سلام!

امیدوارم خوب باشید :)

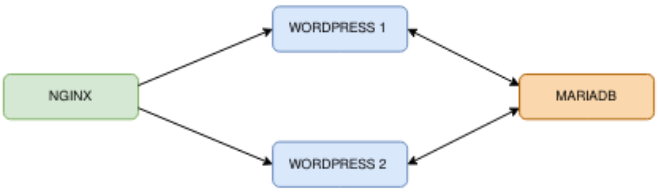
من آرمان رضا بدیع هستم و قصد دارم توضیحاتی رو در مورد پروژه ی خودم در آزمون عملی دوره Cangrow

با شما به اشتراک بزارم؛ خب شروع کنیم با اینکه تایتل پروژمون چی بوده..

"در این پروژه قصد داریم تا یک سایت وردپرسی با قابلیت مقیاس‌پذیری راه‌اندازی کنیم."

بیاین باهم مرحله مرحله جلو بریم و پروژه رو حل کنیم..

اول به راهنمای پروژه نگاهی میندازیم..



با استفاده nginx یک لودبالانسر ایجاد کنید و درخواست ها را بین دو سرویس wordpress تقسیم کنید.
برای دیتابیس از یک سرویس mariadb استفاده کنید.

پروژه ما با داکر کامپوز هست..

پس باید با استفاده از داکر کامپوز کانتینر های مورد نیازمون رو بالا بیاریم.

4 تا کانتینر نیاز داریم، دوتا کانتینر وردپرس که از کانتینر ماریا دیبی به عنوان دیتابیس استفاده میکنه و کانتینز nginx هم داریم که باید وردپرس هامون رو بهش لینک کنیم تا بین درخواست هایی که به سایتمون ارسال میشه لود بالانس رو انجام بده.

بیاین با دیتا بیس شروع کنیم چون توی اولویت اجرا هم اول قرار میگیره..

```yaml
services:
  db:
    image: mariadb
    container_name: Data_Base
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: somewordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress
    networks:
      - cangrownet
```

خب توی این قسمت از کد از ایمیج رسمی mariadb استفاده میکنیم و برای دیتایی که قراره ذخیره کنه، یه volume با نام db_data در نظر میگیریم (در ادامه باید درجای دیگه هم تعریفش کنیم). پارامتر restart هم مشخص میکنه که هرموقع کانتینر exit شد تحت هر شرایطی دوباره اونو بالا بیاره. در ادامه یسری متغیر مورد نیاز دیتا بیس و لینک کردنش به وردپرس ها تعریف کردیم که وصلش کردیم به شبکه ی cangrownet (اون رو هم باید در ادامه تعریف کنیم)..

بعد از اینکه دیتا بیسمون بالا اومد، میریم که ورد پرس هامون رو بیاریم بالا و کانفیگ کنیم:

```yaml
  wordpress1:
    container_name: wordpress1
    depends_on:
    - nginx
    image: wordpress
    volumes:
    - wordpress_data:/var/www/html
    restart: always
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
    networks:
      - cangrownet

  wordpress2:
    container_name: wordpress2
    depends_on:
    - nginx
    image: wordpress:latest
    volumes:
    - wordpress_data:/var/www/html
    restart: always
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
    networks:
      - cangrownet
```

در کد مربوط به وردپرس ها ما دوتا کانتینر وردپرس تعریف میکنیم که هر دو از ایمیج رسمی wordpress استفاده میکنن، برای هردو یه volume درنظر میگیریم که مشترک براشون استفاده میشه و پارامتر restart رو برای این دوتا هم تنظیم میکنیم، و بهشون متغیر های لازمه برای لینک شدن به دیتا بیس رو هم میدیم و به شبکه ی cangrownet وصلشون میکنیم..

و میریم سراغ کانتیر آخرمون که قراره نقش یه reverse proxy رو برامون بازی کنه و لود بالانس انجام بده..

```
18      nginx:
19        image: nginx
20        container_name: Nginx_LoadBalance
21        volumes:
22          - "./nginx/nginx.conf:/etc/nginx/nginx.conf"
23        ports:
24          - "80:80"
25        depends_on:
26        - db
27        networks:
28          - cangrownet
```

در اینجا از ایمیج رسمی nginx برای پروژمون استفاده کردیم و پورت 80 رو براش اکسپوز کردیم(که طبق *ضروریات پروژه تنها کانتینری هست که باید براش پورت در نظر گرفته بشه)

اجرا شدنش رو هم وابسته به دیتا بیسمون گذاشتیم(چون در هر صورت دیتابیس باید اول ران بشه تا ورد پرس ها که میخوان بیان بالا بتونن دیتا مورد نیازشون رو از دیتا بیس بردارن.)

و از طرفی برای اینکه nginx بتونه عمل لودبالانس رو انجام بده ما نیاز به فایل کانفیگ داریم که تو اون فایل کانفیگ های مد نظرمون رو برای nginx تنظیم کنیم.

ما اینجا با استفاده از volumes محل دقیق فایل کانفیگمون رو به داخل کانتینر لینک کردیم.

و در نهایت مثل بقیه کانتینر ها به شبکه ی cangrownet وصلش کردیم.

و حالا لازمه در مورد کانفیگ های ست شده برای nginx هم توضیحات لازمه رو بهتون بدم..

```
events {
    worker_connections 1024;
}

http {
    upstream wordpress_backend {
        server wordpress1;
        server wordpress2;
    }

    server {
        listen 80;

        location / {
            proxy_pass http://wordpress_backend;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }
    }
}
```

نگاهی به کد بندازیم..

بخش http مربوط به کانفیگ های پروتکل HTTP است. که در این بخش upstream تعریف کردیم، که در این upstream یک گروه از سرورها تعریف شدن که به عنوان مقصد برای لودبالانسینگ درخواست‌ها استفاده میشن. در اینجا دو سرور(کانیتر هامون) به نام‌های wordpress1 و wordpress2 رو بهش دادیم که از اونا به عنوان منبع برای توزیع درخواست‌هامون استفاده میشه.

بخش server شامل تنظیمات مربوط به سرور Nginx هست..

Listen80 در این کد به nginx میگه درخواست هایی که به این پورت اومد رو بر طبق کانفیگ های ما مدیریت کن..

در بخش location / مسیر پیشفرض برای درخواست هایی که به سرور ارسال میشود در نظر گرفته میشه که درواقع کابران وقتی به ریشه اصلی سایت یا همون / متصل میشن این بخش اجرا میشه..

میرسیم به کدی که کار لود بالانس رو انجام میده "proxy_pass http://wordpress_backend" این دستور به Nginx میگه که تمامی درخواست‌هایی که به این مسیر میرسه به گروه سرورهای معرفی شده در بخش upstream بفرست که درواقع همون دوتا سرور وردپرسمون هستن..

وچیزی که خیلی وقتم رو گرفت همین بخشیه که الان میخوام براتون توضیح بدم.

من در طی انجام پروژه یه مشکلی داشتم که اون هم این بود که هر هدری(header) که به درخواست ها اضافه میشد رو Nginx نمیتونست مسیر درستش رو تشخیص بده..

برای مثال: برای وارد شدن به بخش ادمین سایت ورد پرس درخواست wp-admin زده میشد اما nginx نمیتونست مقصد درست رو برای این درخواست پیدا کنه

بلاخره طی تحقیقاتی که انجام دادم و به کمک هوش مصنوعی عزیز chatgpt متوجه شدم باید این کد پایین رو به کانفیگ اضافه کنم..

```
16              proxy_set_header Host $host;
17              proxy_set_header X-Real-IP $remote_addr;
18              proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
19              proxy_set_header X-Forwarded-Proto $scheme;
```

بطور کلی این بخش کد میاد و درصورتی که به درخواستی که nginx قراره به سمت سرورهامون بفرسته، هدر ای(header) اضافه بشه، میاد و اون درخواست رو با همون هدر به سرورمون (کانتینرمون) میفرسته؛

برای مثال اگر wp-admin رو به درخواستمون اضافه کنیم، اون درخواسته ای که به سمت کانتینر ها باید بفرسته رو هم همینجوری میفرسته.

امیدوارم تونسته باشم خوب توضیح بدم چون فهم این بخش برای خودمم هم کمی سخت بود.

میرسیم به تعریف والیوم ها و شبکه پروژه:

```
61
62    volumes:
63        db_data: {}
64        wordpress_data: {}
65
```

در اینجا والیوم های پروژه رو تعریف میکنیم، که شامل یه والیوم برای دیتا بیس و یه والیوم برای دو ورس پرس میشه..

```
66    networks:
67      cangrownet:
68          driver: bridge
69          ipam:
70              driver: default
71              config:
72                  - subnet: "172.20.0.0/28"
73                    gateway: "172.20.0.1"
```

و در اینجا هم یه شبکه با نام cangrownet درست میکنیم و یه آدرس شبکه و سابنت براش تعیین میکنیم.

فاز دوم پروژه (ناقص انجام شده)..

در این فاز از پروژه ما باید یه دیتا بیس به پروژمون اضافه کنیم که نقش دیتا بیس replica رو بازی کنه.

یا میشه اینطوری گفت که ما یک دیتا بیس Master و یه دیتا بیس Slave داریم.

خب اولین کاری که باید بکنیم اضافه کردن دیتابیس جدیدمون به کد داکر کامپوزمون هست. و در ادامه برای اینکه این دو دیتا بیس همدیگر رو پیدا کنن و بتونن دیتاشون رو باهم یکی کنن ما نیاز به یه فایل کانفیگ براشون داریم..

ابتدا بریم و کد دیتابیس Master رو باهم نگاهی بندازیم..

```
4       db_master:
5         image: mariadb
6         container_name: Master_Database
7         volumes:
8           - db_master_data:/var/lib/mysql
9           - ./masterdb:/masterdb
10        restart: always
11        environment:
12          MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
13          MYSQL_DATABASE: ${MYSQL_DATABASE}
14          MYSQL_USER: ${MYSQL_USER}
15          MYSQL_PASSWORD: ${MYSQL_PASSWORD}
16        command: ["./masterdb/sync_setup.sh"]
17        networks:
18          - cangrownet
```

تنها تغییراتی که نسبت به کد قبلی دادیم یکی تغییر دادن نام کانتینر و سرویس هست و دیگری این که یه فایل اسکریپت بش رو به داخل کانتینر در قسمت volumes لینک کردیم و در بخش command دستور اجرای اون فایل بش رو بهش دادیم که در ادامه به اون کانفیگ ها هم میرسیم

و حالا دیتا بیس Replica :

```
20      db_replica:
21        image: mariadb
22        container_name: Replica_Database
23        volumes:
24          - db_replica_data:/var/lib/mysql
25          - ./replicadb:/replicadb
26        restart: always
27        environment:
28          MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
29          MYSQL_DATABASE: ${MYSQL_DATABASE}
30          MYSQL_USER: ${MYSQL_USER}
31          MYSQL_PASSWORD: ${MYSQL_PASSWORD}
32          MYSQL_REPLICA_USER: ${MYSQL_REPLICA_USER}
33          MYSQL_REPLICA_PASSWORD: ${MYSQL_REPLICA_PASSWORD}
34        command: ["./replicadb/sync_setup.sh"]
35        depends_on:
36          - db_master
37        networks:
38          - cangrownet
39
```

کلا ما این بخش رو در کد قبلی نداشتیم ولی شباهت زیادی به کد دیتابیس دیگرمون داره.. اون کانفیگ بش رو اینجا هم به داخل کانتینر در بخش volumes لینک کردیم و دستور اجرای فایل رو هم در بخش command بهش دادیم.. یسری environment جدید هم به عنوان دیتا بیس رپلیکا باید براش تعریف میکردیم که اینکارو انجام دادیم.

و حالا میریم سراغ بخشی که به مشکل خوردم و باعث شد نتونم بقیه پروژه رو انجام بدم، که درواقع مربوط به همین دو اسکریپت برای لینک کردن دو دیتا بیس بود.

این اسکریپت برای دیتا بیس مستر هست..

```
#!/bin/bash

apt update && apt upgrade
apt-get install -y mariadb-client

mariadb -u root -p"SQL_Arman_Pass" <<EOF
CREATE USER 'SQL_USRtest'@'%' IDENTIFIED BY 'SQL_Passtest';
GRANT ALL PRIVILEGES ON Master_Database.* TO 'SQL_USRtest'@'%';
FLUSH PRIVILEGES;
EOF

sleep infinity
```

این الان اخرین ورژن کد من هست که در گیت هاب پوش کردم اما من این کد رو با تمام متغیر های متفاوتی که در بخش environment تعریف کرده بودیم هم تست کردم اما جواب نداد..

تا خط چهارم کد بهش دستور نصب mariadb-client رو دادم جالبه بدونید که در ابتدا با mysql-client کد هارو نوشته بودم و دقیقا به همین اروری که الان بر میخورم، بر میخوردم و گفتم با mariadb-client هم تست کنم..

در کد خط شیش میاد و mariadb رو فراخوانی میکنه با یوزر پسوورد روت وارد میشه و مابقی کد هارو اجرا میکنه. مابقی کد هارو هم که با تحقیقاتی که انجام دادم گفته بود میتونین این کانفیگ رو ست کنین..

و اما مشکل اصلی اینجا بود که به محض فراخوانی mariadb یا میگم چون با mysql هم تست کردم، به محض فراخواندن هرکدوم در همون ابتدای خط 6، به ارور زیر بر میخورم..

ERROR 2002 (HY000): Can't connect to local MySQL server through socket " "/var/run/mysqld/mysqld.sock''

برای حل این مشکل هم که سرچ کردم گفته بودن کافیگ سوکت ها در فایل etc/mysql/my.cnf وجود داره اما با تغییراتی که اونجا میدادم باز هم به محض اجرای اولین کامند mysql به همین ارور برمیخوردم در کانفیگ دیتا بیس replica هم همینطور:

```
#!/bin/bash

apt update && apt upgrade
apt-get install -y mariadb-client

mariadb -u root -p"SQL_Arman_Pass" <<EOF
CREATE USER 'SQL_USRtest'@'%' IDENTIFIED BY 'SQL_Passtest';
GRANT REPLICATION SLAVE ON *.* TO 'SQL_USRtest'@'%';
FLUSH PRIVILEGES;
EOF

sleep infinity
```

مانند کد قبلی ابتدا دستور نصب mariadb-client رو بهش دادم و در ادامه دستورات مربوط به دیتابیس replica اما در اینجا هم دقیقا به محض فراخوانی mariadb به این ارور برمیخوردم:

ERROR 2002 (HY000): Can't connect to local MySQL server through socket " "/var/run/mysqld/mysqld.sock''

و خب متاسفانه در همینجا متوقف شدم و نتونستم بقیه پروژه رو انجام بدم..

در کل پروژه جذابی بود و باعث شد کلی چیز جدید یاد بگیرم و تجربه های جدید کسب کنم..

در آخر از شرکت پارت بابت راه اندازی این دوره تشکر میکنم و همچنین ممنون از تیم حرفه ای خوبتون بابت آموزش ها و کلا شرایطی که برای ما فراهم کردین :)