Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Εργαστήριο Υπολογιστικής Νέφους και Υπηρεσιών

Project Εργαστηρίου 2023

Μέλη Ομάδας

- Χαράλαμπος Παπαδόπουλος | **cs141184**
- Κουτσοδήμος Μελέτιος | **ice18390168**

Εισαγωγή

Αυτό το project δημιουργήθηκε για το εργαστήριο υπολογιστικής νέφους και παρέχει ένα εικονικό εργαστηριακό περιβάλλον χρησιμοποιώντας Docker. Προσφέρει μια σειρά από λειτουργίες και υπηρεσίες για τη διευκόλυνση της εύκολης ανάπτυξης και διαχείρισης διαφόρων εφαρμογών. Το παρών README αρχείο παρέχει μια επισκόπηση του project, συμπεριλαμβανομένων των χαρακτηριστικών του και των υπηρεσιών που χρησιμοποιούνται.

Χαρακτηριστικά

Το Docker project περιλαμβάνει τα ακόλουθα χαρακτηριστικά:

- Κάθε εικονικό εργαστήριο που δημιουργείται έχει το δικό του δίκτυο, επιτρέποντας στα container να επικοινωνούν με ασφάλεια μεταξύ τους.
- Προσφέρει μια γραφική διεπαφή χρήστη (GUI) και μια διεπαφή γραμμής εντολών (CLI) για την προβολή των αποτελεσμάτων των deployed υπηρεσιών.

- Παρέχει δυνατότητες αποθήκευσης για τη διαρκή διαχείριση δεδομένων. Αυτό διασφαλίζει ότι τα δεδομένα διατηρούνται ακόμη και όταν τα container σταματούν ή επανεκκινούνται.
- Αυτόματη εκτέλεση και τερματισμός υπηρεσιών με τη χρήση ενός makefile χωρίς χειροκίνητη παρέμβαση.
- Κάθε υπηρεσία εκτελείται σε ξεχωριστό container. Το WordPress χρησιμοποιείται ως η κύρια σελίδα, ενώ η MySQL χειρίζεται τη λειτουργικότητα της βάσης δεδομένων. Το phpMyAdmin παρέχει μια φιλική προς το χρήστη διεπαφή για τη διαχείριση της βάσης δεδομένων MySQL και το Portainer προσφέρει δυνατότητες διαχείρισης και παρακολούθησης των πόρων του κάθε container.
- Περιλαμβάνει διαμόρφωση πόρων για τη βελτιστοποίηση της κατανομής πόρων και την αποτροπή εξαιρέσεων εκτός μνήμης (OOME). Τα αρχεία Docker Compose καθορίζουν όρια πόρων για όλα τα container που χρησιμοποιούμε.

Υπηρεσίες που χρησιμοποιούνται

Το Docker project χρησιμοποιεί τις ακόλουθες υπηρεσίες:

- WordPress: Η υπηρεσία WordPress είναι ένα δημοφιλές σύστημα διαχείρισης περιεχομένου που χρησιμοποιείται για τη δημιουργία και τη διαχείριση ιστοσελίδων. Παρέχει μια φιλική προς τον χρήστη διεπαφή για διαχείριση ιστοτόπων και δημιουργία κειμένου.
- MySQL: Η MySQL είναι ένα ευρέως χρησιμοποιούμενο σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων ανοιχτού κώδικα. Παρέχει μια ισχυρή και επεκτάσιμη λειτουργία για την αποθήκευση και τη διαχείριση δεδομένων.
- phpMyAdmin: Το phpMyAdmin είναι μια γραφική διεπαφή για τη διαχείριση βάσεων δεδομένων MySQL. Απλοποιεί τη διαδικασία διαχείρισης της βάσης δεδομένων και επιτρέπει την εύκολη διαχείριση των λειτουργιών της βάσης δεδομένων.
- Portainer: Το Portainer είναι μια ελαφριά διεπαφή διαχείρισης για το Docker. Παρέχει μια φιλική προς το χρήστη διεπαφή για την παρακολούθηση και τη διαχείριση container, images, δικτύων και volumes Docker.

Οι παραπάνω υπηρεσίες συνεργάζονται για να δημιουργήσουν ένα ολοκληρωμένο περιβάλλον ανάπτυξης ιστού. Στις επόμενες ενότητες, θα παρέχουμε λεπτομερείς πληροφορίες για κάθε container, συμπεριλαμβανομένων του κώδικα και των config αρχείων που χρησιμοποιούνται. Θα συζητήσουμε επίσης το makefile που χρησιμοποιείται και θα παρέχουμε οδηγίες για την εγκατάσταση και τη ρύθμιση.

WordPress Container

Το WordPress container χρησιμεύει ως η κύρια σελίδα για το project. Παρέχει ένα σύστημα διαχείρισης περιεχομένου που επιτρέπει στους χρήστες να δημιουργούν και να διαχειρίζονται ιστοσελίδες με ευκολία.

Σκοπός και Λειτουργικότητα

Το container WordPress προσφέρει τις ακόλουθες δυνατότητες:

- Δημιουργία και διαχείριση ιστότοπου: Το WordPress επιτρέπει στους χρήστες να δημιουργούν, να προσαρμόζουν και να διαχειρίζονται εύκολα ιστότοπους.
- Δημοσίευση περιεχομένου: Με το WordPress, οι χρήστες μπορούν να δημιουργούν και να δημοσιεύουν αναρτήσεις ιστολογίου, άρθρα, και άλλους τύπους περιεχομένου ιστού με απλό τρόπο. Προσφέρει οπτικό πρόγραμμα επεξεργασίας και υποστηρίζει διάφορες μορφές πολυμέσων.
- Διαχείριση χρηστών: Το WordPress παρέχει δυνατότητες διαχείρισης χρηστών, επιτρέποντας στους διαχειριστές να δημιουργούν λογαριασμούς χρηστών, να έχουν ρόλους και δικαιώματα και να ελέγχουν την πρόσβαση σε διαφορετικές ενότητες του ιστότοπου.

Configuration

Για να υλοποιήσουμε το container WordPress, χρησιμοποιούμε δύο αρχεία Docker Compose: docker-compose.yaml και docker-compose.override.yaml. Ας τα εξηγήσουμε:

docker-compose.override.yaml

```
version: '3.4'

services:
    wordpress:
        container_name: wordpress
        ports:
         - '8080:80'
        restart: always
        volumes:
         - '/wp_data:/var/www/html'
        environment:
         WORDPRESS_DB_HOST: db
         WORDPRESS_DB_USER: wordpress
         WORDPRESS_DB_PASSWORD: apostolos
         WORDPRESS_DB_NAME: wordpress
        deploy:
          resources:
            limits:
              cpus: '0.10'
              memory: '500M'
            reservations:
              cpus: '0.05'
              memory: '50M'
        networks:
          - wordpressNetwork

volumes:
    data:
      driver: local
```

- container_name: Καθορίζουμε το όνομα του container του WordPress ως "wordpress".
- ports: Αντιστοιχίζουμε το port 8080 του κεντρικού υπολογιστή στη θύρα 80 του container, επιτρέποντας την πρόσβαση στο WordPress μέσω του http://localhost:8080.
- restart: Εξασφαλίζουμε ότι το container κάνει επαννεκίνηση αυτόματα εάν σταματήσει ή αντιμετωπίσει κάποιο σφάλμα.
- volumes: Δημιουργούμε ένα volume με το όνομα "/wp_data" και τον κάνουμε mount στον κατάλογο "/var/www/html" του container. Αυτό επιτρέπει στα αρχεία WordPress να αποθηκεύονται εκτός του container, διατηρώντας τα δεδομένα ακόμα και αν το container έχει σταματήσει ή αφαιρεθεί.
- environment: Με τα environment variables καθορίζουμε την σύνδεση του WordPress με το container MySQL.
- deploy: Ορίζουμε τα όρια πόρων για το container, συμπεριλαμβανομένων των περιορισμών της CPU και της μνήμης.
- network: Συνδέουμε το container του WordPress με το δίκτυο "wordpressNetwork".

docker-compose.yaml

```
version: '3.4'

services:

  wordpress:
    image: wordpress:latest


networks:
    wordpressNetwork:
        name: wordpressNetwork
        driver: bridge
```

Σε αυτό το αρχείο, ορίζουμε την υπηρεσία WordPress χρησιμοποιώντας το WordPress Docker Image, που γίνεται pull από το Docker Hub με το πιο πρόσφατο tag, αν το image δεν υπάρχει ήδη στον υπολογιστή.

Με το "networks" συνδέουμε το container του WordPress με το δίκτυο "wordpressNetwork". Ορίζουμε το όνομα δικτύου ξεχωριστά στην ενότητα networks, προσδιορίζοντας το όνομά του ως "wordpressNetwork" και το driver ως "bridge".

Μαζί, αυτά τα αρχεία Docker Compose διαμορφώνουν και αναπτύσσουν το container του WordPress, διασφαλίζοντας ότι είναι συνδεδεμένο στο καθορισμένο δίκτυο και έχει πρόσβαση στους απαραίτητους πόρους και volumes.

MySQL Container

Το container MySQL έχει τη βάση δεδομένων MySQL που χρησιμοποιείται από την εφαρμογή WordPress. Εξασφαλίζει αξιόπιστη αποθήκευση και ανάκτηση δεδομένων για τον ιστότοπο. Ακολουθεί εξήγηση της διαμόρφωσης και της λειτουργικότητάς του.

Σκοπός και Λειτουργικότητα

Το container MySQL προσφέρει τις ακόλουθες δυνατότητες:

- Αποθήκευση και διαχείριση δεδομένων: Η MySQL χρησιμεύει ως μια αποτελεσματική λύση βάσης δεδομένων για την αποθήκευση δεδομένων ιστότοπου. Εξασφαλίζει ακεραιότητα δεδομένων, αξιοπιστία και πρόσβαση υψηλής απόδοσης για το WordPress.
- Αποθήκευση δομημένων δεδομένων: Η MySQL οργανώνει δεδομένα σε tables, επιτρέποντας την αποτελεσματική αποθήκευση και ανάκτηση δομημένων πληροφοριών.
- Ασφαλές: Η MySQL παρέχει προηγμένες δυνατότητες ασφαλείας, όπως έλεγχο ταυτότητας χρήστη, έλεγχος πρόσβασης και κρυπτογράφηση ευαίσθητων δεδομένων.
- Διαχείριση βάσεων δεδομένων: Η MySQL προσφέρει εργαλεία και βοηθητικά προγράμματα για τη διαχείριση βάσεων δεδομένων, επιτρέποντας στους διαχειριστές να διαχειρίζονται βάσεις δεδομένων, πίνακες, ευρετήρια και αντίγραφα ασφαλείας. Παρέχει μια διεπαφή γραμμής

εντολών (CLI) και διάφορα γραφικά εργαλεία για εύκολη διαχείριση.

Configuration

Για να υλοποιήσουμε το container mySQL, χρησιμοποιούμε δύο αρχεία Docker Compose: docker-compose.yaml και docker-compose.override.yaml. Ας τα εξηγήσουμε:

docker-compose.override.yaml

```
version: '3.4'

services:
    db:
        container_name: sqlDatabase
        ports:
         - '3306:3306'
        restart: always
        volumes:
         - /db_data:/var/lib/mysql
        environment:
         MYSQL_ROOT_PASSWORD: apostolos
         MYSQL_DATABASE: wordpress
         MYSQL_USER: wordpress
         MYSQL_PASSWORD: apostolos

        deploy:
         resources:
            limits:
              cpus: '0.10'
              memory: 500M
            reservations:
              cpus: '0.05'
              memory: 50M

        networks:
          - wordpressNetwork

volumes:
    data:
        driver: local
```

- container_name: Ορίζουμε το όνομα του container ως sqlDatabase.
- ports: Αντιστοιχίζουμε το port κεντρικού υπολογιστή 3306 στο port container 3306, επιτρέποντας την πρόσβαση στο mySQL GUI μέσω http://localhost:3306.
- restart: Καθορίζουμε ότι το container πρέπει πάντα να επανεκκινείται, διασφαλίζοντας τη διαθεσιμότητά του.
- volumes: Κάνουμε mount τον κατάλογο κεντρικού υπολογιστή "/db_data" στον κατάλογο "/var/lib/mysql" του container, παρέχοντας μόνιμο χώρο αποθήκευσης για τη βάση δεδομένων

MySQL.

- environment: Ορίζουμε μεταβλητές περιβάλλοντος για τη διαμόρφωση της MySQL, συμπεριλαμβανομένων του κωδικού πρόσβασης root, του ονόματος της βάσης δεδομένων κλπ.
- deploy: Ορίζουμε τα όρια πόρων για χρήση CPU και μνήμης για βελτιστοποίηση της απόδοσης.
- networks: Συνδέουμε το container στο δίκτυο wordpressNetwork, επιτρέποντας την επικοινωνία με άλλα container.

docker-compose.yaml

```
version: '3.4'

services:

  db:
    image: mysql:5.7


networks:
    wordpressNetwork:
        driver: bridge
        external: true
```

Σε αυτό το αρχείο, ορίζουμε την υπηρεσία mySQL χρησιμοποιώντας το Docker Image, που γίνεται pull από το Docker Hub με το πιο πρόσφατο tag, αν το image δεν υπάρχει ήδη στον υπολογιστή.

Με το "networks" συνδέουμε το container του mySQL με το δίκτυο "wordpressNetwork". Ορίζουμε το όνομα δικτύου ξεχωριστά στην ενότητα networks, προσδιορίζοντας το όνομά του ως "wordpressNetwork" και το driver ως "bridge" και έτσι εποικινωνεί με τα υπόλοιπα container.

Μαζί, αυτά τα αρχεία Docker Compose διαμορφώνουν και αναπτύσσουν το container της mySQL, διασφαλίζοντας ότι είναι συνδεδεμένο στο καθορισμένο δίκτυο και έχει πρόσβαση στους απαραίτητους πόρους και volumes.

phpMyAdmin Container

Το container phpMyAdmin είναι ένα βασικό συστατικό του project, παρέχοντας ένα γραφικό περιβάλλον χρήστη (GUI) που βασίζεται στο web για τη διαχείριση βάσεων δεδομένων MySQL, απλοποιεί τη διαχείριση και την αλληλεπίδραση με τη βάση δεδομένων MySQL.

Σκοπός και Λειτουργικότητα

Το container phpMyAdmin προσφέρει τις ακόλουθες δυνατότητες:

- GUI για διαχείριση βάσεων δεδομένων MySQL: Το phpMyAdmin παρέχει μια διαισθητική διεπαφή βασισμένη στον ιστό για τη διαχείριση βάσεων δεδομένων MySQL. Επιτρέπει στους χρήστες να εκτελούν διάφορες λειτουργίες, όπως τη δημιουργία βάσεων δεδομένων, τη διαχείριση πινάκων, κλπ.

- Αποτελεσματική διαχείριση βάσεων δεδομένων: Με το phpMyAdmin, οι χρήστες μπορούν εύκολα να προβάλουν και να επεξεργαστούν τη δομή της βάσης δεδομένων, να εισάγουν και να εξάγουν δεδομένα, να βελτιστοποιήσουν την απόδοση της βάσης δεδομένων και άλλα.

Configuration

Για να υλοποιήσουμε το container phpMyAdmin, χρησιμοποιούμε δύο αρχεία Docker Compose: docker-compose.yaml και docker-compose.override.yaml. Ας τα εξηγήσουμε:

docker-compose.override.yaml

```
version: '3.4'

services:
    phpmyadmin:
        container_name: phpMyAdming
        ports:
         - '8888:80'
        restart: always
        environment:
          PMA_HOST: db
          PMA_USER: wordpress
          PMA_PASSWORD: apostolos
        deploy:
          resources:
            limits:
              cpus: '0.10'
              memory: 200M
            reservations:
              cpus: '0.05'
              memory: 50M

        networks:
          - wordpressNetwork
volumes:
    data:
      driver: local
```

- container_name: Ορίζουμε το όνομα του container ως phpMyAdming.
- ports: Αντιστοιχίζουμε στο port 8888 στον κεντρικό υπολογιστή, επιτρέποντας την πρόσβαση στο phpMyAdmin GUI μέσω http://localhost:8888.
- restart: Καθορίζουμε ότι το container πρέπει πάντα να επανεκκινείται, διασφαλίζοντας τη διαθεσιμότητά του.
- environment: Ορίζουμε μεταβλητές περιβάλλοντος για τη διαμόρφωση του phpMyAdmin, συμπεριλαμβανομένων του κωδικού πρόσβασης root, του ονόματος της βάσης δεδομένων κλπ.
- deploy: Ορίζουμε τα όρια πόρων για χρήση CPU και μνήμης για βελτιστοποίηση της απόδοσης.
- networks: Συνδέουμε το container στο δίκτυο wordpressNetwork, επιτρέποντας την

επικοινωνία με άλλα container.

docker-compose.yaml

```
version: '3.4'

services:

  phpmyadmin:
    image: phpmyadmin/phpmyadmin


networks:
    wordpressNetwork:
        driver: bridge
        external: true
```

Σε αυτό το αρχείο, ορίζουμε την υπηρεσία phpMyAdmin χρησιμοποιώντας το Docker Image, που γίνεται pull από το Docker Hub με το πιο πρόσφατο tag, αν το image δεν υπάρχει ήδη στον υπολογιστή.

Με το "networks" συνδέουμε το container με το δίκτυο "wordpressNetwork". Ορίζουμε το όνομα δικτύου ξεχωριστά στην ενότητα networks, προσδιορίζοντας το όνομά του ως "wordpressNetwork" και το driver ως "bridge" και έτσι εποικινωνεί με τα υπόλοιπα container.

Μαζί, αυτά τα αρχεία Docker Compose διαμορφώνουν και αναπτύσσουν το container του phpMyAdmin, διασφαλίζοντας ότι είναι συνδεδεμένο στο καθορισμένο δίκτυο και έχει πρόσβαση στους απαραίτητους πόρους και volumes.

Portainer Container

Το container Portainer είναι ένα ουσιαστικό στοιχείο του project, παρέχοντας μια φιλική προς το χρήστη γραφική διεπαφή για τη διαχείριση και την παρακολούθηση container και πόρων Docker.

Σκοπός και Λειτουργικότητα

Το container Portainer προσφέρει τις ακόλουθες δυνατότητες:

- Διαχείριση container: Το Portainer επιτρέπει στους χρήστες να διαχειρίζονται και να ελέγχουν τα container Docker μέσω GUI. Οι χρήστες μπορούν να δουν, να ξεκινήσουν, να σταματήσουν και να επανεκκινήσουν τα container, καθώς και να παρακολουθήσουν τη χρήση των πόρων και την κατάσταση της υγείας τους.
- Παρακολούθηση πόρων: Το Portainer παρέχει παρακολούθηση σε πραγματικό χρόνο της χρήσης πόρων του container, συμπεριλαμβανομένης της CPU, της μνήμης και της χρήσης δικτύου.
- Διαχείριση χρηστών και πρόσβασης: Το Portainer προσφέρει δυνατότητες διαχείρισης χρήστη και πρόσβασης, επιτρέποντας στους admins να δημιουργούν λογαριασμούς χρηστών, να εκχωρούν ρόλους και δικαιώματα και να ελέγχουν την πρόσβαση στους πόρους του Docker.

Ενισχύει την ασφάλεια και διευκολύνει τη συνεργασία μεταξύ των μελών της ομάδας.

- Δικτύωση container και volumes: Το Portainer παρέχει ένα GUI για τη διαχείριση δικτύων container και volumes. Οι χρήστες μπορούν να δημιουργούν και να διαχειρίζονται δίκτυα για να επιτρέπουν την επικοινωνία μεταξύ container και να διαμορφώνουν data volumes για μόνιμη αποθήκευση.

Configuration

Για να υλοποιήσουμε το container Portainer, χρησιμοποιούμε δύο αρχεία Docker Compose: docker-compose.yaml και docker-compose.override.yaml. Ας τα εξηγήσουμε:

docker-compose.override.yaml

```
version: '3.4'

services:
    portainer:
        container_name: portainer
        ports:
         - '9443:9443'
        restart: always
        volumes:
         - data:/data
         - /var/run/docker.sock:/var/run/docker.sock
        deploy:
          resources:
            limits:
              cpus: '0.10'
              memory: 50M
            reservations:
              cpus: '0.05'
              memory: 50M
        networks:
          - wordpressNetwork
volumes:
    data:
      driver: local
```

- container_name: Ορίζουμε το όνομα του container ως "portainer".
- ports: Αντιστοιχίζουμε το port 9443 του κεντρικού υπολογιστή στο port 9443 του container, επιτρέποντας την πρόσβαση στο GUI του Portainer. Με την πρόσβαση στη διεύθυνση IP του κεντρικού υπολογιστή με το port 9443, https://localhost:9443), μπορούμε να έχουμε πρόσβαση στο Portainer με ασφάλεια μέσω HTTPS.
- restart: Το πεδίο restart διασφαλίζει ότι το Portainer επανεκκινείται αυτόματα εάν σταματήσει ή αντιμετωπίσει κάποιο σφάλμα.
- volumes: Ορίζουμε volumes για μόνιμη αποθήκευση δεδομένων. Σε αυτήν την περίπτωση, αντιστοιχίζουμε το volume "data" σε "/data" εντός του container και επίσης κάνουμε mount στο

Docker socket ("/var/run/docker.sock") για να ενεργοποιήσουμε την επικοινωνία με το Docker daemon του host.

- deploy: Ορίζουμε τα όρια πόρων για το container. Σε αυτό το παράδειγμα, ορίσαμε όρια CPU σε 0,10 και κρατήσεις CPU σε 0,05 κλπ.
- networks: Συνδέουμε το container του Portainer με το δίκτυο "wordpressNetwork", επιτρέποντάς του να επικοινωνεί με άλλα container εντός του ίδιου δικτύου.

docker-compose.yaml

```
version: '3.4'

services:

  portainer:
    image: portainer/portainer-ce:latest


networks:
    wordpressNetwork:
        driver: bridge
        external: true
```

Σε αυτό το αρχείο, ορίζουμε την υπηρεσία Portainer χρησιμοποιώντας το Docker Image, που γίνεται pull από το Docker Hub με το πιο πρόσφατο tag, αν το image δεν υπάρχει ήδη στον υπολογιστή.

Με το "networks" συνδέουμε το container με το δίκτυο "wordpressNetwork". Ορίζουμε το όνομα δικτύου ξεχωριστά στην ενότητα networks, προσδιορίζοντας το όνομά του ως "wordpressNetwork" και το driver ως "bridge" και έτσι εποικινωνεί με τα υπόλοιπα container.

Μαζί, αυτά τα αρχεία Docker Compose διαμορφώνουν και αναπτύσσουν το container του Portainer, διασφαλίζοντας ότι είναι συνδεδεμένο στο καθορισμένο δίκτυο και έχει πρόσβαση στους απαραίτητους πόρους και volumes.

Makefile και Installation

Με το Makefile του project, έχουμε απλοποίησει την εγκατάσταση, τη λειτουργία και τη διακοπή των container Docker ώστε να γίνεται αυτόματα και όχι manually. Πιο αναλυτικά:

makefile

```
NETWORK_NAME := wordpressNetwork

SERVICE1_DIR := ../wordpress
SERVICE2_DIR := ../mysql
SERVICE3_DIR := ../phpmydamin
SERVICE4_DIR := ../portainer

COMPOSE_FILE := docker-compose.yaml
COMPOSE_FILE_OVERRIDE := docker-compose.override.yaml

.PHONY: start-project

start-project:
    @docker network create $(NETWORK_NAME)
    @cd $(SERVICE1_DIR) && docker-compose -f $(COMPOSE_FILE) -f
$(COMPOSE_FILE_OVERRIDE) up -d
    @cd $(SERVICE2_DIR) && docker-compose -f $(COMPOSE_FILE) -f
$(COMPOSE_FILE_OVERRIDE) up -d
    @cd $(SERVICE3_DIR) && docker-compose -f $(COMPOSE_FILE) -f
$(COMPOSE_FILE_OVERRIDE) up -d
    @cd $(SERVICE4_DIR) && docker-compose -f $(COMPOSE_FILE) -f
$(COMPOSE_FILE_OVERRIDE) up -d

.PHONY: stop-project

stop-project:
    @cd $(SERVICE2_DIR) && docker-compose -f $(COMPOSE_FILE) -f
$(COMPOSE_FILE_OVERRIDE) down
    @cd $(SERVICE3_DIR) && docker-compose -f $(COMPOSE_FILE) -f
$(COMPOSE_FILE_OVERRIDE) down
    @cd $(SERVICE4_DIR) && docker-compose -f $(COMPOSE_FILE) -f
$(COMPOSE_FILE_OVERRIDE) down
    @cd $(SERVICE1_DIR) && docker-compose -f $(COMPOSE_FILE) -f
$(COMPOSE_FILE_OVERRIDE) down
```

- NETWORK_NAME: Αυτή η μεταβλητή αποθηκεύει το όνομα του δικτύου που θα δημιουργηθεί για τη σύνδεση των container.
- SERVICE1_DIR, SERVICE2_DIR, SERVICE3_DIR, SERVICE4_DIR: Αυτές οι μεταβλητές αποθηκεύουν τα paths προς τα directories όπου βρίσκονται τα αρχεία Docker Compose της αντίστοιχης υπηρεσίας.
- COMPOSE_FILE, COMPOSE_FILE_OVERRIDE: Αυτές οι μεταβλητές καθορίζουν τα ονόματα του κύριου αρχείου Docker Compose και του αρχείου override, τα οποία χρησιμοποιούνται για τη διαμόρφωση και την εκκίνηση των υπηρεσιών.
- start-project: Δημιουργεί το δίκτυο, μετά πηγαίνει στον κατάλογο κάθε υπηρεσίας και εκτελεί την εντολή Docker Compose για να τρέξει τα container. Το flag -f χρησιμοποιείται για τον καθορισμό των αρχείων Docker Compose που θα χρησιμοποιηθούν. Έτσι συνδέουμε όλα τα

container μαζί στο ίδιο δίκτυο.

- stop-project: Σταματά και αφαιρεί τα container. Πηγαίνει στο directory κάθε υπηρεσίας και χρησιμοποιεί την εντολή Docker Compose με τα κατάλληλα αρχεία για να κατεβάσει τα container.

Για να εγκαταστήσετε και να εκτελέσετε το project, ακολουθήστε τα εξής βήματα:

Προαπαιτούμενα

- Βεβαιωθείτε ότι το Docker και το Docker Compose είναι εγκατεστημένα στο σύστημά σας.
- Κάντε clone του project repository στον υπολογιστή σας.

Step 1: Start the Project

- Ανοίξτε ένα τερματικό.
- Μεταβείτε στο directory του project.
- Μεταβείτε στο directory όπου βρίσκεται το Makefile.
- Εκτελέστε την εντολή: `sudo make start-project`. Αυτό θα δημιουργήσει το απαραίτητο δίκτυο και θα ξεκινήσει τα container για WordPress, MySQL, phpMyAdmin και Portainer.
- Περιμένετε να ξεκινήσουν, μόλις εκτελεστούν, μπορείτε να αποκτήσετε πρόσβαση στο WordPress στη διεύθυνση http://localhost:8080, στο mySQL στο http://localhost:3306, στο phpMyAdmin στο http://localhost:8888 και στο Portainer στο https://localhost:9443.

Step 2: Stop the Project

- Για να σταματήσετε το project και να αφαιρέσετε τα container, εκτελέστε την εντολή `sudo make stop-project`.

Αυτή η εντολή θα σταματήσει και θα αφαιρέσει τα container για WordPress, MySQL, phpMyAdmin και Portainer, αλλά θα διατηρήσει το δίκτυο και τα data volumes για μελλοντική χρήση.

Έτσι λοιπόν τρέχετε και σταματάτε το project με container WordPress, MySQL, phpMyAdmin και Portainer συνδεδεμένα μαζί. Μπορείτε να προσαρμόσετε και να επεκτείνετε το project με βάση τις απαιτήσεις σας.

Χρησιμοποιώντας το Makefile, μπορείτε εύκολα να εγκαταστήσετε, να εκτελέσετε και να διακόψετε το project μέσω της γραμμής εντολών, απλοποιώντας τη διαχείριση των container και διασφαλίζοντας τη σωστή συνδεσιμότητα τους εντός του καθορισμένου δικτύου.