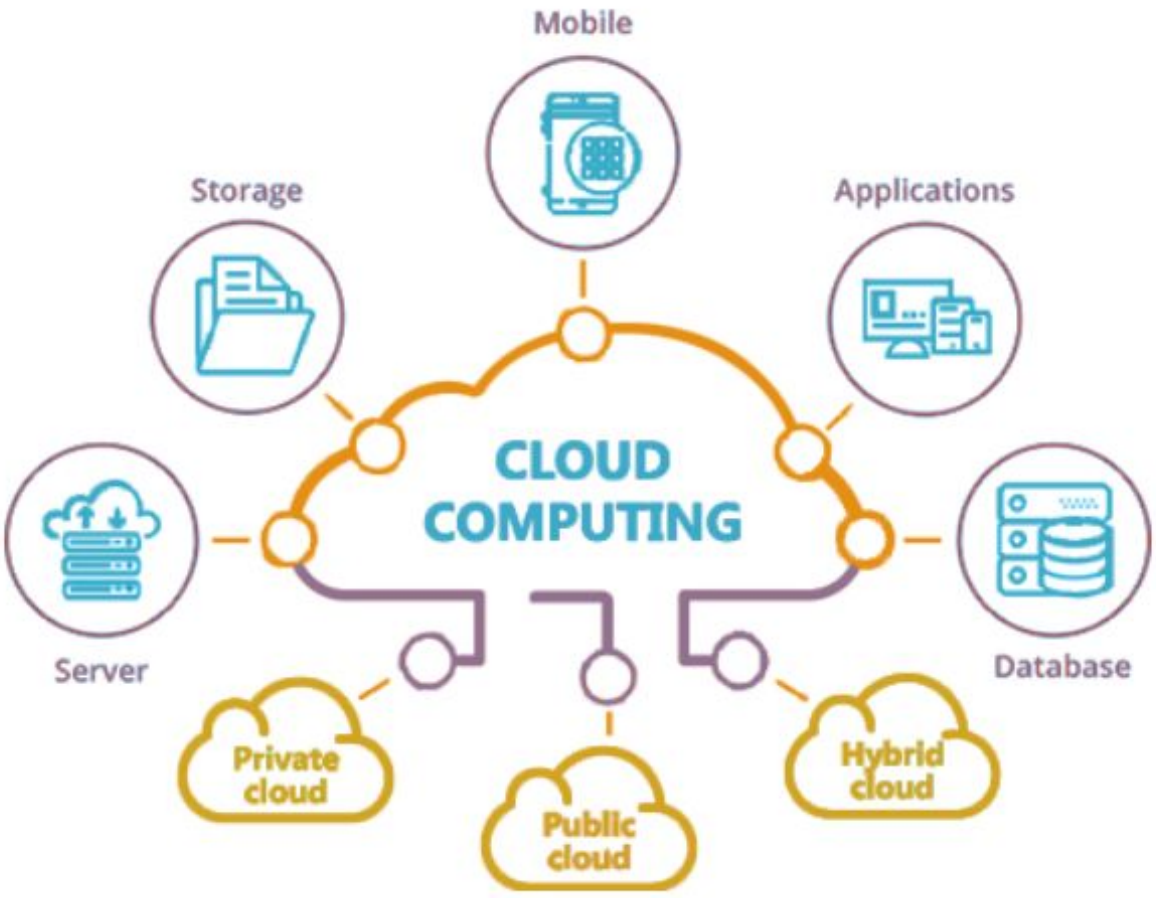
Υπολογιστική Νέφους και Υπηρεσίες
Τελικό Project Εξαμήνου : Jupyter Notebook Docker Swarm
Τμήμα Η3



Ομάδα :

- ***Χαράλαμπος Ασημακόπουλος, cs141098***
- ***Ιωάννα Βασιλείου, cs141190***
- ***Ακριβή Παπαγεωργίου, cs141026***
- ***Φανερωμένη Τηλιακού, cs141199***

Πίνακας Περιεχομένων

Περίληψη	**2**
Υπηρεσίες	**2**
Βασικές Έννοιες	**2**
Kubernetes vs Docker Swarm	**4**
GlusterFS	**6**
Ανάλυση Αρχείων	**7**
Βήματα Εγκατάστασης	**8**
Τεχνική Περιγραφή	**9**
Παραδείγματα	**10**
Βιβλιογραφία	**14**

Περίληψη

Σκοπός της εργασίας μας είναι να παρέχουμε τη δυνατότητα σε όσους δεν έχουν "ισχυρό υπολογιστή" να εκτελεί οποιαδήποτε γλώσσα προγραμματισμού. Η εφαρμογή βασίζεται στη βιβλιοθήκη Tensorflow και στο περιβάλλον Jupyter, όπου είναι μία real time πλατφόρμα. Οι γλώσσες που υποστηρίζει είναι Java, PHP, Python 3, Python 3.7, R, Matlab, NodeJS. Για την εκτέλεση της εργασίας δημιουργήθηκαν διάφορα αρχεία όπως: Dockerfile, docker-compose.yaml, swarm και gluster.

Υπηρεσίες

- Jupyter Notebook - 2 replicas / worker
- Nginx Reverse Proxy - 2 replicas / manager
- Portainer Monitoring - 1 replica / manager
- GlusterFS - τοπική εγκατάσταση σε κάθε server για διαμοιραζόμενο χώρο αποθήκευσης των container.

Βασικές Έννοιες

- **Tensorflow :** Είναι μια ολοκληρωμένη πλατφόρμα ανοιχτού κώδικα για Machine Learning (ML). Διαθέτει ένα ολοκληρωμένο και ευέλικτο σύστημα εργαλείων, βιβλιοθηκών και πόρων, όπου επιτρέπει στους ερευνητές να προωθήσουν την τεχνολογία στο ML και οι προγραμματιστές να δημιουργήσουν και να αναπτύξουν εύκολα αντίστοιχες εφαρμογές και μοντέλα.

- **Jupyter :** Χρησιμοποιείται για την ανάπτυξη open-source software, εξελίχθηκε για την υποστήριξη της διαδραστικής επιστήμης και του επιστημονικού υπολογισμού σε όλες τις γλώσσες προγραμματισμού. Επιτρέπει την δημιουργία και τον διαμοιρασμό εγγράφων που περιέχουν ζωντανό κώδικα.

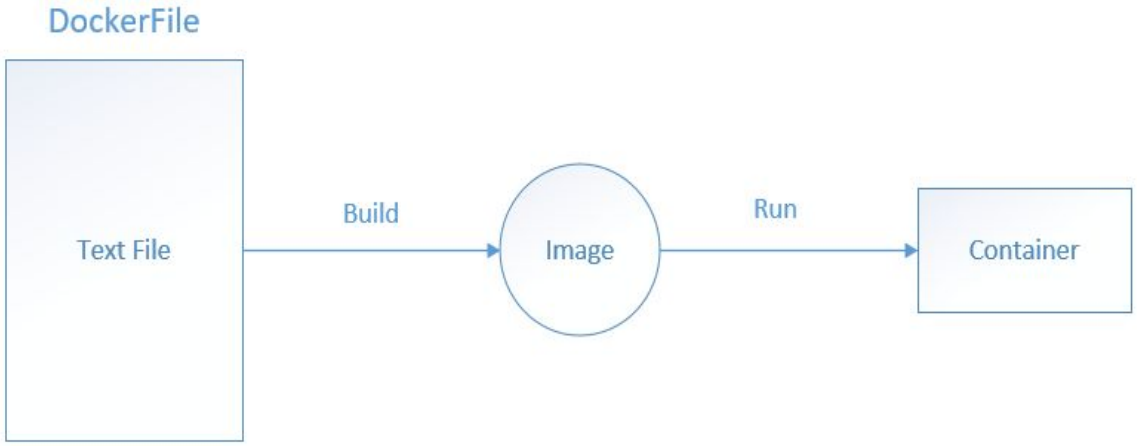
- **Docker :** Είναι ένα εργαλείο που έχει σχεδιαστεί για να διευκολύνει τη δημιουργία, την ανάπτυξη και την εκτέλεση εφαρμογών χρησιμοποιώντας containers. Ακόμα, βασίζεται στις εικόνες (images). Το Docker μοιάζει λίγο με μια εικονική μηχανή, αλλά αντί να δημιουργεί ένα ολόκληρο εικονικό λειτουργικό σύστημα, επιτρέπει στις εφαρμογές να χρησιμοποιούν τον ίδιο πυρήνα Linux με το σύστημα στο οποίο εκτελούνται, και απαιτεί μόνο την αποστολή εφαρμογών που δεν εκτελούνται ήδη στον κεντρικό υπολογιστή. Αυτό δίνει σημαντική ώθηση στην απόδοση και μειώνει το μέγεθος της εφαρμογής.

- **Docker Images :** Είναι συνδυασμός ενός συστήματος αρχείων και παραμέτρων. Για να δούμε τη λίστα των διαθέσιμων εικόνων εκτελούμε *docker images*.

- **Containers :** Είναι instances του Docker Image. Επιτρέπουν σε έναν προγραμματιστή να πακετάρει μια εφαρμογή με όλα τα μέρη που χρειάζεται, όπως βιβλιοθήκες ώστε να την αναπτύξει ως ένα πακέτο. Με αυτόν τον τρόπο, ο προγραμματιστής μπορεί να είναι σίγουρος ότι η εφαρμογή θα εκτελεστεί σε οποιοδήποτε άλλο μηχάνημα Linux.

- **DockerFile :** Είναι ένα απλό έγγραφο κειμένου που περιέχει όλες τις εντολές που απαιτούνται για τη δημιουργία μιας εικόνας. Όσο για τη δομή του DockerFile πρέπει να γνωρίζουμε τα εξής :
    1. Πάντα ξεκινάει με το FROM.
    2. Με το COPY γίνεται αντιγραφή των απαραίτητων αρχείων για διαμόρφωση.
    3. Με το ENV προσθέτουμε μεταβλητές σε Linux.
    4. Με το EXPOSE θέτουμε την πόρτα.
    5. Με το CMD ορίζουμε ποια υπηρεσία θα τρέξει.

  Με τις εντολές *docker build* και *docker run*, γίνονται αντίστοιχα build το image και run το container.



- **Docker Compose :** Χρησιμοποιείται για την εκτέλεση πολλαπλών containers ως μία υπηρεσία, δηλαδή τη διαμόρφωση και την εκκίνηση τους στον ίδιο υπολογιστή. Αν για παράδειγμα, υποθέσουμε ότι έχουμε μια εφαρμογή που απαιτεί NGINX θα μπορούσαμε να δημιουργήσουμε ένα αρχείο που θα ξεκινούσε και τα δύο containers ως υπηρεσία, χωρίς να χρειάζεται να ξεκινήσουμε το καθένα ξεχωριστά.

- **Docker Swarm :** Είναι μια ομάδα φυσικών ή εικονικών μηχανών που εκτελούν την εφαρμογή Docker και έχουν διαμορφωθεί ώστε να ενώνονται σαν cluster. Το σμήνος είναι ένα εργαλείο που επιτρέπει στον χρήστη να διαχειρίζεται πολλαπλά container που

έχουν αναπτυχθεί σε πολλούς κεντρικούς υπολογιστές. Τα αρχεία YAML μπορούν να χρησιμοποιηθούν για την αναγνώριση πολλών containers.

- **Replicas :** Είναι η υπηρεσία που συνδέεται με το Docker. Πόσες υπηρεσίες θα "σηκώνονται" κάθε φορά.

- **Clusters :** Μια ομάδα υπολογιστών είναι μια συλλογή στενά συνδεδεμένων υπολογιστών που συνεργάζονται έτσι ώστε, με πολλούς τρόπους, να μπορούν να θεωρηθούν ως ένα ενιαίο σύστημα. Σε αντίθεση με τους υπολογιστές δικτύου, οι ομάδες υπολογιστών έχουν κάθε κόμβο καθορισμένο για να εκτελούν την ίδια εργασία, ελέγχονται και προγραμματίζονται από λογισμικό. Η ομαδοποίηση διακομιστών αναφέρεται σε μια ομάδα διακομιστών που συνεργάζονται σε ένα μόνο σύστημα για να παρέχουν στους χρήστες μεγαλύτερη διαθεσιμότητα. Αυτά τα συμπλέγματα χρησιμοποιούνται για τη μείωση του χρόνου διακοπής λειτουργίας και των διακοπών επιτρέποντας σε έναν άλλο διακομιστή να αναλάβει την περίπτωση, αν υπάρξει διακοπή. Μια ομάδα διακομιστών είναι συνδεδεμένη σε ένα μόνο σύστημα.

- **Kubernetes :** Είναι ένα σύστημα ανοιχτού κώδικα για την αυτοματοποίηση της ανάπτυξης, κλιμάκωσης και διαχείρισης εφαρμογών σε κοντέινερ.

Kubernetes vs Docker Swarm

Και τα δύο αυτά εργαλεία μας επιτρέπουν να χειριζόμαστε ένα cluster από servers που εκτελούν μία ή περισσότερες υπηρεσίες σε αυτούς. Το Kubernetes είναι μια open-source πλατφόρμα που δημιουργήθηκε από την Google για την ανάπτυξη των containers. Αυτή η πλατφόρμα μπορεί να χρησιμοποιηθεί για οποιαδήποτε ανάπτυξη αρχιτεκτονικής. Κατανέμει επίσης, το φορτίο μεταξύ των containers. Στόχος του είναι να απαλλάξει τα εργαλεία από το πρόβλημα που αντιμετωπίζει λόγω της εκτέλεσης εφαρμογών σε ιδιωτικά και δημόσια σύννεφα. Η ισχύς του έγκειται στην εύκολη κλιμάκωση και την ευέλικτη ανάπτυξη. Υποστηρίζει υψηλότερες απαιτήσεις με μεγαλύτερη πολυπλοκότητα, ενώ το Docker Swarm προσφέρει μια απλή λύση που μπορεί να ξεκινήσει γρήγορα. Το Docker Swarm ήταν αρκετά δημοφιλές στους προγραμματιστές που προτιμούσαν γρήγορη ανάπτυξη και απλότητα. Ταυτόχρονα, το Kubernetes χρησιμοποιείται σε περιβάλλοντα παραγωγής από διάφορες εταιρείες διαδικτύου υψηλού προφίλ που εκτελούν δημοφιλείς υπηρεσίες. Τόσο το Kubernetes όσο και το Docker Swarm μπορούν να εκτελέσουν πολλές από τις ίδιες υπηρεσίες, αλλά μπορεί να χρειαστούν ελαφρώς διαφορετικές προσεγγίσεις σε ορισμένες λεπτομέρειες. Έτσι, μαθαίνοντας το Kubernetes και το Docker και συγκρίνοντάς τα, μπορούμε να αποφασίσουμε ώστε να επιλέξουμε το σωστό εργαλείο για τα container μας.

Application definition

Kubernetes : Μια εφαρμογή μπορεί να αναπτυχθεί σε Kubernetes χρησιμοποιώντας έναν συνδυασμό υπηρεσιών, αναπτύξεων και pods
Docker Swarm : Οι εφαρμογές μπορούν να αναπτυχθούν ως μικρο-υπηρεσίες ή υπηρεσίες σε ένα swarm cluster στο Docker Swarm. Τα αρχεία yaml μπορούν να χρησιμοποιηθούν για την αναγνώριση πολλών κοντέινερ.

Networking

Kubernetes : Το μοντέλο δικτύωσης είναι ένα επίπεδο δικτύου, που επιτρέπει σε όλα τα pods να αλληλεπιδρούν μεταξύ τους. Οι πολιτικές δικτύου καθορίζουν τον τρόπο αλληλεπίδρασης των ομάδων μεταξύ τους. Το επίπεδο δικτύου εφαρμόζεται συνήθως ως επικάλυψη.
Docker Swarm : Δημιουργεί ένα δίκτυο επικάλυψης για υπηρεσίες που εκτείνονται σε κάθε κεντρικό υπολογιστή στο σμήνος και ένα δίκτυο μόνο για containers. Οι χρήστες έχουν την επιλογή να κρυπτογραφήσουν τα δεδομένα, ενώ δημιουργούν ένα δίκτυο επικάλυψης από μόνα τους στο σμήνος.

Scalability

Kubernetes : Για κατανεμημένα συστήματα, το Kubernetes είναι περισσότερο ένα all-in-one framework. Είναι ένα πολύπλοκο σύστημα, επειδή παρέχει ισχυρές εγγυήσεις σχετικά με την κατάσταση συμπλέγματος και ένα ενοποιημένο σύνολο API. Αυτό επιβραδύνει την κλιμάκωση και την ανάπτυξη των containers.
Docker Swarm : Το Docker Swarm, σε σύγκριση με το Kubernetes, μπορεί να αναπτύξει container γρηγορότερα οπότε επιτρέπει ταχύτερους χρόνους αντίδρασης.

High Availability

Kubernetes : Όλα τα pods στο Kubernetes κατανέμονται μεταξύ κόμβων και αυτό προσφέρει υψηλή διαθεσιμότητα.
Docker Swarm : Καθώς οι υπηρεσίες μπορούν να αναπαραχθούν σε κόμβους Swarm, το Docker Swarm προσφέρει επίσης υψηλή διαθεσιμότητα. Οι κόμβοι στο Docker Swarm είναι υπεύθυνοι για ολόκληρο το cluster και διαχειρίζονται τους πόρους των κόμβων.

Container Setup

Kubernetes : Χρησιμοποιεί τους δικά του yaml και api. Δεν μπορούμε να χρησιμοποιήσετε το Docker Compose για να ορίσουμε container. Κατά την εναλλαγή πλατφορμών, οι ορισμοί και οι εντολές yaml πρέπει να ξανα γραφούν.
Docker Swarm : Δεν περιλαμβάνει πλήρως όλες τις εντολές του Docker, αλλά προσφέρει πολλές από τις οικείες λειτουργίες του. Υποστηρίζει τα περισσότερα εργαλεία που λειτουργούν με το Docker. Ωστόσο, εάν το Docker API είναι ανεπαρκές για μια συγκεκριμένη λειτουργία, δεν υπάρχει ένας εύκολος τρόπος για να το χρησιμοποιήσει το Swarm.

Load Balancing

Kubernetes : Τα pods εκτίθενται μέσω υπηρεσίας, τα οποία μπορούν να χρησιμοποιηθούν ως load balancing εντός του cluster. Γενικά, μια είσοδος χρησιμοποιείται για load balancing.
Docker Swarm : Αποτελείται από ένα στοιχείο DNS που μπορεί να χρησιμοποιηθεί για τη διανομή εισερχόμενων αιτημάτων σε ένα όνομα υπηρεσίας. Οι υπηρεσίες μπορούν να εκχωρηθούν αυτόματα ή να εκτελούνται σε θύρες που καθορίζονται από τον χρήστη.

GlusterFS

GlusterFs είναι ένα επεκτάσιμο, κατανεμημένο σύστημα αρχείων δικτύου κατάλληλο για εργασίες υψηλής έντασης δεδομένων, όπως αποθήκευση cloud και ροή πολυμέσων, που συγκεντρώνει πόρους αποθήκευσης δίσκων από πολλούς διακομιστές σε έναν ενιαίο χώρο ονομάτων.

Πλεονεκτήματα :

- Καινοτομία - Εξαλείφει τα μεταδεδομένα και μπορεί δραματικά να βελτιώσει την απόδοση που θα μας βοηθήσει να ενοποιήσουμε δεδομένα και αντικείμενα.
- Ελαστικότητα - Προσαρμοσμένη στην ανάπτυξη και τη μείωση του μεγέθους των δεδομένων.
- Γραμμική κλίμακα - Διατίθεται σε petabytes και πέραν αυτής.
- Απλότητα - Είναι εύκολο στη διαχείριση και ανεξάρτητο από τον πυρήνα ενώ εκτελείται σε χώρο χρήστη.
- Πωλήσιμο- Η απουσία διακομιστή μεταδεδομένων παρέχει ένα γρηγορότερο σύστημα αρχείων.
- Προσιτό - Εφαρμόζεται σε υλικό εμπορευμάτων.

- Ευέλικτο -Το GlusterFS είναι ένα σύστημα αρχείων μόνο λογισμικού . Εδώ τα δεδομένα αποθηκεύονται σε εγγενή συστήματα αρχείων όπως ext4, xfs κ.λπ.
- Ανοιχτό λογισμικό - Αυτή τη στιγμή το GlusterFS διατηρείται από την Red Hat Inc, μια εταιρεία ανοιχτού κώδικα δισεκατομμυρίων δολαρίων, ως μέρος της Red Hat Storage.

Ανάλυση Αρχείων

Dockerfile - image

Dockerfile: Σύνολο εντολών ώστε να δημιουργήσουμε την εικόνα. Παίρνει εξωτερικά αρχεία και τα προσθέτει στο container (tensoflow, packages.r). Επίσης, στο conda κάνει install τις γλώσσες προγραμματισμού που θα χρησιμοποιήσουμε, ορίζουμε την εσωτερική πόρτα του container και ποια υπηρεσία θα τρέξει.
Dockerfile.build: Κάνει build το Dockerfile.
Dockerfile.run: Κάνει Build το Image.

Packages.r : Πακέτα για τη γλώσσα R
Tensorflow.yml : Σύστημα εργαλείων, βιβλιοθηκών και πόρων στην Τεχνητή Νοημοσύνη.

gluster

Gluster - manager-install.sh:
Τρέχουν Υπηρεσίες, δημιουργούνται τα volumes δηλαδή χώρους στον σκληρό δίσκο. Επίσης ο manager τρέχει και το script με όνομα *Gluster - gluster-manager-commands.sh*

Gluster - worker-install.sh:
Συνδέεται στον server και τρέχει το script *Gluster - gluster-worker-commands.sh*

Swarm

Swarm.init : Εντολή που τρέχει ο manager για να πάρει το token.

Docker-compose

Docker-compose.yaml:Περιέχεται τα servers και δηλώνουμε τα χαρακτηριστικά τους name, image, volumes, replica, limits.

Docker-compose.yaml.run : Κανει upload το σμηνος , από αυτό τρέχουν όλα .

Conf

Jupyter-notebook.conf: Δηλώνονται οι server που θα χρησιμοποιήσουμε ,σύμφωνα με το least_conn επιλέγετε ο server με τις λιγότερες συνδέσεις. Επίσης, περνάμε το δημόσιο,ιδιωτικό κλειδί, πρωτόκολλο και τους αλγορίθμους ssl.

Key : Κλειδί που φτιάξαμε για το ssl πρωτόκολλο

Nginx.conf: Πλατφόρμα για το web server. Επίσης βάλαμε κώδικα έτσι ώστε ο nginx να επιτρέπει τα web-socket .

Openssl.conf : Περιγράφει την πιστοποίηση.

Private.key: Αυτόματη δημιουργία ιδιωτικού κλειδιού απο το key, secret.

Public.crt: Αυτόματη δημιουργία δημόσιου κλειδιού απο το key, secret.

Secret : Κλειδί που φτιάξαμε για το ssl πρωτόκολλο

Βήματα Εγκατάστασης

- Εγκατάσταση υπηρεσίας Docker σε όλους τους υπολογιστές (manager/workers).
- Εγκατάσταση υπηρεσίας Glusterfs σε όλους τους υπολογιστές για διαμοιρασμένη μνήμη. (https://thenewstack.io/tutorial-create-a-docker-swarm-with-persistent-storage-using-glusterfs/)
- Εκτέλεση των παρακάτω εντολών:
  Make
  Σημείωση: κατά την εγκατάσταση μέσω του αρχείου Makefile γίνεται εγκατάσταση του manager κόμβου και εμφανίζεται το κλειδί του worker όπου και το σκριπτ περιμένει μέχρις ότου ενταχθούν όλοι οι επιθυμητοί κόμβοι στο σμήνος.
  εναλλακτικά για custom build της εικόνας:
  make build_dockerfile
  make run_dockerfile

Τεχνική Περιγραφή

Γενικά :

Εφόσον αποφασίσαμε το θέμα και το σκοπό της εργασίας μας, αποφασίσαμε ποιοί κόμβοι θα είναι workers και ποιοι manager. Εφόσον εγκαταστήσαμε το Docker σε όλους τούς κόμβους αποφασίσαμε να χρησιμοποιήσουμε το glusterfs για να έχουμε

έναν κοινό διαμοιρασμένο χώρο σε όλους τους κόμβους έτσι ώστε να μπορέσουμε να χρησιμοποιήσουμε όλους τούς κόμβους για να σηκώσουμε την ίδια υπηρεσία μας.
Έπειτα απο τα παραπάνω, ξεκινήσαμε να υλοποιούμε το Dockerfile μας το οποίο χτίσαμε βήμα-βήμα. Εφόσον ολοκληρώσαμε και τη διαδικασία με το Dockerfile το ανεβάσαμε στην πλατφόρμα Docker Hub όπου μπορούν να βρεθούν πολλές εικόνες
προς εγκαταστάση σύμφωνα με τις ανάγκες μας. Δημιουργήθηκαν τα docker-compose.yml και παραμετροποιήθηκε ο nginx έτσι ώστε να σερβίρει την υπηρεσία μας.

Dockerfile :
Jupyter Notebook Image με προεγκατεστημένες τις γλώσσες R, Python, Java, Matlab, Tensorflow Enviroment.

Docker-compose.yaml:
Το σύνολο όλων των υπηρεσιών καθώς και η παραμετροποίηση τους έτσι ώστε να εγκαταστηθούν σε κάθε υπολογιστή που έχουμε ορίσει. Ακόμα περιγράφεται και ορίζεται το πόση μνήμη μπορεί να χρησιμοποιηθεί απο το κάθε container καθώς και τα πόσα Instances θα δημιουργηθούν (replicas).

Nginx configuration:
Η λειτουργία του Nginx είναι καθαρά σαν Proxy. Ο Nginx Webserver είναι υπεύθυνος για την ανακατεύθυνση των ερωτημάτων στους Workers καθώς και το "σερβίρισμα" των απαντήσεων απο τους Workers στον Client. Ο Server που θα κληθεί να εξυπηρετήσει το ερώτημα θα είναι αυτός
ο οποίος έχει τις λιγότερες συνδέσεις εκείνη τη χρονική στιγμή. Επίσης, με την ίδια ακριβώς λογική θα ανακατευθύνει και τα αιτήματα των websockets που απαιτεί η υπηρεσίας μας για την εκτέλεση του κώδικα.

Παραδείγματα

- **Java :**

```
In [1]: System.out.println("hello")

hello
```

```
In [5]: public class MyClass {
          static void myMethod() {
            System.out.println("Hello World!");
          }
        }
```

```
In [6]: MyClass mc = new MyClass();
        mc.myMethod();

Hello World!
```

- **PHP :**

```
In [5]: echo "hello world"
Out[5]: hello world
```

```
In [9]: for ($x = 0; $x <= 10; $x++) {
          echo "The number is: $x";
        }
Out[9]: The number is: 0
Out[9]: The number is: 1
Out[9]: The number is: 2
Out[9]: The number is: 3
Out[9]: The number is: 4
Out[9]: The number is: 5
Out[9]: The number is: 6
Out[9]: The number is: 7
Out[9]: The number is: 8
Out[9]: The number is: 9
Out[9]: The number is: 10
```

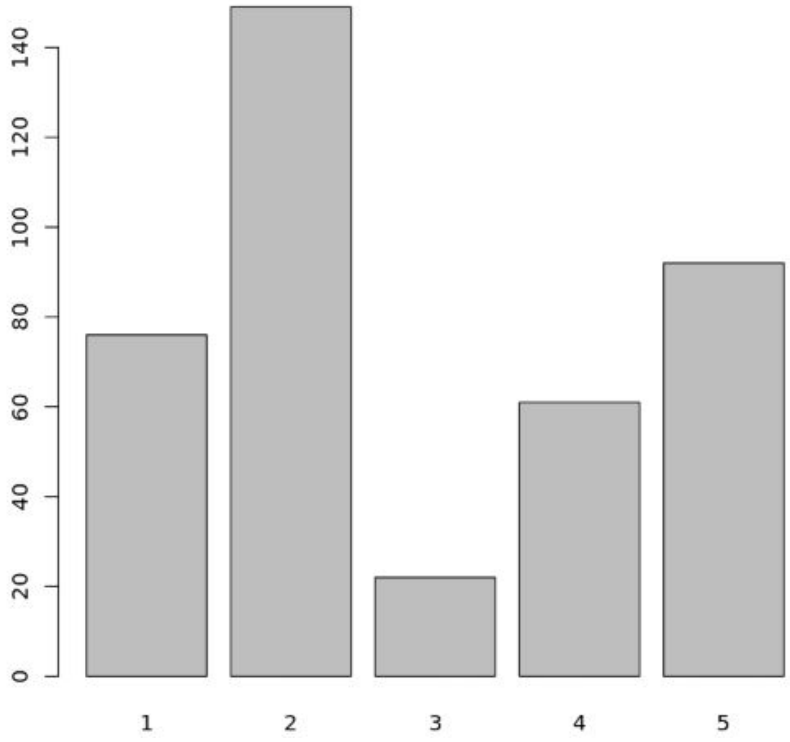
- **Python 3**

```
In [7]: print("hello world");

        hello world
```

```
In [12]: class Awesome():
             def greet(self, txt):
                 print(txt)

In [13]: awes = Awesome()

In [14]: awes.greet("hello from python 3")

         hello from python 3
```

- **R :**

```
In [3]: barplot(table(sample(1:5, size=400, replace=TRUE, prob=c(.30,.60,.10,.20,.30))))
```



- **Python 3.7 (Tensorflow) :**

In [1]:
```
print("hello world")
```

```
hello world
```

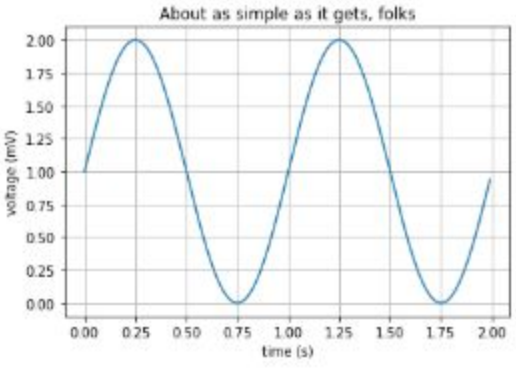
In [2]:
```
import tensorflow as tf
```

In [3]:
```
x = tf.Variable(3.0)
x2 = tf.Variable(3.0)
with tf.GradientTape(persistent=True) as g:
    y = x * x
    y2 = y * x2

dy_dx = g.gradient(y, x)       # 6
dy2_dx2 = g.gradient(y2, x2)   # 9
del g  # Drop the reference to the tape
print(dy_dx)
print(dy2_dx2)
```

```
tf.Tensor(6.0, shape=(), dtype=float32)
tf.Tensor(9.0, shape=(), dtype=float32)
```

In [2]:
```
import matplotlib.pyplot as plt
import numpy as np
```

In [5]:
```
t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin(2*np.pi*t)
plt.plot(t, s)

plt.xlabel('time (s)')
plt.ylabel('voltage (mV)')
plt.title('About as simple as it gets, folks')
plt.grid(True)
plt.savefig("ai-plot-example.png")
plt.show()
```



- **NodeJS :**

Βιβλιογραφία

- AsciiDocSyntaxQuickReference
- Asciidoc Writers guide
- User-manual-asciidoctor
- How to asciidoc/install
- DockerSwarm
- GlusterFS Documentation

- Kubernetes vs docker
- What is a Docker
- Docker Compose
- Makefile