Hey everyone I'm mohammad Mohammadi and I'm so proud to present you with this awesome project. which I like to name it TrafficPress.

How to use the project:

1-Install git (for getting the newest updates):

```
sudo apt install git
```

2-Then we have to install the docker engine:

To install the docker engine please click on the link below:

Link

3-clone the project to the desired directory

4-start the project with this command:

```
sudo bash install.sh
```

Note: you can stop the project by : sudo bash uninstall.sh

How things work:

In this project, we have docker with some containers linked together by a network named: **cangrownet**

We have a nginx server that exposes port 80 to show the website and it will connect to the 2 WordPress servers and will balance the load between them. Then we will route the queries of WordPress servers to proxysql. we will route write queries to the master database and all read queries to the slave database.

Docker Compose file:

In this file, we created our services and assigned container names and hostnames to them. we will also put them in **cangrownet.** also, we address the dependencies of containers to start them in order.

For the required environment variables of each service, we use the variables of the .env file. WordPress servers will connect to proxysql on port 6033.

We mount a shared volume between WordPress servers and separate volumes for databases.

We mount the WordPress theme folder in the theme folder in services to make changes from outside.

Also, we mount configuration files of Maria DBs and proxysql.then we will mount our initial scripts in docker specified folder in containers.

In the network section, we address the cangrownet and set the subnet to /28.

---

start.bash:

in this file we generate the proxysql config file and then start docker containers

nginx/nginx.conf:

In the nginx config file, we will make a worker and set max connections to 1024. then we will address the WordPress servers with port 80. Then listen on port 80 and pass the incoming requests to WordPress (we put two WordPress servers before)

masterdb/masterdb.cnf:

in this config file, we will enable replication for master db by enabling binary logging. then we will give it a unique ID and log base name for better clarification.

masterdb/initial.sh:

in this bash script, we will create the slave user, grant privileges to be a slave, and replicate the master DB. Then we will create the database needed for WordPress servers and lastly, we will create the user with shared credentials and grant privileges to have all accesses.and we create the monitor user for proxysql.

replicadb/replicadb.cnf:

in this, we will only give a unique ID to the server and log base name for better clarification.

replicadb/initial.sh:

in this script we will get the status of the master database then we will save the log file and log position in variables. Then we will change the master of replica db to master db and reset the slave to make changes. lastly, we will create the user with shared credentials and grant privileges to have all accesses. and we create the monitor user for proxysql.

NOTE: we use separate shared user and monitor user in both databases for more security and better clarification.

proxysql/proxysql.cnf:

This file will be auto-generated by the initial script of proxysql

proxysql/initial.sh:

this script is a config generator for proxysql. It will generate admin credentials and will change some default variables. This section also has credentials for the monitor user. In the MySQL servers section, we will add our MariaDB servers in two host groups. Then we will address the shared user and the database that it has access to. In the MySQL query rules section, we will route all written queries with shared user credentials to the master db and also all read queries to replicate.

Phase 4 (CI/CD Actions):

In this phase I setup a github action in **.github/workflows/server_auto_pull.yml** to first checkout and then connect server with ssh and pull changes automatticaly. so the theme folder that mounted to wordpress servers will update changes automatticaly.

credits to : **appleboy/ssh-action**

In the end, I wanted to thank everybody that helped me with this project.