κόμβο εργαζομένου που συνδέεται με το υπάρχον σμήνος:

```
$ docker swarm join \
  --token  SWMTKN-1-49nj1cmql0jkz5s954yi3oex3nedyz0fb0xx14ie39trti4wxv-8vxv8rssmk743ojnwacrr2e7c \
  192.168.99.100:2377

This node joined a swarm as a worker.
```

Εάν δεν έχουμε διαθέσιμη την εντολή, μπορούμε να εκτελέσουμε την ακόλουθη εντολή σε έναν κόμβο διαχειριστή για να ανακτήσουμε την εντολή σύνδεσης για έναν worker:

```
$ docker swarm join-token worker

To add a worker to this swarm, run the following command:

    docker swarm join \
    --token SWMTKN-1-49nj1cmql0jkz5s954yi3oex3nedyz0fb0xx14ie39trti4wxv-8vxv8rssmk743ojnwacrr2e7c \
    192.168.99.100:2377
```

Ανοίγουμε ένα τερματικό και κάνουμε ssh στο μηχάνημα όπου θέλουμε να εκτελέσουμε έναν δεύτερο κόμβο εργαζομένου. Θα το ονομάσουμε worker2. Εκτελούμε την εντολή που παράγεται από την έξοδο του `$docker swarm init` από το παραπάνω βήμα στη δημιουργία σμήνους για να δημιουργήσουμε άλλον έναν κόμβο εργαζομένου που συνδέεται με το υπάρχον σμήνος:

```
$ docker swarm join \
  --token SWMTKN-1-49nj1cmql0jkz5s954yi3oex3nedyz0fb0xx14ie39trti4wxv-8vxv8rssmk743ojnwacrr2e7c \
  192.168.99.100:2377

This node joined a swarm as a worker.
```

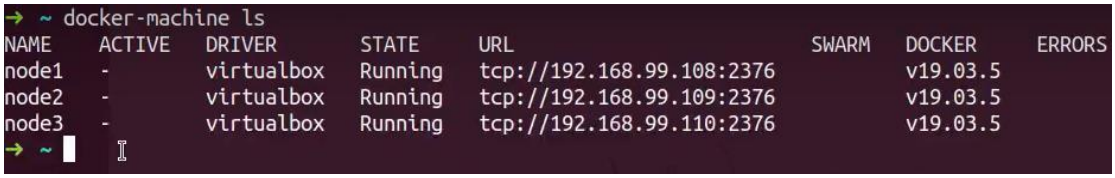
Ανοίγουμε ένα τερματικό και κάνουμε ssh στο μηχάνημα όπου εκτελείται ο κόμβος διαχειριστή και εκτελέστε την εντολή `$docker node ls` για να δείτε τους κόμβους εργαζομένων:

```
$ docker node ls
ID                            HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
03g1y59jwfg7cf99w4lt0f662     worker2   Ready   Active
9j68exjopxe7wfl6yuxml7a7j     worker1   Ready   Active
dxn1zf6l61qsb1josjja83ngz *   manager1  Ready   Active        Leader
```

Η στήλη MANAGER προσδιορίζει τους κόμβους διαχειριστή στο σμήνος. Η κενή κατάσταση σε αυτήν τη στήλη για τον worker1 και τον worker2 τους προσδιορίζει ως κόμβους εργαζομένων. Οι εντολές διαχείρισης σμήνους, όπως η εντολή `$docker node ls` λειτουργούν μόνο σε κόμβους διαχειριστή.

5. Δημιουργία μια nginx υπηρεσίας στο cluster του Docker swarm

Ξεκινάμε δημιουργώντας τρία Docker Machines. Και τα τρία είναι VirtualBox VM's



Έχοντας δημιουργήσει τα τρία Docker Machines, προχωράμε στο να τα βάλουμε στο swarm cluster, στο οποίο cluster θα πρέπει να έχουμε τουλάχιστον ένα manager node και worker nodes. Στο node που θα διαλέξουμε να ορίσουμε ως manager node θα πρέπει να αρχικοποιήσουμε το swarm cluster μας και στη συνέχεια θα προσθέσουμε τους worker nodes στο cluster. Στο παράδειγμα αυτό θα ορίσουμε το node1 ως manager και node2, node3 ως workers. Εκτελώντας `$docker-machine ssh node1` συνδεόμαστε στο node1 στον οποίο θα αρχικοποιήσουμε το cluster μας ορίζοντάς τον έτσι ως manager node του swarm μας. Εκτελώντας `$docker info` εμφανίζονται πληροφορίες σχετικά με το node. Παρατηρούμε πως το swarm είναι ανενεργό.

```
Images: 0
Server Version: 19.03.5
Storage Driver: overlay2
 Backing Filesystem: extfs
 Supports d_type: true
 Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host ipvlan macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: b34a5c8af56e510852c35414db4c1f4fa6172339
runc version: 3e425f80a8c931f88e6d94a8c831b9d5aa481657
init version: fec3683
Security Options:
 seccomp
  Profile: default
Kernel Version: 4.14.154-boot2docker
Operating System: Boot2Docker 19.03.5 (TCL 10.1)
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 989.5MiB
Name: node1
ID: PCJC:5WIQ:NTM2:V2E3:CXNX:7EAY:IP2L:GXXZ:D3H7:SDUN:JIZP:4FWT
Docker Root Dir: /mnt/sda1/var/lib/docker
Debug Mode: false
Registry: https://index.docker.io/v1/
Labels:
 provider=virtualbox
Experimental: false
Insecure Registries:
 127.0.0.0/8
Live Restore Enabled: false
Product License: Community Engine
```

Αρχικά πρέπει να βρούμε την IP διεύθυνση του node1 και στη συνέχεια εκτελώντας `$docker swarm init  - -advertise addr <IP Address>` βλέπουμε πως το swarm μας έχει αρχικοποιηθεί (swarm initialized) και πως ο node1 είναι πλέον manager.

```
docker@node1:~$ ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:f2:52:c2 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fef2:52c2/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:82:b8:d0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.99.108/24 brd 192.168.99.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe82:b8d0/64 scope link
       valid_lft forever preferred_lft forever
4: sit0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/sit 0.0.0.0 brd 0.0.0.0
6: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:0e:79:d8:d8 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
docker@node1:~$
```

```
docker@node1:~$ docker swarm init --advertise-addr 192.168.99.108
Swarm initialized: current node (jaik34u24f0jh5h36a99k9aon) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-418majl05xpka0ciwsp9z9m13okopdzfwmz29vwz0aqdoyecrg-74096wi4dxmm2j3unuc0vt20x 192.168.99.108:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

docker@node1:~$
```

Στην έξοδο της εντολής `$docker swarm init - -advertise-addr 192.168.99.108` μας δίνονται και πληροφορίες σχετικά με το πώς θα προσθέσουμε worker στο cluster μας. Μας δίνετε το TOKEN, η IP διεύθυνση και η πόρτα του manager node. Επίσης μπορούμε να εντοπίσουμε τις παραπάνω πληροφορίες εκτελώντας `$docker swarm join-token worker`

```
docker@node1:~$ docker swarm join-token worker
To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-418majl05xpka0ciwsp9z9m13okopdzfwmz29vwz0aqdoyecrg-74096wi4dxmm2j3unuc0vt20x 192.168.99.108:2377

docker@node1:~$
```

Εκτελώντας ξανά `$docker info` βλέπουμε πως το swarm πλέον έχει αρχικοποιηθεί, το ID του node μας και πως είναι manager, το ID του cluster μας και πως περιέχει συνολικά ένα node ο οποίος είναι και manager. Επίσης βλέπουμε και τη IP διεύθυνση του node.

```
 Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host ipvlan macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
Swarm: active
 NodeID: jaik34u24f0jh5h36a99k9aon
 Is Manager: true
 ClusterID: szlrgv60w3cvjbqisuji689dz
 Managers: 1
 Nodes: 1
 Default Address Pool: 10.0.0.0/8
 SubnetSize: 24
 Data Path Port: 4789
 Orchestration:
  Task History Retention Limit: 5
 Raft:
  Snapshot Interval: 10000
  Number of Old Snapshots to Retain: 0
  Heartbeat Tick: 1
  Election Tick: 10
 Dispatcher:
  Heartbeat Period: 5 seconds
 CA Configuration:
  Expiry Duration: 3 months
  Force Rotate: 0
 Autolock Managers: false
 Root Rotation In Progress: false
 Node Address: 192.168.99.108
 Manager Addresses:
  192.168.99.108:2377
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: b34a5c8af56e510852c35414db4c1f4fa6172339
runc version: 3e425f80a8c931f88e6d94a8c831b9d5aa481657
init version: fec3683
Security Options:
```

Κάνουμε logout από το node1 και συνδεόμαστε στο node2 εκτελώντας $docker-machine ssh node2 . Εκτελώντας εδώ `$docker info` θα δούμε πως το swarm είναι ανενεργό (inactive).

Εκτελούμε την εντολή που μας δόθηκε στην έξοδο παραπάνω ώστε να προστεθεί ο node2 ως worker στο swarm μας, και `$docker info` για να εμφανίσουμε πληροφορίες σχετικά με το node2.

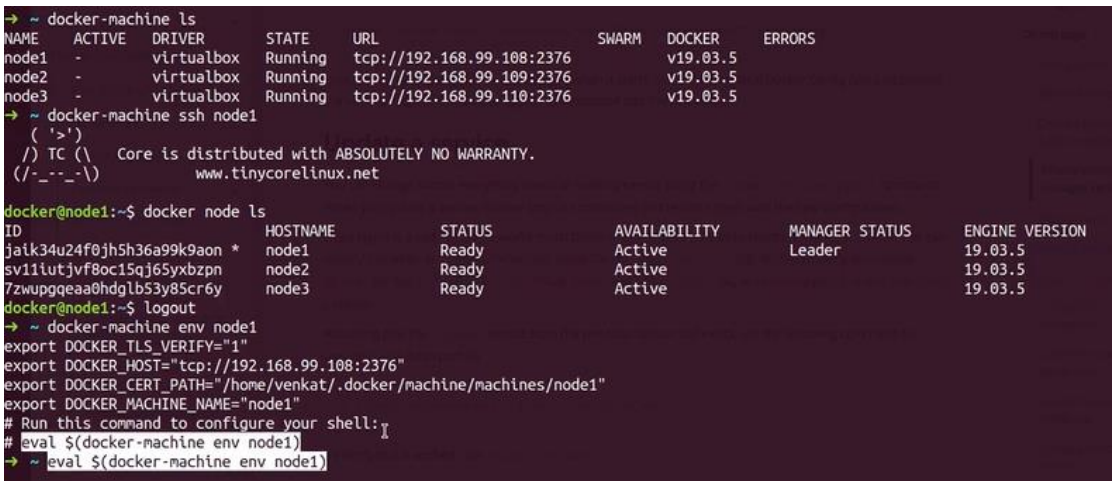
```
docker@node2:~$ docker swarm join --token SWMTKN-1-418majl05xpka0ciwsp9z9m13okopdzfwmz29vwz0aqdoyecrg-74096wi4dxmm2j3unuc0vt20x 192.168.99.108:2377
This node joined a swarm as a worker.
docker@node2:~$ docker info
```

```
 Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host ipvlan macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
Swarm: active
 NodeID: sv11iutjvf8oc15qj65yxbzpn
 Is Manager: false
 Node Address: 192.168.99.109
 Manager Addresses:
  192.168.99.108:2377
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: b34a5c8af56e510852c35414db4c1f4fa6172339
runc version: 3e425f80a8c931f88e6d94a8c831b9d5aa481657
init version: fec3683
Security Options:
 seccomp
  Profile: default
Kernel Version: 4.14.154-boot2docker
Operating System: Boot2Docker 19.03.5 (TCL 10.1)
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 989.5MiB
Name: node2
ID: 4ZAV:LCN2:4ZSS:AQOS:BKBJ:RTYY:VZSN:52W5:HGRH:7BBJ:P2XO:JGNV
Docker Root Dir: /mnt/sda1/var/lib/docker
Debug Mode: false
Registry: https://index.docker.io/v1/
Labels:
 provider=virtualbox
Experimental: false
Insecure Registries:
 127.0.0.0/8
Live Restore Enabled: false
Product License: Community Engine
```

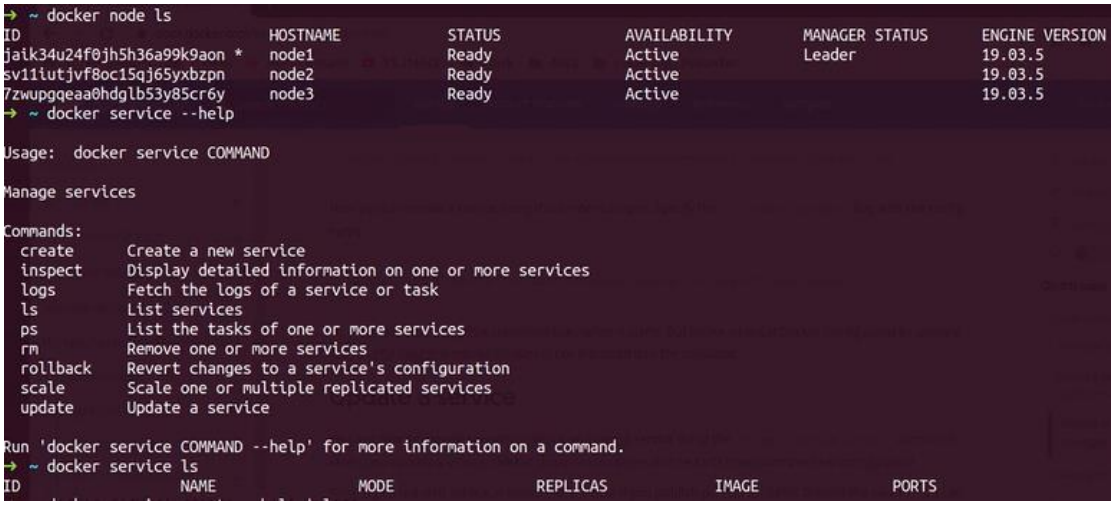
Παρατηρούμε πως το swarm έχει ενεργοποιηθεί, πως ο node2 δεν είναι manager στο swarm cluster μας, την IP address του και την IP address του manager node1.Ομοίως εκτελούμε τα παραπάνω και στο node3.

```
 Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host ipvlan macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
Swarm: active
 NodeID: ns6ac8wkmrrayylvdkpcg91p7
 Is Manager: false
 Node Address: 192.168.99.110
 Manager Addresses:
  192.168.99.108:2377
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: b34a5c8af56e510852c35414db4c1f4fa6172339
runc version: 3e425f80a8c931f88e6d94a8c831b9d5aa481657
init version: fec3683
Security Options:
 seccomp
  Profile: default
Kernel Version: 4.14.154-boot2docker
Operating System: Boot2Docker 19.03.5 (TCL 10.1)
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 989.5MiB
Name: node3
ID: OEJD:MQPC:JCZL:7PFA:CDCE:CXLW:IRJM:G3R4:3FPU:ELLR:EFBZ:CNGO
Docker Root Dir: /mnt/sda1/var/lib/docker
Debug Mode: false
Registry: https://index.docker.io/v1/
Labels:
 provider=virtualbox
Experimental: false
Insecure Registries:
 127.0.0.0/8
Live Restore Enabled: false
Product License: Community Engine
```

Συνεχίσουμε αναπτύσσοντας τα service μας στο docker cluster που δημιουργήσαμε παραπάνω. Για να εκτελέσουμε εντολές στο swarm cluster μας θα πρέπει να είμαστε συνδεδεμένοι στο manager node, καθώς δεν εκτελούνται από τους workers. Αντ' αυτού μπορούμε να χρησιμοποιήσουμε την εντολή docker-machine env node1 και στη συνέχεια την εντολή eval $(docker-machine env node1) που μας δίνεται στην έξοδο. Με αυτόν τον τρόπο οι εντολές που θα «τρέξουμε» στη συνέχεια θα πηγαίνουν στο Docker Engine που βρίσκεται στο node1. Με αυτόν τον τρόπο, δεν είμαστε συνδεδεμένοι στο node1, βρισκόμαστε στο host μας, αλλά είμαστε συνδεδεμένοι στο docker Runtime του manager node1.



Προκειμένου να αναπτύξουμε μια υπηρεσία, εκτελούμε `$docker service <COMMAND>`. Εκτελώντας `$docker service - -help` μας εμφανίζονται όλες οι δυνατές εντολές που μπορούμε να χρησιμοποιήσουμε.



Υπάρχουν δύο τύποι ανάπτυξης υπηρεσιών, replicated mode και global mode. Αναπτύσσοντας μια υπηρεσία σε replicated mode μας δίνεται η δυνατότητα να καθορίσουμε τον αριθμό των πανομοιότυπων εργασιών που θέλουμε να εκτελέσουμε. Μια global υπηρεσία είναι μια υπηρεσία που εκτελεί μια υπηρεσία σε κάθε κόμβο. Δεν υπάρχει προκαθορισμένος αριθμός εργασιών. Κάθε φορά που προσθέτετε έναν κόμβο στο σμήνος, ο ενορχηστρωτής δημιουργεί μια εργασία και ο προγραμματιστής αναθέτει την εργασία στον νέο κόμβο.

Παρακάτω δημιουργούμε ένα nginx container με όνομα mynginx. Αφού δεν προσδιορίσαμε το mode σε replicated ή global, το docker ορίζει το mode σε replicated.

```
→ ~ docker service create --name mynginx nginx
iix1m4de5c62eza0jr1cd3ogw
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
→ ~
```

Μας δίνεται η δυνατότητα να προσθέσουμε Replicas είτε κατά τη δημιουργία είτε αργότερα κάνοντας Scale. Παρακάτω εκτελούμε `$docker scale mynginx=2` οπότε πλέον έχουμε δύο replicas που εκτελούνται στους node3 και node1.

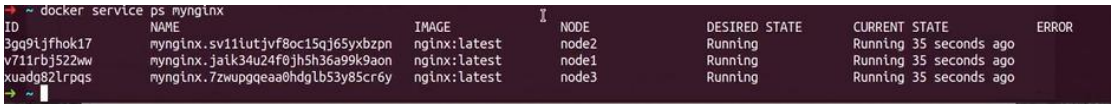
```
→ ~ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
iix1m4de5c62        mynginx             replicated          1/1                 nginx:latest
→ ~ docker service scale mynginx=2
mynginx scaled to 2
overall progress: 2 out of 2 tasks
1/2: running   [==================================================>]
2/2: running   [==================================================>]
verify: Service converged
→ ~ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
iix1m4de5c62        mynginx             replicated          2/2                 nginx:latest
→ ~ docker service ps mynginx
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE               ERROR               PORTS
53cdu7hmrlkt        mynginx.1           nginx:latest        node3               Running             Running about a minute ago
qvmgxog2v64l        mynginx.2           nginx:latest        node1               Running             Running 15 seconds ago
→ ~
```

Για να προσδιορίσουμε τον αριθμό των replicas κατά τη δημιουργία εκτελούμε `$docker service create - - name mynginx - -replicas 2 nginx`. Κάνουμε remove την υπηρεσία και την εκτελούμε με την παραπάνω εντολή.

```
→ ~ docker service rm mynginx
mynginx
→ ~ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
→ ~ docker service create --name mynginx --replicas 2 nginx
hzmxygij0qwdfofkhzvwp8xsp
overall progress: 2 out of 2 tasks
1/2: running   [==================================================>]
2/2: running   [==================================================>]
verify: Service converged
→ ~ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
hzmxygij0qwd        mynginx             replicated          2/2                 nginx:latest
→ ~
```

Στην περίπτωση που επιθυμούμε να αναπτύξουμε ένα κοντέινερ σε όλο το swarm cluster μας θα πρέπει να αναπτύξουμε την υπηρεσία μας σε global mode. Παρατηρούμε ότι έχουν ξεκινήσει και τα τρία κοντέινερ που διαθέτουμε και τα replicas είναι τρία καθώς διαθέτουμε τρία nodes στο cluster μας. Επίσης εκτελώντας `$docker service ps mynginx` παρατηρούμε πως ένα κοντέινερ εκτελείται σε έναν node.

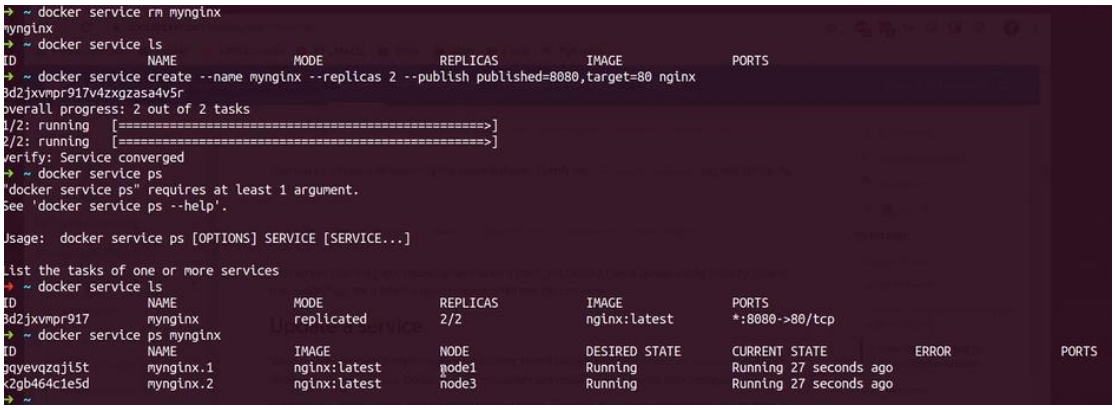
```
→ ~ docker service rm mynginx
mynginx
→ ~ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
→ ~ docker service create --name mynginx --mode global nginx
twcdd1ikhu6lxpx2q2h8y6m8i
overall progress: 3 out of 3 tasks
sv11iutjvf8o: running   [==================================================>]
jaik34u24f0j: running   [==================================================>]
7zwupgqeaa0h: running   [==================================================>]
verify: Service converged
→ ~ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
jaik34u24f0jh5h36a99k9aon *   node1               Ready               Active              Leader              19.03.5
sv11iutjvf8oc15qj65yxbzpn     node2               Ready               Active                                  19.03.5
7zwupgqeaa0hdglb53y85cr6y     node3               Ready               Active                                  19.03.5
→ ~ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
twcdd1ikhu6l        mynginx             global              3/3                 nginx:latest
→ ~
```



Παραπάνω είδαμε πως να δημιουργούμε μια υπηρεσία, στη συνέχεια θα δούμε πώς γίνεται να έχουμε πρόσβαση σε αυτήν. Προκειμένου η υπηρεσία μας να είναι προσβάσιμη θα πρέπει να καθορίσουμε το port mapping. Στη συνέχεια θα αναπτύξουμε την υπηρεσία μας εκτελώντας και το port mapping.

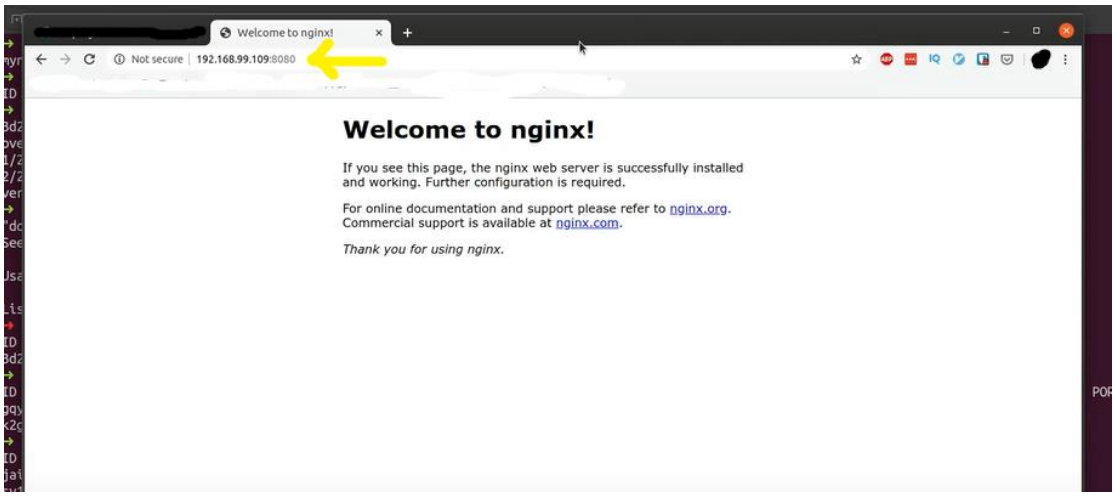
Αρχικά αφαιρούμε την ήδη υπάρχουσα υπηρεσία που είχαμε δημιουργήσει και στη συνέχεια δημιουργούμε την υπηρεσία με όνομα mynginx με δύο replicas. Οι δύο τιμές που ακολουθούν είναι η πόρτα του nginx container, που είναι η 80 και η πόρτα στην οποία κοινοποιούμε την υπηρεσία μας είναι η 8080.

Εκτελώντας $docker service ls βλέπουμε την υπηρεσία που αναπτύξαμε, ότι έχουμε δύο replicas και πως «ακούει» στην πόρτα 8080 σε όλο το docker swam. Εκτελώντας `$docker service ps mynginx` παρατηρούμε πως τα δύο replicas «τρέχουν» στους node1 και node3, αλλά υπάρχει πρόσβαση στο nginx application που αναπτύξαμε από όλους τους nodes του swarm μας, είτε πρόκειται για master είτε για worker node.



Εκτελώντας `$docker-machine ls` βλέπουμε τους τρείς κόμβους nodes που έχουμε στο swarm cluster μας, και τα nginx containers εκτελούνται στους κόμβους node1 και node3. Όμως υπάρχει πρόσβαση στη πόρτα 8080 από οποιοδήποτε μηχάνημα-node στο swarm cluster μας και θα μπορούμε να «φτάσουμε» στο nginx application που δημιουργήσαμε. Οπότε παίρνοντας την IP διεύθυνση του node2, στον οποίο δεν εκτελείται η υπηρεσία μας, και κάνουμε search στο browser μας βάζοντας την πόρτα 8080 θα πρέπει να δούμε την αρχική σελίδα του nginx.

```
→ ~ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
jaik34u24f0jh5h36a99k9aon *   node1               Ready               Active              Leader              19.03.5
sv11iutjvf8oc15qj65yxbzpn     node2               Ready               Active                                  19.03.5
7zwupgqeaa0hdglb53y85cr6y     node3               Ready               Active                                  19.03.5
→ ~ docker-machine ls
NAME    ACTIVE   DRIVER       STATE     URL                         SWARM   DOCKER     ERRORS
node1   *        virtualbox   Running   tcp://192.168.99.108:2376           v19.03.5
node2   -        virtualbox   Running   tcp://192.168.99.109:2376           v19.03.5
node3   -        virtualbox   Running   tcp://192.168.99.110:2376           v19.03.5
→ ~
```



6. Δημιουργία μιας εφαρμογής Word Press μέσω του Docker

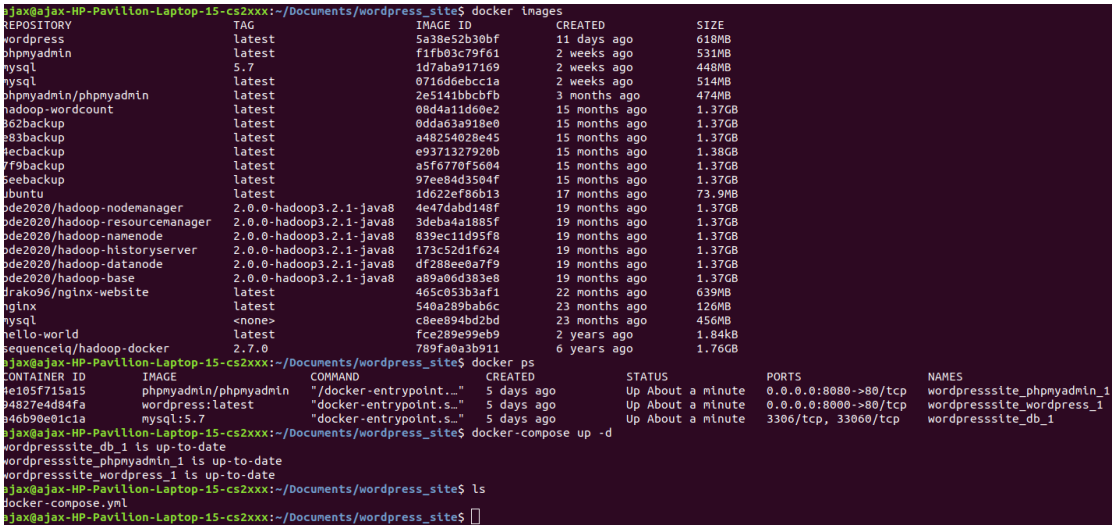
Το Word Press είναι μια δωρεάν πλατφόρμα δημιουργίας ιστότοπων ανοιχτού κώδικα. Σε πιο τεχνικό επίπεδο, το Word Press είναι ένα σύστημα διαχείρισης περιεχομένου (CMS) γραμμένο σε PHP που χρησιμοποιεί μια βάση δεδομένων MySQL. Πρόκειται για μια πλατφόρμα από την οποία μπορούν να σχεδιαστούν διάφοροι τύποι ιστότοπων. Από το blogging έως το ηλεκτρονικό εμπόριο και τους ιστότοπους επιχειρήσεων και χαρτοφυλακίων, το Word Press είναι ένα ευέλικτο CMS (content management system). Σχεδιασμένο με γνώμονα τη χρηστικότητα και την ευελιξία, είναι μια εξαιρετική λύση τόσο για μεγάλους όσο και για μικρούς ιστότοπους.

Προκειμένου να δημιουργήσουμε το δικό μας εξατομικευμένο Word Press Blog το οποίο δεν θα «τρέχει» στον υπολογιστή μας αλλά σε docker container μέσω του Docker που έχουμε εγκαταστήσει στον υπολογιστή μας, θα πρέπει δημιουργήσουμε ένα docker compose file το οποίο θα περιέχει κάποιες απαραίτητες υπηρεσίες (services) έτσι ώστε να εγκαταστήσουμε την εικόνα του Word Press (Word Press Image) καθώς και την εικόνα MySQL, που πρόκειται για τη βάση δεδομένων μας αλλά και την εικόνα phpmyadmin, που πρόκειται για ένα Web Interface της MySQL, η οποία μας επιτρέπει να ελέγχουμε τους πίνακες και τα στοιχεία της βάσης μας.

Ουσιαστικά μέσω του API του Word Press χρησιμοποιήσουμε το content management system και μέσω του REST API που επίσης παρέχει το Word Press μπορούμε να έχουμε react με το front-end.

Το docker-compose.ymp αρχείο που δημιουργήσαμε:

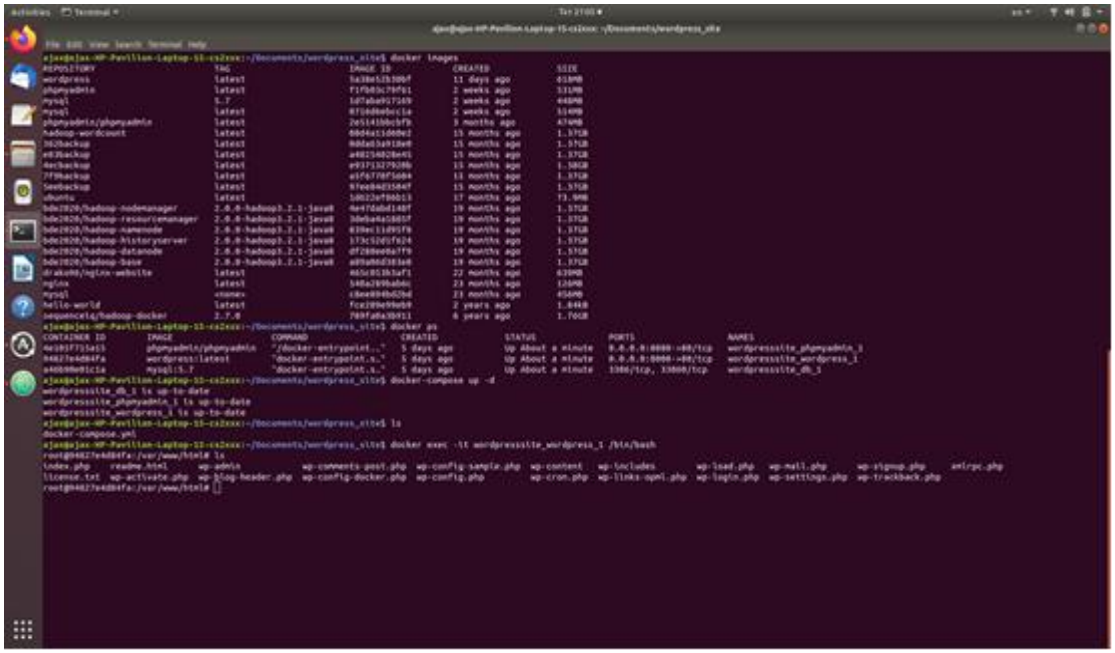
```
version: '3.4'

services:
  #Database
  db:
    image: mysql:5.7
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: password
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress
    networks:
      - wpsite
#phpmyadmin
  phpmyadmin:
    depends_on:
      - db
    image: phpmyadmin/phpmyadmin
    restart: always
    ports:
      - '8080:80'
    environment:
      PMA_HOST: db
      MYSQL_ROOT_PASSWORD: password
    networks:
      - wpsite
#wordpress
  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    ports: ["8000:80"]
    restart: always
  #  volumes: ['./:var/www/html'] #webroot for apache
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
    networks:
      - wpsite
networks:
  wpsite:
volumes:
  db_data:
```



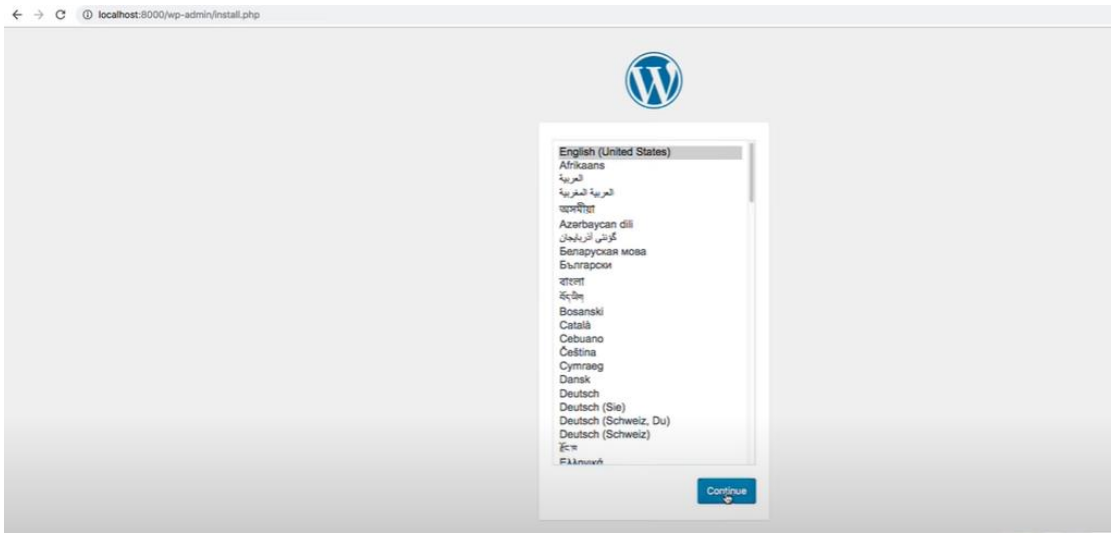
Με την εντολή `$docker images` εμφανίζονται όλες οι εικόνες που έχουμε στον υπολογιστή μας. Με `$docker ps` εμφανίζονται τα container που τρέχουν αυτή τη στιγμή. Παρατηρούμε πως τα container που επιθυμούμε είναι ενεργά. Στην περίπτωση που δεν είχαμε τις εικόνες στον υπολογιστή μας και εκτελούσαμε `$docker-compose up -d` θα γινόταν λήψη αυτών μέσω του Docker Hub.

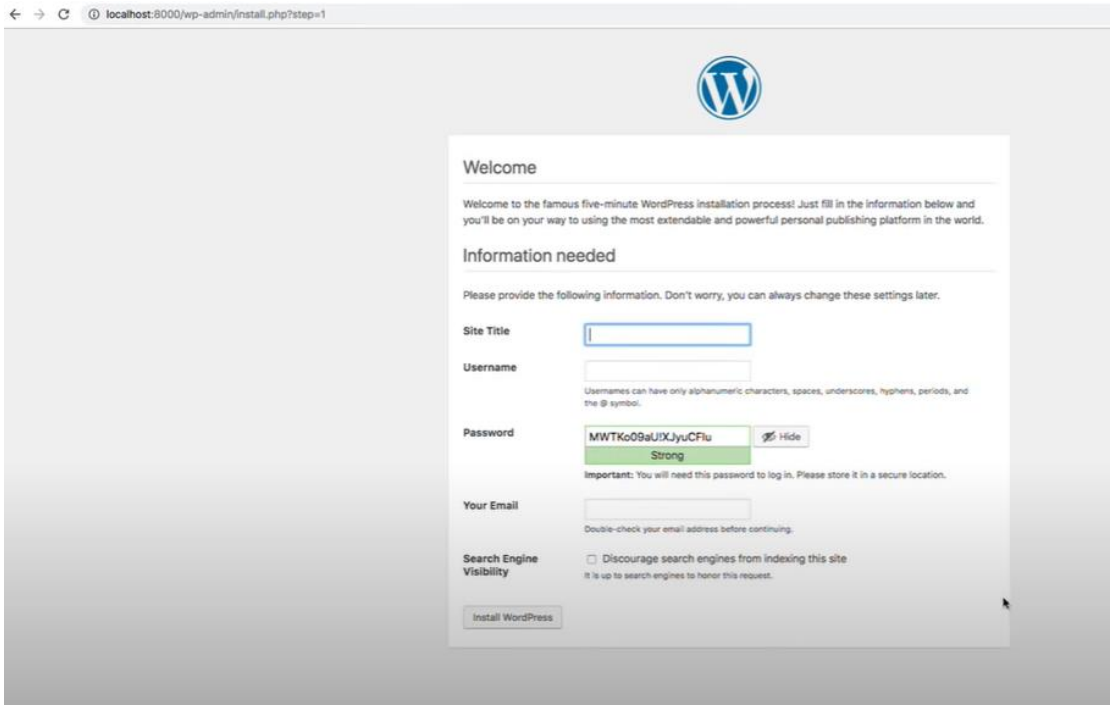
Εφόσον εκτελέσουμε `$docker-compose up -d` γίνεται λήψη των container που έχουμε ορίσει στο yml αρχείο μας, και ξεκινήσουν να «τρέχουν» και να δημιουργούνται όλοι οι κατάλογοι και τα php αρχεία πάνω στα οποία χτίζεται η ιστοσελίδα μας. Ανοίγοντας έναν browser και πηγαίνοντας στο http://localhost:8000 θα πάμε στο installation page του Word Press, θα προχωρήσουμε στην εγκατάσταση, και στο τέλος κάνοντας login με τα στοιχεία που δώσαμε θα μεταβούμε στο Dashboard της Word Press ιστοσελίδας μας.

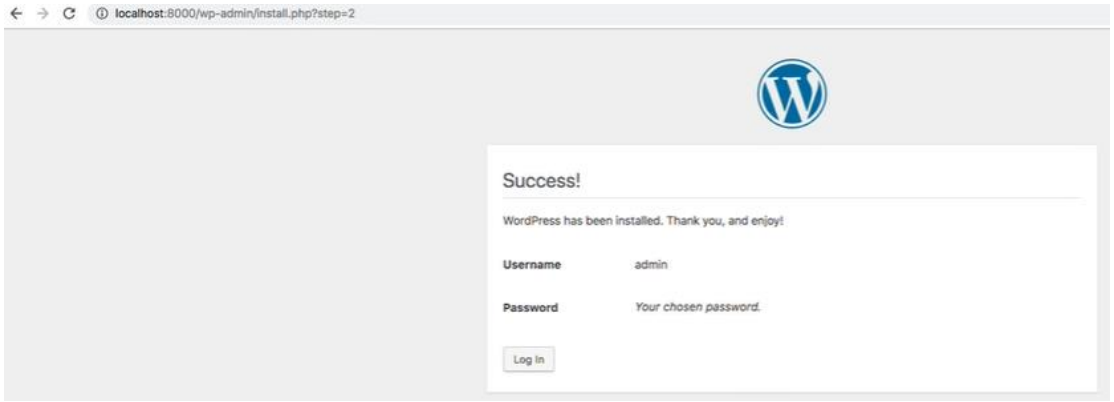
Εκτελώντας `$docker exec -it wordpresssite_wordpress_1 bin/bash` μπαίνουμε μέσα στο container μας όπου υπάρχουν όλοι οι κατάλογοι και τα php αρχεία πάνω στα οποία χτίζεται η ιστοσελίδα μας.

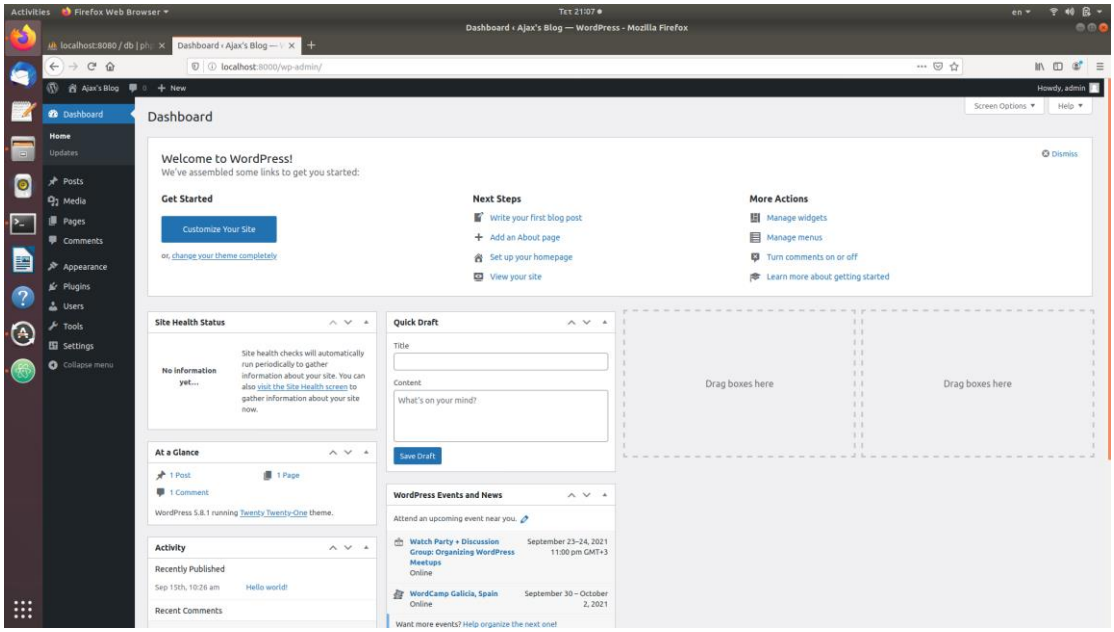


Ξεκινώντας επιλέγουμε γλώσσα, δίνουμε τίτλο στο site μας και ορίσουμε username, password και email και αφού ολοκληρώσουμε τη διαδικασία εγκατάστασης μεταβαίνουμε στο Dashboard όπου μας δίνεται η δυνατότητα να δουλέψουμε πάνω στο site μας.

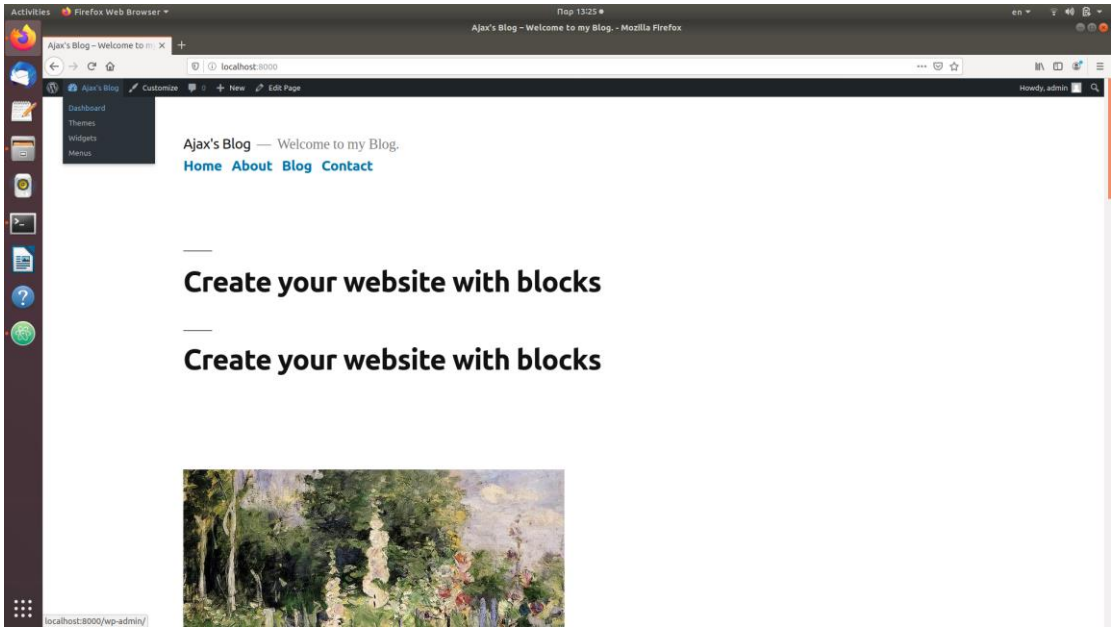



localhost:8000/wp-admin/install.php?step=1
Welcome
Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.
Information needed
Please provide the following information. Don't worry, you can always change these settings later.
Site Title
Username
Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.
Password
MWTKo09aUfXJyuCFlu
Hide
Strong
Important: You will need this password to log in. Please store it in a secure location.
Your Email
Double-check your email address before continuing.
Search Engine Visibility
Discourage search engines from indexing this site
It is up to search engines to honor this request.
Install WordPress

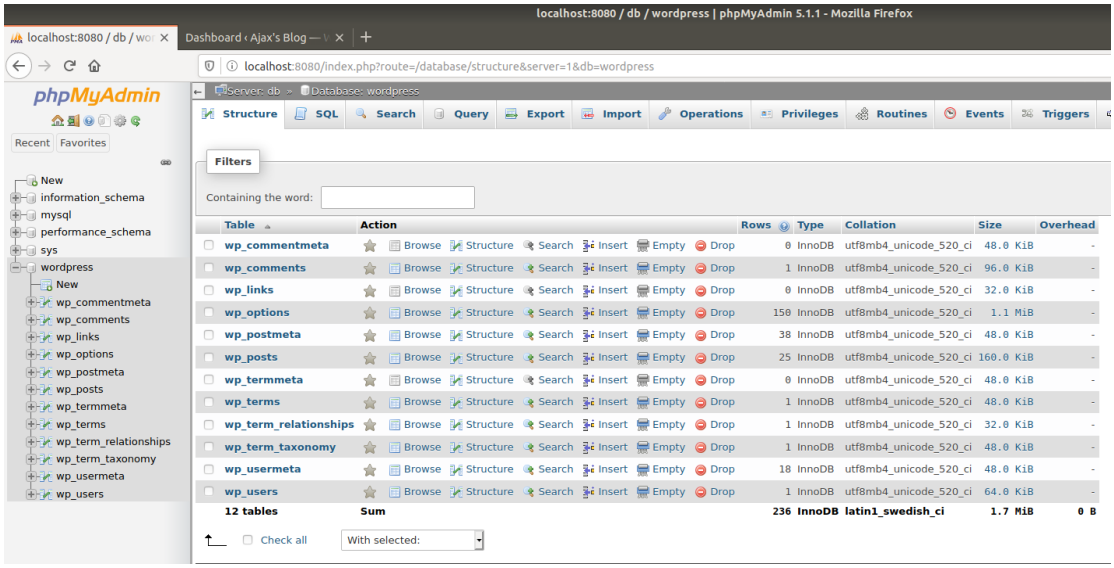

localhost:8000/wp-admin/install.php?step=2
Success!
WordPress has been installed. Thank you, and enjoy!
Username
admin
Password
Your chosen password.
Log In

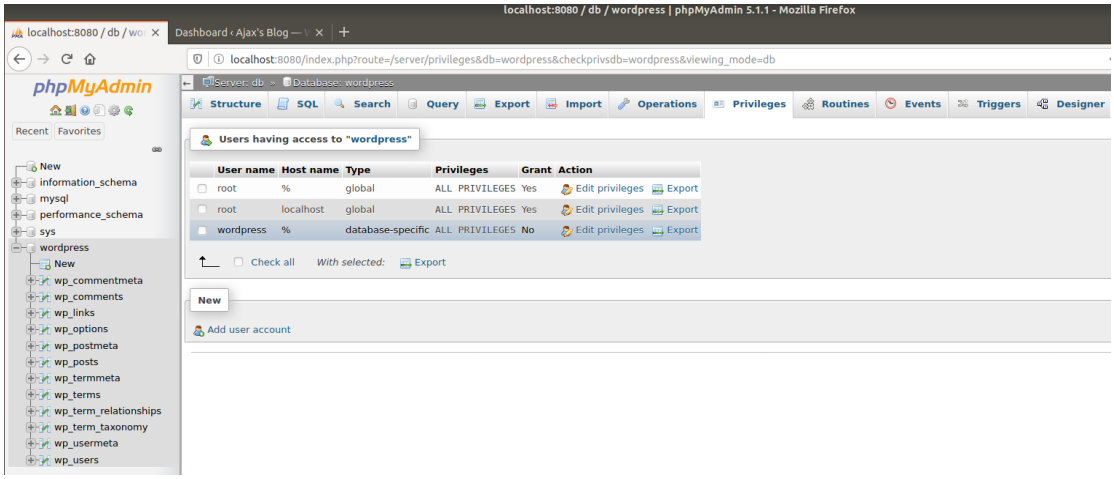

Activities
Firefox Web Browser
Dashboard ‹ Ajax's Blog — WordPress - Mozilla Firefox
localhost:8000/wp-admin/
Ajax's Blog
New
Howdy, admin
Dashboard
Screen Options
Help
Welcome to WordPress!
We've assembled some links to get you started:
Dismiss
Get Started
Customize Your Site
or, change your theme completely
Next Steps
Write your first blog post
Add an About page
Set up your homepage
View your site
More Actions
Manage widgets
Manage menus
Turn comments on or off
Learn more about getting started
Site Health Status
No information yet...
Site health checks will automatically run periodically to gather information about your site. You can also visit the Site Health screen to gather information about your site now.
Quick Draft
Title
Content
What's on your mind?
Save Draft
Drag boxes here
Drag boxes here
At a Glance
1 Post
1 Page
1 Comment
WordPress 5.8.1 running Twenty Twenty-One theme.
Activity
Recently Published
Sep 15th, 10:26 am
Hello world!
Recent Comments
WordPress Events and News
Attend an upcoming event near you.
Watch Party + Discussion Group: Organizing WordPress Meetups
Online
September 23–24, 2021
11:00 pm GMT+3
WordCamp Galicia, Spain
Online
September 30 – October 2, 2021
Want more events? Help organize the next one!

Επίσης επιλέγοντας πάνω αριστερά το όνομα του Blog μας μπορούμε να μεταβούμε στο front-end της ιστοσελίδας μας και να βλέπουμε τις όποιες αλλαγές κάνουμε.



Στο http://localhost:8080 τρέχει το phpmyadmin, κάνοντας login βρίσκουμε τη βάση δεδομένων μας η οποία περιέχει όλους τους πίνακες, και στα privileges εμφανίζονται οι χρήστες.



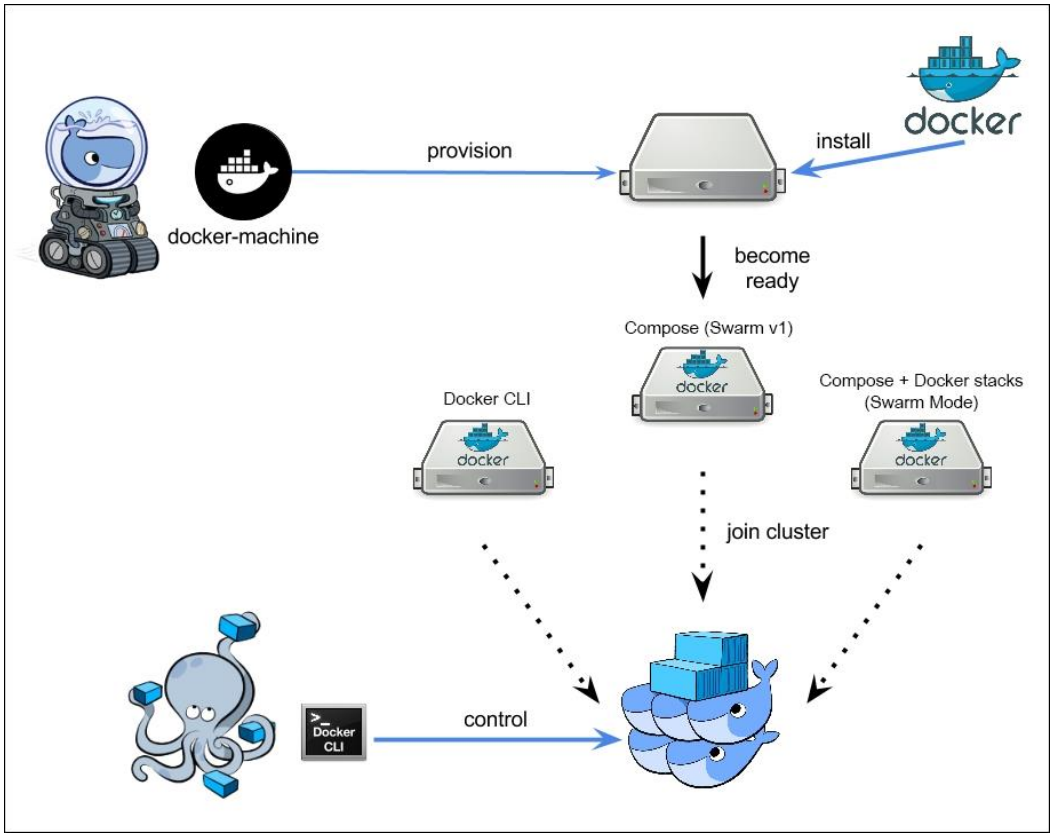


Κλείνοντας κάνοντας `$docker-compose down` σταματάμε τα containers που «τρέχουν».

```
ajax@ajax-HP-Pavilion-Laptop-15-cs2xxx:~/Documents/wordpress_site$ docker-compose down
Stopping wordpresssite_phpmyadmin_1 ... done
Stopping wordpresssite_wordpress_1  ... done
Stopping wordpresssite_db_1         ... done
Removing wordpresssite_phpmyadmin_1 ... done
Removing wordpresssite_wordpress_1  ... done
Removing wordpresssite_db_1         ... done
Removing network wordpresssite_wpsite
ajax@ajax-HP-Pavilion-Laptop-15-cs2xxx:~/Documents/wordpress_site$
```

7. Σύγκριση υλοποιήσεων και συμπεράσματα

Κλείνοντας, θα παραθέσουμε σε σύγκριση τη λειτουργία σμήνους του Docker (Docker Swarm) με τη μέθοδο του Docker Compose. Το Docker- Compose είναι ένα από τα πιο απλά εργαλεία που μας βοηθά να μετατρέψουμε την ιδέα των μικροϋπηρεσιών σε ένα λειτουργικό σύνολο Docker containers. Το Docker Swarm μας επιτρέπει να εκτελούμε πολλά αντίγραφα της εφαρμογής μας σε πολλούς διακομιστές. Εάν η μικροϋπηρεσία μας είναι γραμμένη με τρόπο ώστε να μπορεί να κλιμακωθεί «οριζόντια», τότε μας δίνεται η δυνατότητα να χρησιμοποιήσουμε το Docker Swarm για να αναπτύξουμε την εφαρμογή ιστού μας σε πολλά κέντρα δεδομένων και πολλούς συνδέσμων δικτύου. Αυτό προσφέρει ανθεκτικότητα έναντι της αποτυχίας ενός ή περισσοτέρων κέντρων δεδομένων ή συνδέσμων δικτύου.



Όσον αφορά τη χρήση και τη ροή εργασίας, και οι δύο τεχνολογίες λειτουργούν πολύ παρόμοια μεταξύ τους. Ο τρόπος με τον οποίο αναπτύσσεται η εφαρμογή χρησιμοποιώντας είτε το Docker Swarm είτε το Docker-Compose είναι πολύ παρόμοιος. Ορίζουμε την εφαρμογή μας σε ένα αρχείο YAML, αυτό το αρχείο θα περιέχει το όνομα της εικόνας, τη διαμόρφωση για κάθε εικόνα και επίσης την κλίμακα (αριθμός αντιγράφων) που κάθε μικροϋπηρεσία θα χρειαστεί να καλύψει κατά την ανάπτυξη. Η διαφορά έγκειται κυρίως στο back-end, όπου το Docker-Compose αναπτύσσει το κοντέινερ σε έναν μόνο κεντρικό υπολογιστή Docker, το Docker Swarm το αναπτύσσει σε πολλούς κόμβους.

Τόσο το Docker Swarm όσο και το Docker-Compose έχουν τις ακόλουθες ομοιότητες:

- Και οι δύο λαμβάνουν ορισμούς μορφοποιημένων YAML της στοίβας εφαρμογής.

- Και οι δύο προορίζονται να ασχοληθούν με εφαρμογές πολλαπλών container (μικροϋπηρεσίες).
- Και οι δύο έχουν παράμετρο κλίμακας (scale).
- Και οι δύο διατηρούνται από την ίδια εταιρεία, δηλαδή, Docker, Inc.

Οι λίγες διαφορές μεταξύ του Docker Swarm και του Docker-Compose:

- Το Docker Swarm χρησιμοποιείται για την κλιμάκωση της εφαρμογής ιστού σε έναν ή περισσότερους διακομιστές. Ως Docker-Compose θα εκτελεί η εφαρμογή ιστού μας σε έναν κεντρικό υπολογιστή Docker.
- Η κλιμάκωση της διαδικτυακής μας εφαρμογής Docker Swarm προσφέρει υψηλή διαθεσιμότητα και ανοχή σε σφάλματα. Η κλιμάκωση της διαδικτυακής μας εφαρμογής χρησιμοποιώντας το Docker-Compose σε έναν μόνο κεντρικό υπολογιστή είναι χρήσιμη μόνο για δοκιμές και ανάπτυξη.
- Το Docker Swarm και οι σχετικές υποεντολές όπως το Docker Swarm και το Docker Stack είναι ενσωματωμένες στο ίδιο το Docker CLI. Είναι όλα μέρος του δυαδικού Docker που καλείτε μέσω του τερματικού μας. Το Docker-Compose είναι αυτόνομο δυαδικό από μόνο του.

Όπως αναλύσαμε παραπάνω, και τα δύο είναι εντελώς διαφορετικά εργαλεία και το καθένα λύνει ένα εντελώς διαφορετικό πρόβλημα, οπότε καταλαβαίνουμε πως το ένα δεν αποτελεί μια εναλλακτική λύση για το άλλο.

Στο Word Press Blog που αναπτύξαμε σε έναν μόνο διακομιστή, η ρύθμιση ή η διατήρησή του δεν είναι κάτι που θέλουμε να κάνουμε, χειροκίνητα, οπότε αυτό που κάναμε είναι να εγκαταστήσουμε το Docker και το Docker-Compose στον υπολογιστή μας και να δημιουργήσουμε ένα αρχείο YAML που καθορίζει όλες τις πτυχές της στοίβας Word Press. Ενώ το Docker-Compose είναι περισσότερο ένα εργαλείο αυτοματισμού, το Docker Swarm προορίζεται για πιο απαιτητικές εφαρμογές. Εφαρμογές ιστού με εκατοντάδες ή χιλιάδες χρήστες ή φόρτο εργασίας που πρέπει να κλιμακωθεί παράλληλα. Οι εταιρείες με μεγάλη βάση χρηστών και αυστηρές απαιτήσεις SLA θα ήθελαν να χρησιμοποιήσουν ένα κατανεμημένο σύστημα όπως το Docker Swarm. Εάν η εφαρμογή μας εκτελείται σε πολλούς διακομιστές και πολλά κέντρα δεδομένων, τότε οι πιθανότητες διακοπής λειτουργίας λόγω επηρεασμένου συνδέσμου DC ή δικτύου μειώνονται σημαντικά.

- Docker - when you want to deploy a single (network accessible) container
- Docker Compose - when you want to deploy multiple containers to a single host from within a single YAML file
- Docker swarm - when you want to deploy a cluster of docker nodes (multiple hosts) for a simple, scalable application

Εικόνα: When to use

8. Βιβλιογραφία

[1] Thomas Erl, Cloud Computing, Αρχές, Τεχνολογίες & Αρχιτεκτονική.

[2] Silberschatz Galvin-Gagne, Λειτουργικά Συστήματα

[3] William Stallings, Λειτουργικά Συστήματα. Αρχές σχεδίασης.

[4] Bugnio, E. *Hardware and Software Support for Virtualization.*

[5] Callman, C. (2019, December 2). *RedHat*. Retrieved from https://www.redhat.com/en/topics/containers/what-is-container-orchestration#overview

[6] Christopher, W. *Networking for VMware Administrators.*

Docs, D. *Swarm mode overview.* Retrieved from https://docs.docker.com/engine/swarm/

[7] Eldridge, I. (2018, 06 17). *New Relic.* Retrieved from https://newrelic.com/blog/best-practices/container-orchestration-explained

[8] Hosseinzadeh, S. (2017). *Diversification and obfuscation techniques for software security: A systematic literature review.*

[9] Kapoor, R. *Nginx Blog*. Retrieved from https://www.nginx.com/blog/

[10] Malcony, T. (2020, Octomber 19). *Baelbung*. Retrieved from https://www.baeldung.com/cs/virtualization-vs-containerization

[11] Markovich, J. (2021, April 18). *Devopscube*. Retrieved from https://devopscube.com/how-to-install-and-configure-docker/

[12] Pollock, A. *Liquid Web*. Retrieved from https://www.liquidweb.com/kb/virtualization-vs-containerization/

[13] Simic, S. *Phonixnap.* Retrieved from https://phoenixnap.com/kb/docker-image-vs-container

[14] Stokes, J. *Inside the Machine.*

[15]Troy, R. *VMware Cookbook: A real world guide to effective VMware use.*

[16]RanvirSingh,DevOps.Retrieved from: https://linuxhint.com/docker_compose_vs_docker_swarm/

[17]Jack Wallen in Data Centers on March 16, 2021, 8:48 AM PST. Retrieved from: https://www.techrepublic.com/article/simplifying-the-mystery-when-to-use-docker-docker-compose-and-kubernetes/

[18] Russ McKendrick, Mastering Docker. Master this widely used containerization tool, 2nd Edition.

[19] Van Ham, J.C., Rijsenbrij, J.C. Development of Containerization. Success through Vision, Drive and Technology

[20] Kubernetes and Docker –An Enterprise Guide

[21] James Turnbull, The Docker book.

[22]Arun Kumar, Docker. A complete beginner's guide.

[23] ProTechGurus, Docker Container Ultimate Beginners Guide.

Docker Concept and Hands-On Exercises.

[24] Thomas Uphill, John Arundel, Neependra Khare, Ke-Jou Carol Hsu, 2017,

DevOps. Puppet, Docker, and Kubernetes.

[25] Joseph Muli, Beginning DevOps with Docker. Automate the deployment of your environment with the power of the Docker toolchain

[26] Nigel Poulton, 2017, Docker Deep Dive

[27] Ian Miell, Aidan Hobson Sayers, 2017, Docker in Practice, Second Edition

[28] Luca Osborne, 2019, Docker. The ultimate step-by-step guide for beginners to learn and master Docker programming.