Εισαγωγή στο Docker

Τι είναι το Docker ?

- Το Docker είναι ένα λογισμικό που προσφέρει «Εικονικοποίηση σε επίπεδο ΛΣ» (Operating-System-Level Virtualization) ή αλλιώς **Containers**.

Βασικά Συστατικά:

- Containers
- Images

Τι είναι ένα Container?

- Ένα απομονωμένο περιβάλλον μέσα στο οποίο μπορούμε να τρέξουμε όποια εφαρμογή θέλουμε.
- Οι πόροι που δίνονται στην εφαρμογή (πχ CPU, RAM) καθώς και πολλές άλλες παράμετροι ρυθμίζονται μέσω του Docker.

Πώς λειτουργεί ?

- Το Docker εκμεταλλεύεται την LXC (Linux Containers) λειτουργικότητα του linux kernel.
- Στηρίζεται πάνω στις low-level λειτουργίες που του παρέχονται ώστε να χτίσει high-level features.

- LXC = Namespaces + Control Groups
  - Τα control groups περιορίζουν τι μπορεί να ***χρησιμοποιήσει*** ένα process (πχ cpu, disk io)
  - Τα namespaces περιορίζουν τι μπορεί να ***δει*** ένα process (πχ ποιες διεργασίες βλέπει, τι δίκτυο βλέπει)
- Πρακτικά μηδενικό overhead για τη χρήση των λειτουργιών αυτών

Images

- Πρακτικά είναι η εφαρμογή που τρέχει μέσα σε ένα docker container.
- Τα images περιέχουν όλες τις βιβλιοθήκες και τα dependencies που χρειάζονται για να τρέξουν
- Μερικά παραδείγματα γνωστών εφαρμογών:
  - MySQL, PSQL
  - Elasticsearch
  - Nodejs
  - Apache / Nginx

Linux Containers vs Docker Containers

Linux Containers

liblxc

namespaces

cgroups

SELinux/AppArmor

Linux kernel

Docker 1.10 and later

runC

runC

runC

containerd-shim

containerd-shim

containerd-shim

containerd

Docker Engine

namespaces

cgroups

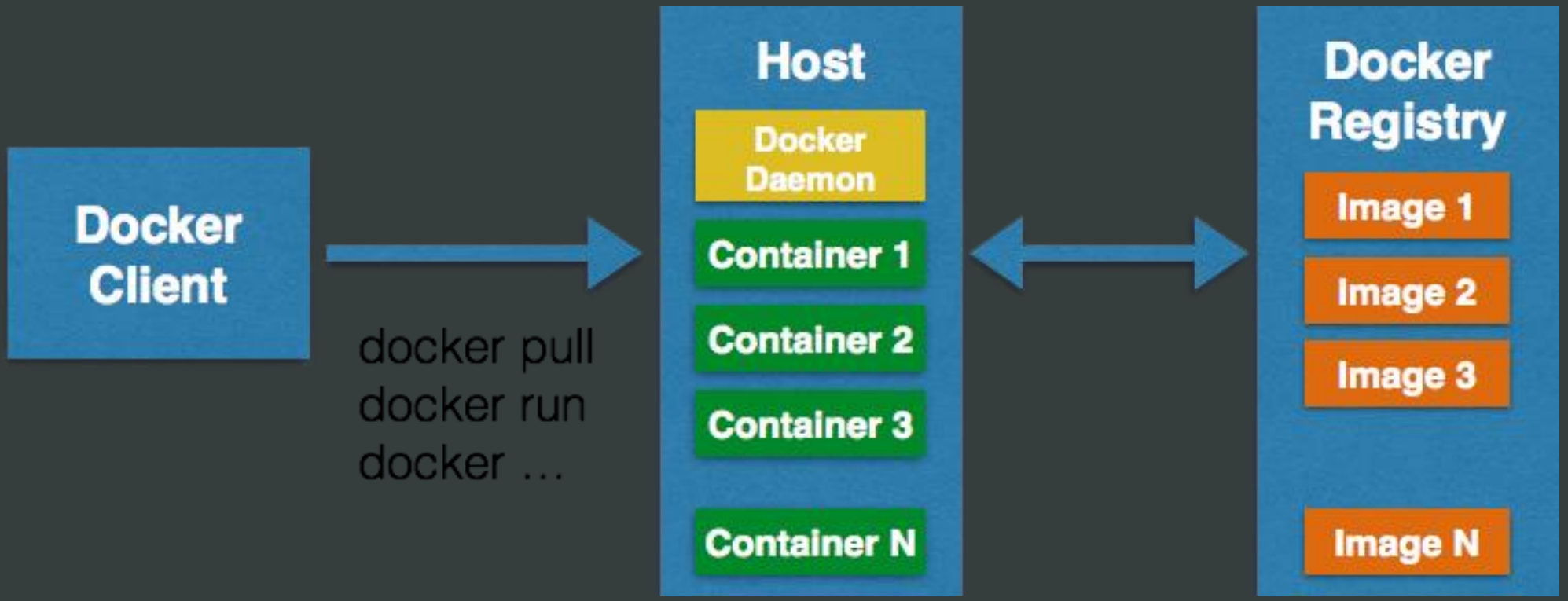
SELinux/AppArmor

Linux kernel

Docker Hub

- Docker Hub is a cloud-based registry service which allows you to link to code repositories, build your images and test them, stores manually pushed images. It provides a centralized resource for container **image discovery**, **distribution** and **change management**, **user and team collaboration**, and **workflow automation** throughout the development pipeline.
- Πρακτικά ένα github για images !
- Απλά τώρα το version control γίνεται ένα επίπεδο πιο πάνω (images αντί code).
- Είναι ένας από τους κύριους λόγους που το docker είχε τεράστια αποδοχή.

Docker Hub


Docker Client
docker pull
docker run
docker …
Host
Docker Daemon
Container 1
Container 2
Container 3
Container N
Docker Registry
Image 1
Image 2
Image 3
Image N

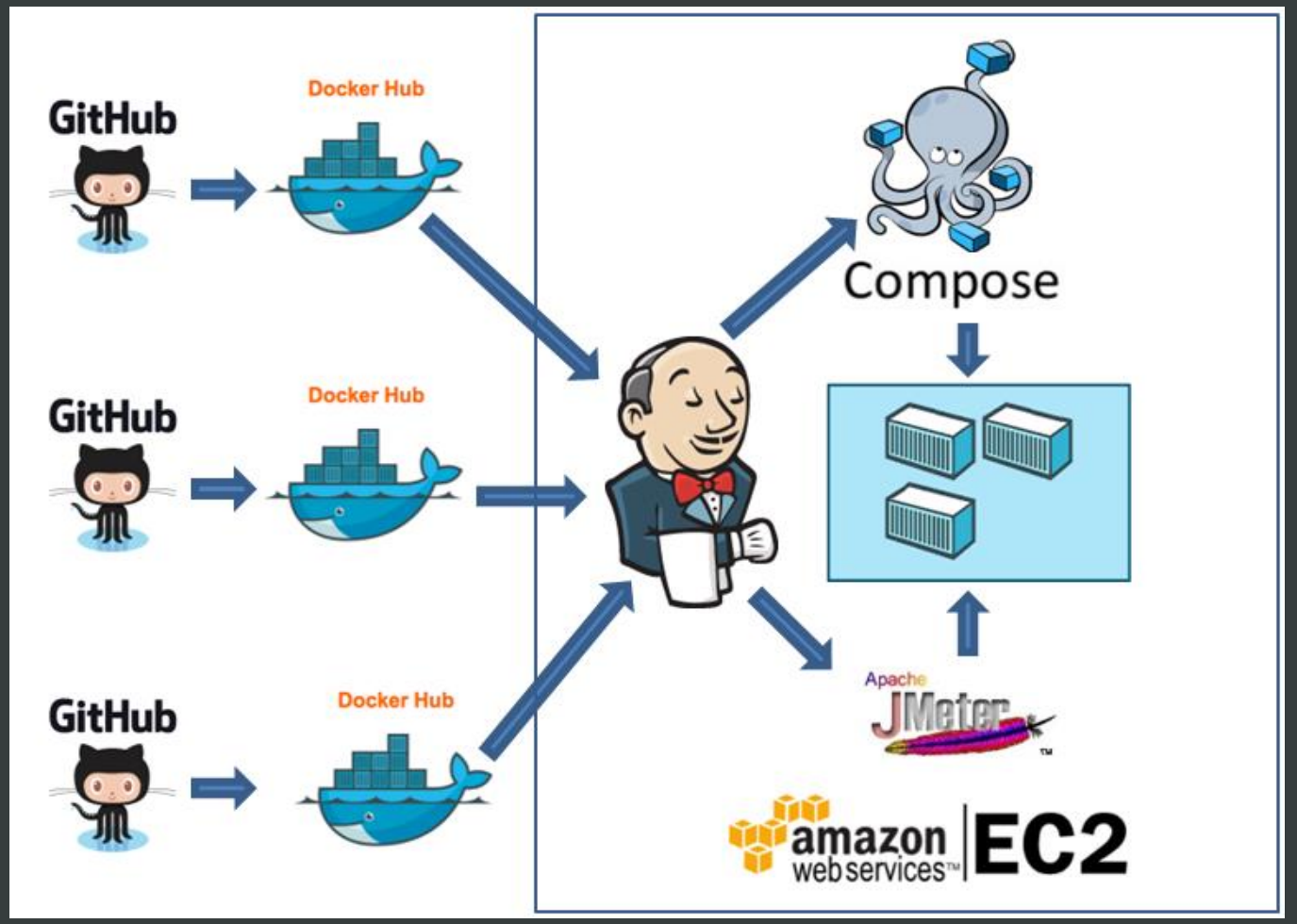
Τι μας προσφέρει όλο αυτό?

- Portability:
  - Τα images μας στηρίζονται στο docker engine για να τρέξουν. Αν τρέχουν σε εμάς θα τρέχουν και σε **οποιονδήποτε άλλον υπολογιστή** με Docker.
  - Πράγμα που σημαίνει ότι όλοι οι developers αναπτύσσουν στο **ΙΔΙΟ** κοινό περιβάλλον.
    - Εύκολο και Ελεγχόμενο Testing
    - No cross-compatibility issues
  - Consistent environment from Development, to Testing, to Production
  - Continuous integration made easy

Τι μας προσφέρει όλο αυτό?

- Deployment:
  - Εξαιρετική ευκολία στο deployment μιας εφαρμογής. Μόλις γίνει build ένα image μπορεί να γίνει deploy παντού (build once, deploy everywhere):
    - Bare Metal
    - Virtual Machines (πχ AWS, Google Cloud)
  - Ταχύτητα στο deployment, αφού η εκκίνηση ενός container δεν απαιτεί την εκκίνηση ενός ΛΣ, συγκριτικά με deployment σε VMs.
- Isolation and Security:
  - Οι εφαρμογές τρέχουν σε containers και δεν μπορούν να δουν η μία στοιχεία της άλλης.

CI/CD with Docker

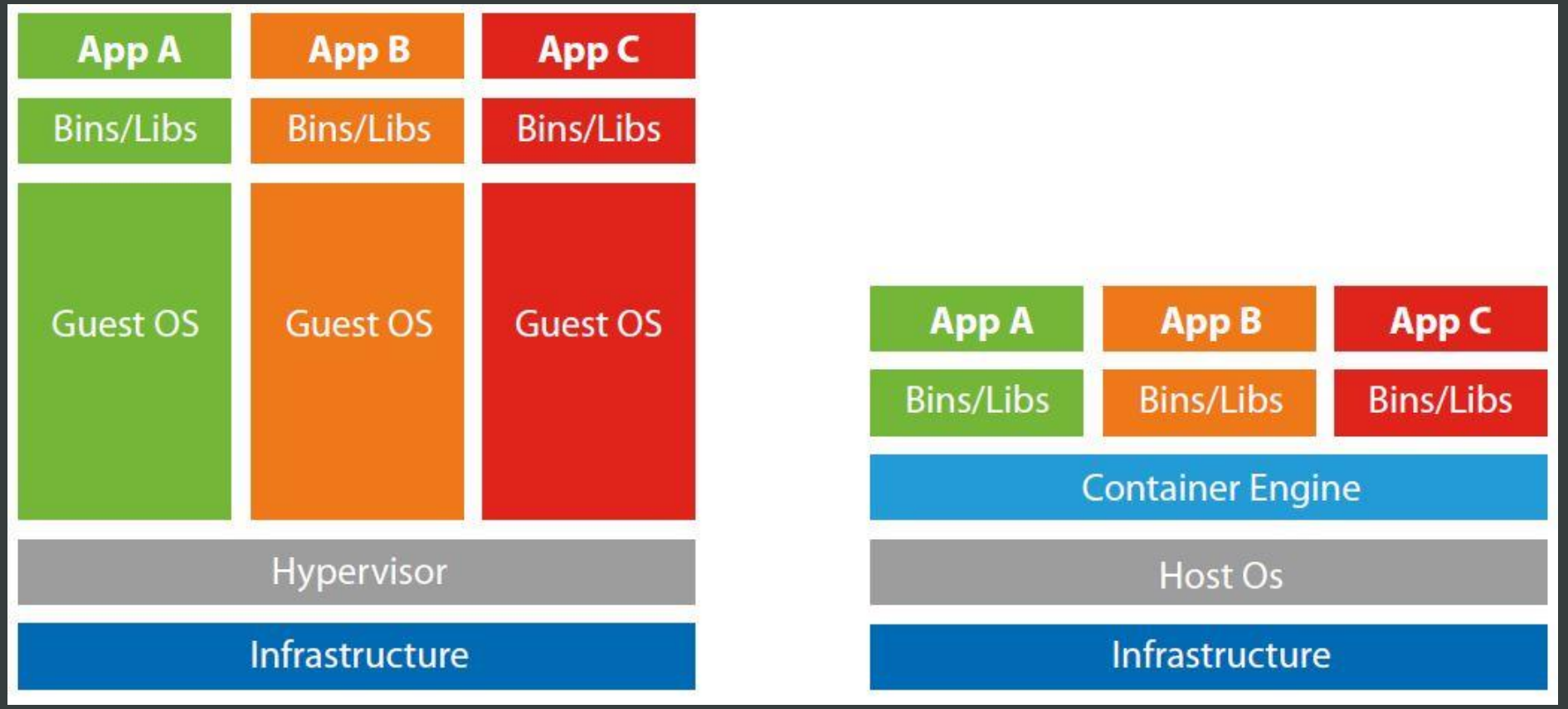


Containers vs VMs

- Finer Grained Control + Zero Down Time:
  - Η αλλαγή ενός app που τρέχει σε VM συνεπάγεται την επανεκκίνηση του VM
  - Δηλαδή επανεκκίνηση του Guest ΛΣ (ακριβή διαδικασία)
  - Το σταμάτημα και ξεκίνημα ενός container είναι μια πολύ γρήγορη διαδικασία, καθώς είναι σαν να ξεκινάμε ένα νέο process
- Λιγότερο overhead:
  - Η χρήση VM συνεπάγεται την εξομοίωση υλικού και το τρέξιμο ενός Guest ΛΣ πάνω από το κανονικό. Αυτό έχει ένα μεγάλο overhead σε υπολογιστικούς πόρους
- No vendor lock-in

Τα VMs μοιράζουν το υλικό σε πολλά συστήματα, ενώ τα containers μοιράζουν ένα σύστημα σε πολλές εφαρμογές.

Containers vs VMs



Ένα απλό παράδειγμα με κώδικα

- Θα στήσουμε ένα πολύ απλό development environment για να δούμε τα πλεονεκτήματα του Docker στην πράξη.
- Εγκατάσταση Postgres:

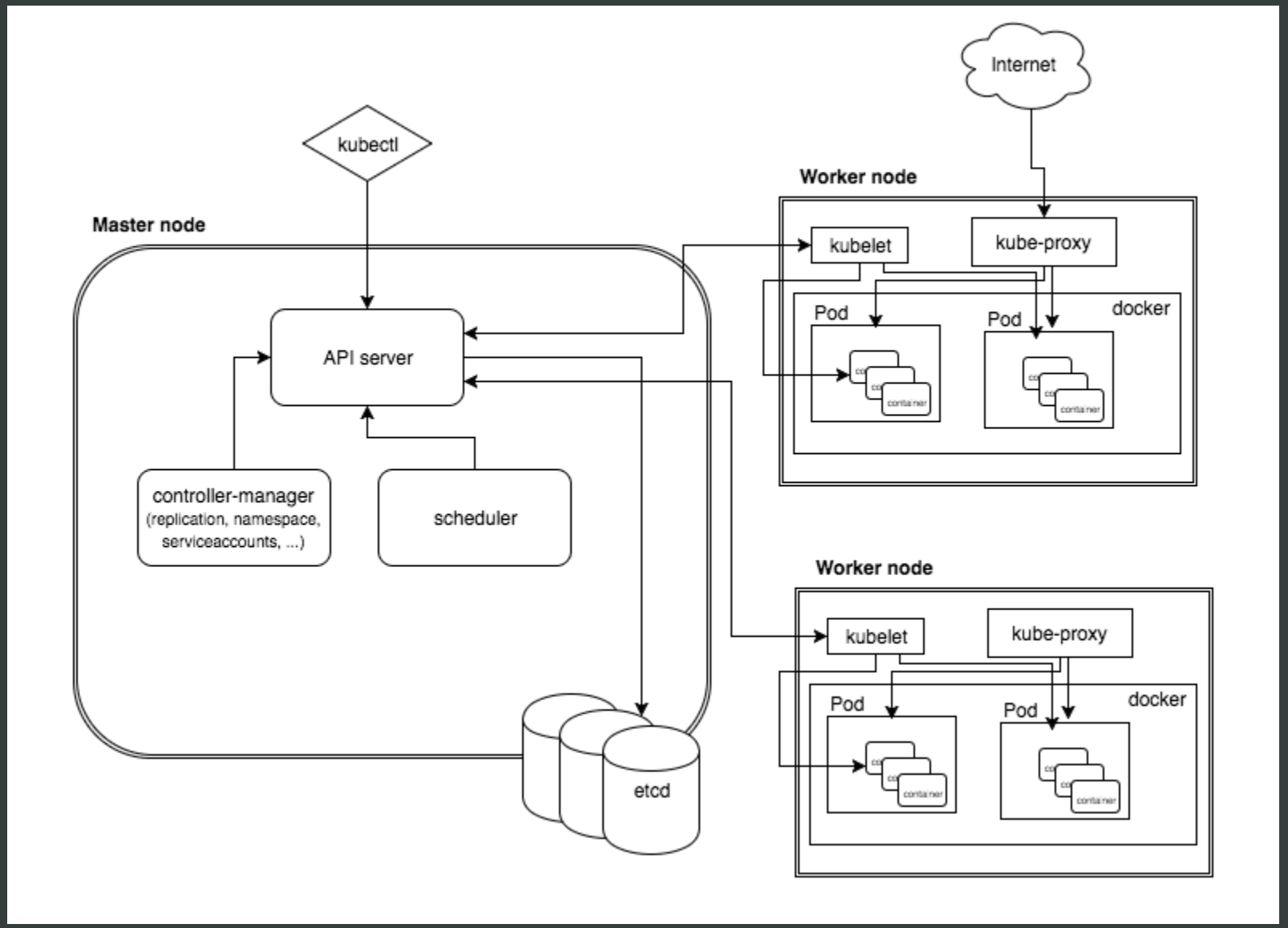
```
# Pull postgres image
docker container pull postgres:9.5.10

# Run postgres image
docker container run \
--name postgresDB \
-p 5432:5432  \
-e POSTGRES_USER=dev \
-e POSTGRES_PASSWORD=password \
-e POSTGRES_DB=devdb \
-d \
postgres:9.5.10
```

Containers + Microservices

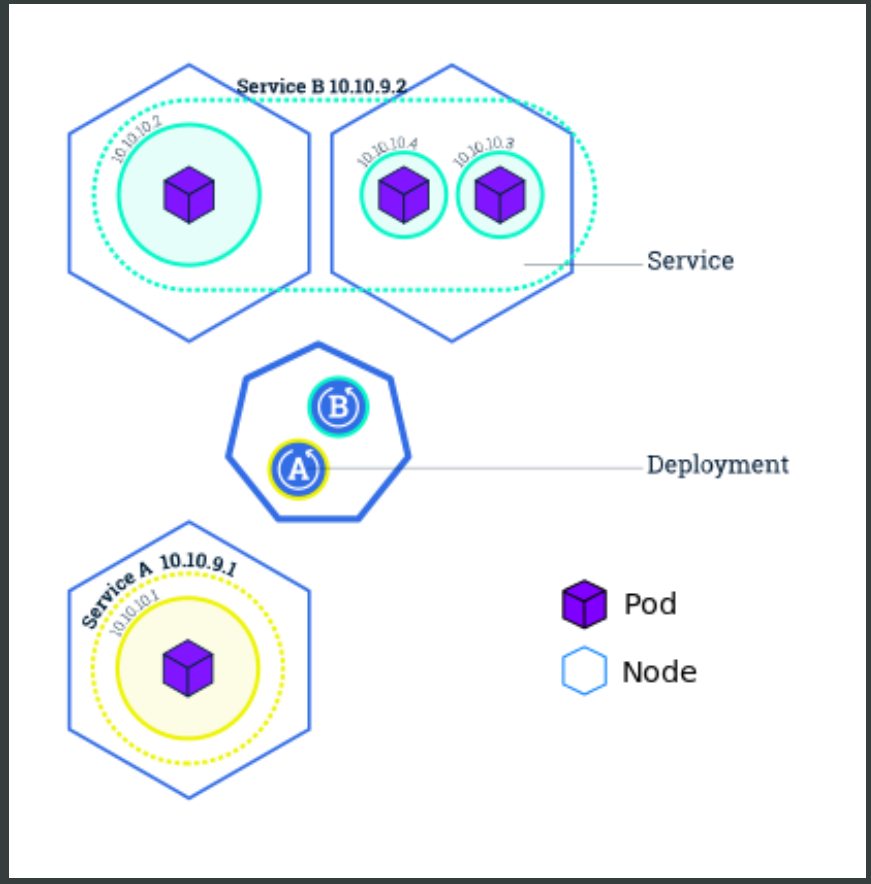
- Όπως έχουμε δει μέχρι στιγμής, το Docker προάγει το loose coupling μεταξύ των συστατικών μιας εφαρμογής
- Δουλεύει ιδιαίτερα καλά με Service Oriented Architectures
- Μπορούμε πχ να αντιστοιχίσουμε κάθε service με ένα Docker Image:
  - Πολύ εύκολο scaling (απλά τρέξε περισσότερα containers)
  - Κάθε service μπορεί να αναπτυχθεί ξεχωριστά και σε ένα consistent περιβάλλον για όλες τις ομάδες
  - Για πιο προχωρημένες λειτουργίες (load balancing, self healing, automated rollouts and rollbacks) έχουν αναπτυχθεί εξειδικευμένες λύσεις γύρω από τη φιλοσοφία των containers
    - Container Orchestrators: πχ Kubernetes

Kubernetes - The future of Microservices



Kubernetes - The future of Microservices

- Container Orchestration System βασισμένο στο Google Borg
- Δομική μονάδα το container
- Out of the box microservices
- Scales to 1000s of nodes
- Self-Healing, Automated rollouts and rollbacks
- Declarative interface



Βιβλιογραφία

- https://docs.docker.com/
- https://kubernetes.io/docs/concepts/
- https://medium.com/aws-activate-startup-blog/using-containers-to-build-a-microservices-architecture-6e1b8bacb7d1
- https://medium.com/jeroen-rosenberg/from-monolith-to-microservice-architecture-on-kubernetes-part-1-the-api-gateway-eb82f8c2d10c
- https://www.infoworld.com/article/3204171/linux/what-is-docker-linux-containers-explained.html