

به نام خدا

مقدمات

۱. اتصال به سرور

Ssh username@ip با این دستور به سرور خود ssh میزنیم

سپس پسورد خود را وارد میکنیم

```
debian@185.240.151.70's password:
```

و اگر پسورد را صحیح وارد کرده باشیم باید به صفحه سرور خود منتقل شویم

```
debian@185.240.151.70's password:
Linux cangrow 6.1.0-20-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.85-1 (2024-04-11) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Apr 21 07:43:03 2024 from 5.126.225.219
debian@cangrow:~$
```

۲. نصب ابزار های مورد نیاز

ما در اولین مرحله از ابزار tmux استفاده میکنیم که یک ابزار کمکی برای کامند لاین است و میتواند به سرعت کارما کمک کند

```
apt install tmux
```

سپس بهتر است برای نوشتن docker compose در vscode پلاگین مربوطه را نصب کنیم تا فرآیند کدنویسی ما راحت تر شود

در اولین مرحله برای نصب داکر با توجه به اینکه سرور نمیتواند به سایت داکر برای دانلود منتقل شود به دلیل تحریم ها باید از dns استفاده کنیم اما مسیر همیشگی dns که /etc/resolve.conf است را اگر تغییر دهیم dns ها اعمال نمیشوند

به همین دلیل باید فایل /etc/systemd/resolved.conf را تغییر دهیم و سپس ریستارتش کنیم `sudo systemctl restart systemd-resolved`

```
# Quad9: 9.9.9.9#dns.quad9.net
DNS=10.202.10.202
FallbackDNS=10.202.10.102
#Domains=
#DNSSEC=no
#DNSOverTLS=no
```

پس از این کار مشکل اتصال ما برطرف میشود

(در مسابقه پس از چند روز این مشکل برطرف شد)

حال نوبت نصب داکر است که ما اینجا از سایت خود داکر استفاده کرده ایم

ابتدا نسخه سیستم عامل خود را به شکل زیر میبینیم

```
root@cangrow:/opt/abolfazl_eshghabadi_cangrow# lsb_release
-a
No LSB modules are available.
Distributor ID: Debian
Description:   Debian GNU/Linux 12 (bookworm)
Release:       12
Codename:      bookworm
```

سپس به سایت داکر رفته و با توجه به این نسخه مراحل را دنبال میکنیم

<https://docs.docker.com/engine/install/debian>

یک اسکریپت ایجاد میکنیم و محتوایی که درون سایت گفته شده است را در آن وارد میکنیم

```
GNU nano 7.2                                install.sh *
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/debian/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/debian \
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

سپس دسترسی به این فایل میدهیم `chmod +x install.sh` بعد اجرا میکنیم `./install.sh`

بعد از آن دستور برای نصب کامل داکر را وارد میکنیم

```

root@cangrow:/opt/abolfazl_eshghabadi_cangrow# sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
containerd.io is already the newest version (1.6.31-1).
docker-compose-plugin is already the newest version (2.26.1-1~debian.12~bookworm).
The following additional packages will be installed:
  docker-ce-rootless-extras
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following packages will be upgraded:
  docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras
4 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 77.8 MB of archives.
After this operation, 15.4 kB disk space will be freed.

```

حال اگر dns را درست ست کرده باشید باید داکرتان کاملاً نصب شده باشد

Git

ما برای انتشار و deploy پروژه خود باید از گیت استفاده کنیم ابتدا ابزار گیت را نصب میکنیم

```
sudo apt update
```

```
sudo apt install git
```

سپس با دستور `git --version` میتوانیم ببینیم که گیت درست نصب شده است

حال باید در گیت هاب خود لاگین کنیم و یک ریپازیتوری پرایوت بسازیم

حال باید یک توکن برای خود بسازیم تا در مواقع نیاز و به راحتی بتوانیم تغییرات را اعمال کنیم

<https://docs.github.com/en/enterprise-server@3.9/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens>

این صفحه به صورت کامل و قدم به قدم آموزش ان را داده است

در پروژه ی خود بعد از انجام هر مقداری از کارهای هر فاز آنها را در گیت هاب push میکنیم

فاز اول

۱.دانلود ایمج ها

ابتدا باید به صورت ساده یک فاز ابتدایی را پیاده کنیم و بعد ان را توسعه دهیم

ایمج هارا از مخزن داکر هاب دانلود میکنیم

```
docker image pull imagename:tag
```

ما فعلاً به سه ایمج `nginx,wordpress,mariadb` نیاز داریم و آنها را یکی یکی دانلود میکنیم

سپس با دستور `docker images` آنها را مشاهده میکنیم

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
wordpress	latest	cc84570a8e5d	5 days ago	685MB
mariadb	latest	bc6434c28e9a	7 weeks ago	405MB
nginx	latest	c613f16b6642	2 months ago	187MB

۲. ساخت داکر کمپوس

سپس یک فایل داکر کمپوس مینویسیم و داخلش ایمج ها را اد میکنیم و سرویس هایی را میسازیم

اینجا سرویس های ما شامل :

Nginx,wordpress1,wordpress2,db

است ابتدا به صورت ساده آنها را ران میکنیم تا از صحت کار مطمئن شویم

Volume و network ها را اد میکنیم

```
networks:
  cangrownnet:
    ipam:
      config:
        - subnet: 172.28.0.0/28

volumes:
  wordpress_data:
    name: wordpress_data
```

(طبق گفته ی فایل پروژه ایپی ها را با سابنت /28 ایجاد میکنیم)

۳. nginx

ایمج nginx را مینویسیم

```
nginx:
  image: nginx:latest
  container_name: nginx_cangrow
  ports:
    - "80:80"
  volumes:
    - ./nginx/nginx.conf:/etc/nginx/nginx.conf

  depends_on:
    - wordpress1
    - wordpress2
  networks:
    - cangrownnet
```

پورت 80 را به 80 سیستم خود فروارد میکنیم تا از طریق ایپی در دسترس باشد

Network را به او میدهیم

Volume را ست میکنیم تا فایل کانفیگ به nginx منتقل شود

پس از این باید سرویس nginx داکر خود را به نحوی کانفیگ کنیم که به عنوان لودبالانسر عمل کند برای این کار باید یک فایل کانفیگ برای او ایجاد کنیم :

```
events {}
|
http {
    upstream wordpress {
        server wordpress1:80;
        server wordpress2:80;
        keepalive 64;
    }

    server {
        listen 80;

        # تنظیمات برای /wp-admin
        location /wp-admin {
            proxy_pass http://wordpress;
            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;
            proxy_redirect off;
            # اضافه کردن تنظیمات امنیتی
            proxy_hide_header X-Powered-By;
        }

        # تنظیمات برای سایر مسیرها
        location / {
            proxy_pass http://wordpress;
            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;
            proxy_redirect off;
            # اضافه کردن تنظیمات امنیتی
            proxy_hide_header X-Powered-By;
        }
    }
}
```

mariadb.۴

برای مثال تعریف سرویس db ما به این شکل است

```
db:
  image: mariadb:latest
  container_name: mariadb_cangrow
  environment:
    MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    MYSQL_DATABASE: ${MYSQL_DATABASE}
    MYSQL_USER: ${MYSQL_USER}
    MYSQL_PASSWORD: ${MYSQL_PASSWORD}
  volumes:
    - db_data:/var/lib/mysql
  networks:
    - cangrownet
```

در اینجا ولوم مربوط را به آن می‌دهیم (باید این ولوم را به بخش ولوم ها در انتهای فایل خود اضافه کنیم)

سپس environment ها را به آن add میکنیم

```
wordpress1:
  image: wordpress:latest
  container_name: wordpress1_cangrow
  environment:
    WORDPRESS_DB_HOST: ${WORDPRESS_DB_HOST}
    WORDPRESS_DB_USER: ${WORDPRESS_DB_USER}
    WORDPRESS_DB_PASSWORD: ${WORDPRESS_DB_PASSWORD}
    WORDPRESS_DB_NAME: ${WORDPRESS_DB_NAME}
  volumes:
    - wordpress_data:/var/www/html
  depends_on:
    - mariadb_replica
  networks:
    - cangrownnet

wordpress2:
  image: wordpress:latest
  container_name: wordpress2_cangrow
  environment:
    WORDPRESS_DB_HOST: ${WORDPRESS_DB_HOST}
    WORDPRESS_DB_USER: ${WORDPRESS_DB_USER}
    WORDPRESS_DB_PASSWORD: ${WORDPRESS_DB_PASSWORD}
    WORDPRESS_DB_NAME: ${WORDPRESS_DB_NAME}
  volumes:
    - wordpress_data:/var/www/html
  depends_on:
    - mariadb_replica
    - wordpress1
  networks:
    - cangrownnet
```

همانطور که میدانید باید ولوم وردپرس که ساختیم را به هردو سرویس وردپرس بدهیم اینگونه دیتای آنها با یکدیگر برابر هستند

۶. env file

سپس یک فایل env. در همان مسیری که پروژه داکر ما در آن قرار دارد ایجاد میکنیم تا متغیرهای محیطی را برای انعطاف پذیری بیشتر کد به آن بدهیم

```
WORDPRESS_DB_HOST=mariadb_master
WORDPRESS_DB_USER=cangrow
WORDPRESS_DB_PASSWORD=123
WORDPRESS_DB_NAME=wordpress
MYSQL_ROOT_PASSWORD=123
MYSQL_DATABASE=wordpress
MYSQL_USER=cangrow
MYSQL_PASSWORD=123
```

توجه !: حتما فایل `env` و `docker-compose.yml` باید در یک دایرکتوری باشند تا بتوانند یکدیگر را بخوانند

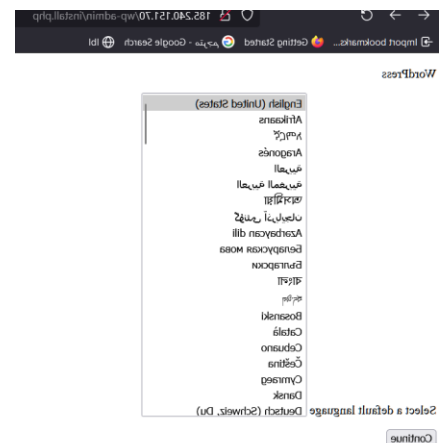
۷. بازکردن پورت

حال با ان کردن docker compose باید ان را در اینترنت مشاهده کنید اما این اتفاق نمی افتد دلیل ان هم فایروال میباشد که باید درخواست ها را به پورت 80 باز کنیم

`sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT` این دستور برای باز کردن پورت

`sudo iptables -D INPUT -p tcp --dport 80 -j ACCEPT` برای بستن پورت اگر خواستیم به حالت اول برگردد

در نهایت باید پس از زدن ایپی با این صفحه مواجه شوید



یکی دیگر از مواردی که باید رعایت شوند ترتیب ران شدن سرویس هاست زیرا اینگونه به همان ترتیب ایپی میگیرند و میتوانیم بدانیم همیشه هر سرور چه ایپی ای دارد

همانطور که در عکس زیر مشاهده میکنید به ترتیب سرویس ها ران شده اند و در نتیجه همیشه یک ایپی خاص را میگیرند


```
43c6849ealbaeab67903f"
  "Gateway": "172.28.0.1",
  "IPAddress": "172.28.0.2",
  "IPPrefixLen": 28,
  "IPv6Gateway": "",
  "GlobalIPv6Address": "",
  "GlobalIPv6PrefixLen": 0,
  "DriverOpts": null,
  "DNSNames": [
    "mariadb_cangrow",
    "db",
    "930a808a510c"
  ]
}
}
}
}
]
root@cangrow:/opt/abolfazl_eshghabadi_cangrow# mariadb_cangrow

39e48389a971fc85c12e7",
  "Gateway": "172.28.0.1",
  "IPAddress": "172.28.0.3",
  "IPPrefixLen": 28,
  "IPv6Gateway": "",
  "GlobalIPv6Address": "",
  "GlobalIPv6PrefixLen": 0,
  "DriverOpts": null,
  "DNSNames": [
    "wordpress1_cangrow",
    "wordpress1",
    "67099ccf89b64"
  ]
}
}
}
}
]
root@cangrow:/home/debian/docker# wordpress1_cangrow
[0] 0: bash*

f1a45b6b6b37e9d256cfe2",
  "Gateway": "172.28.0.1",
  "IPAddress": "172.28.0.5",
  "IPPrefixLen": 28,
  "IPv6Gateway": "",
  "GlobalIPv6Address": "",
  "GlobalIPv6PrefixLen": 0,
  "DriverOpts": null,
  "DNSNames": [
    "nginx_cangrow",
    "nginx",
    "d7f79ad05e98"
  ]
}
}
}
}
]
root@cangrow:/opt/abolfazl_eshghabadi_cangrow# nginx_cangrow

98abe9f2ff4c906b53a5ef",
  "Gateway": "172.28.0.1",
  "IPAddress": "172.28.0.4",
  "IPPrefixLen": 28,
  "IPv6Gateway": "",
  "GlobalIPv6Address": "",
  "GlobalIPv6PrefixLen": 0,
  "DriverOpts": null,
  "DNSNames": [
    "wordpress2_cangrow",
    "wordpress2",
    "b34ca6de32c1"
  ]
}
}
}
}
]
root@cangrow:/home/debian/docker# |wordpress2_cangrow
"cantrow" 10:59 15-Apr-24
```

فاز دوم

حال باید به جای یک سرویس db باید دو سرویس دیگر به نام های replica_cangrow و master_cangrow بسازیم

اینطوری هر دیتایی که در دیتابیس master ذخیره شود در دیتابیس replica نیز mirror میشود

1. docker compose edit

```
mariadb_master:
  image: mariadb:latest
  container_name: master_cangrow
  environment:
    MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    MYSQL_DATABASE: ${MYSQL_DATABASE}
    MYSQL_USER: ${MYSQL_USER}
    MYSQL_PASSWORD: ${MYSQL_PASSWORD}
  command: --server-id=1 --log-bin=mysql-bin --binlog-do-db=${MYSQL_DATABASE} --binlog-format=mixed
  volumes:
    - mariadb_master_data:/var/lib/mysql
    - ./masterdb/docker-entrypoint-initdb.d
  networks:
    - cangrownet
```

در اینجا دستورات ابتدایی برای کانفیگ db master را با استفاده از command وارد کرده ایم و سپس ولوم مربوط را به او داده ایم

```

mariadb_replica:
  image: mariadb:latest
  container_name: replica_cangrow
  environment:
    MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    MYSQL_DATABASE: ${MYSQL_DATABASE}
    MYSQL_USER: ${MYSQL_USER}
    MYSQL_PASSWORD: ${MYSQL_PASSWORD}
  command: --replicate-do-db=${MYSQL_DATABASE} --server-id=2 --log-bin=mysql-bin --binlog-do-db=${MYSQL_DATABASE}
  depends_on:
    - mariadb_master
  volumes:
    - mariadb_replica_data:/var/lib/mysql
    - ./replicadb:/docker-entrypoint-initdb.d
  networks:
    - cangrownet

```

تعریف سرویس های به این شکل است بخش command برای دادن ایدی سرور و روشن کردن binlog است که برای کانفیگ به صورت replica و master به آنها نیاز پیدا میکنیم

نکته !: من در ابتدا این کانفیگ ها را در فایل های initial.sh انجام داده بودم اما برای اجرای آنها نیاز بود تا mysql client را نصب کنم و این مقدار حجم زیادی مصرف میکرد برای هر بار ران شدن، پس از این روش استفاده کرده ام .

۳. ایجاد فایل initial

ابتدا ما باید یک اسکریپت برای ایجاد تنظیمات سرویس master انجام دهیم

```

CREATE USER 'rep_cangrow'@'%' IDENTIFIED BY 'cangrow';
GRANT REPLICATION SLAVE ON *.* TO 'rep_cangrow'@'%';

FLUSH PRIVILEGES;
SHOW MASTER STATUS;

```

در اینجا ما ابتدا نام یوزر replication را انتخاب کرده ایم و سپس با علامت % مشخص کردیم درخواست از هر آیی میتواند قبول شود میتوانیم برای افزایش امنیت به جای علامت % از آیی سرویس نیز استفاده کنیم.

اسکریپت را با نام initial.sql در مسیر masterdb/ پروژه خود ذخیره میکنیم

سپس یک اسکریپت برای سرویس replica_cangrow میسازیم

```
CHANGE MASTER TO
MASTER_HOST='master_cangrow',
MASTER_USER='rep_cangrow',
MASTER_PASSWORD='cangrow',
MASTER_LOG_FILE='mysql-bin.000001',
MASTER_LOG_POS=328;

START SLAVE;

SHOW SLAVE STATUS\G
```

در اینجا با استفاده از همان یوزر و پسوری که ساختیم اطلاعات master را کامل میکنیم در بخش های MASTER_LOG_FILE='mysql-bin.000001' باید نام فایل باینری را به دیتابیس بدهیم تا با توجه به آن فایل بتواند بکاپ ها را از سرور master دریافت کند

۳. تست سلامت

نکته مهم !: اگر در حال تست پروژه خود هستید و فرآیند کامل نشده است باید قبل از هربار روشن کردن کانتینر ها یکبار دو volume مربوط به دیتابیس ها را خالی کنید

با استفاده از دستور `docker volume inspect [name volume]` میتوانید از مسیر آن در سرور خود مطلع شوید و آن مسیر را خالی کنید

```
root@cangrow:/var/lib/docker/volumes/abolfazl_eshghabadi_cangrow_mariadb_replica_data/_data# rm -r *
root@cangrow:/var/lib/docker/volumes/abolfazl_eshghabadi_cangrow_mariadb_replica_data/_data# cd /var/lib/docker/volumes/abolfazl_eshghabadi_cangrow_mariadb_
master_data/_data
root@cangrow:/var/lib/docker/volumes/abolfazl_eshghabadi_cangrow_mariadb_master_data/_data# rm -r *
root@cangrow:/var/lib/docker/volumes/abolfazl_eshghabadi_cangrow_mariadb_master_data/_data# ls
root@cangrow:/var/lib/docker/volumes/abolfazl_eshghabadi_cangrow_mariadb_master_data/_data#
```

پس از انجام این کارها برای چک کردن صحت اینکه آیا دیتابیس ها به درستی کار میکنند چند کار را باید انجام دهید

1. بررسی لاگ های کانتینر ها با این دستور `docker logs -f [container name]` و ببینید آیا ارور خاصی برنخورده اید و فایل ها به درستی اجرا شده اند
2. رفتن به دیتابیس و چک کردن نحوه برقراری صحیح ارتباط بین دیتابیس ها :

```
mysql -u root -p -h replica_canGrow|
```

```
mysql -u root -p -h master_canGrow
```

با این روش شما به هرکدام از سرور های دیتابیس خود متصل میشوید البته دقت کنید که باید `mysql client` را در سرور خود نصب کنید (میتوانید در ابتدا با دستور `exec` به کانتینر های خود وصل شوید و از آنجا اجرا کنید اما اینگونه برای هربار تست یکبار نیاز است تا `mysql client` را نصب کنید)

دستور برای نصب `mysql client` : `sudo apt-get install default-mysql-client`

حال برای چک کردن دیتابیس ها دستور های زیر را بزنید و نتیجه را ببینید

SHOW MASTER STATUS; برای دیدن صحت کانفیگ master که باید چیزی شبیه به این را نمایش دهد

```
+-----+-----+-----+-----+
| File           | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+-----+-----+-----+-----+
| mysql-bin.000002 |      342 | wordpress    |                   |
+-----+-----+-----+-----+
1 row in set (0.001 sec)
```

SHOW SLAVE STATUS\G این دستور باید در سرور replica اعمال شود و باید بدون ارور اتصال را برای ما نمایش دهد

```
Replicate_Wild_Ignore_Table:
      Last_Errno: 0
      Last_Error:
      Skip_Counter: 0
Exec_Master_Log_Pos: 342
      Relay_Log_Space: 1883
      Until_Condition: None
      Until_Log_File:
      Until_Log_Pos: 0
      Master_SSL_Allowed: No
      Master_SSL_CA_File:
      Master_SSL_CA_Path:
      Master_SSL_Cert:
      Master_SSL_Cipher:
      Master_SSL_Key:
Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
      Last_IO_Errno: 0
      Last_IO_Error:
      Last_SQL_Errno: 0
      Last_SQL_Error:
Replicate_Ignore_Server_Ids:
      Master_Server_Id: 1
      Master_SSL_Crl:
      Master_SSL_Crlpath:
           Using_Gtid: No
           Gtid_IO_Pos:
Replicate_Do_Domain_Ids:
Replicate_Ignore_Domain_Ids:
           Parallel_Mode: optimistic
           SQL_Delay: 0
           SQL_Remaining_Delay: NULL
      Slave_SQL_Running_State: Slave has read all relay log; waiting for more updates
           Slave_DDL_Groups: 3
Slave_Non_Transactional_Groups: 1
           Slave_Transactional_Groups: 0
1 row in set (0.001 sec)
```

نقاطی که زیر آنها خط کشیده شده باید به همین شکل باشند و اروری نداشته باشند

۳. رفع ارور

در این صورت شما به صورت درست کانفیگ بخش replica و master را انجام داده اید اما در این مراحل به یک مشکل برمیخورید و آن این است که host ها که به جای ایپی از آنها استفاده کرده ایم در مسیر /etc/hosts تعریف نشده اند و ما برای این کار یک اسکریپت نوشته ایم که قبل از اجرای docker compose باید ران شود :

```
#!/bin/bash

#masir host ro behesh midim
HOSTS_FILE="/etc/hosts"

#yek araye az service ha tarif mikonim
declare -A services=(
    [master_cangrow]="172.28.0.2"
    [replica_cangrow]="172.28.0.3"
    [wordpress1_cangrow]="172.28.0.4"
    [wordpress2_cangrow]="172.28.0.5"
    [nginx_cangrow]="172.28.0.6"
)

#add kardan ip ha be hosts va agar tekrari nabod
for service in "${!services[@]}"; do
    if ! grep -q "${services[$service]} $service" "$HOSTS_FILE"; then
        echo "${services[$service]} $service" | sudo tee -a "$HOSTS_FILE" > /dev/null
    fi
done

echo "all ips added to $HOSTS_FILE "
```

در اینجا با توجه به اولویت بندی که اول در بخش docker compose کرده ایم میتوانیم بفهمیم کدام ایپی برای کدام سرویس اختصاص داده میشود و به این شکل آنها را در هاست ها اد کنیم

در کد ابتدا ما یک آرایه تعریف کرده ایم و تمام هاست ها و ایپی ها را در آن قرار داده ایم سپس با استفاده از یک حلقه for چک کرده ایم که آیا آن هاست در مسیر ما وجود دارد یا خیر و اگر وجود نداشت آن را در آن مسیر قرار داده ایم

نکته : فایل /dev/null فایلی است که هرچه در آن نوشته شود به صورت خودکار حذف میشود و حکم یک نوع سطل زباله را در کد ما دارد

پس از اجرای اسکریپت میبینیم که به درستی اد شده اند

```
127.0.0.1 localhost
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

127.0.0.1 cangrow
172.28.0.6 nginx_cangrow
172.28.0.5 wordpress2_cangrow
172.28.0.4 wordpress1_cangrow
172.28.0.2 master_cangrow
172.28.0.3 replica_cangrow
```

همانطور که در عکس پایین مشاهده میکنید تا قبل از اجرای اسکریپت به هاست ها دسترسی نداریم اما بعد از ان با انها اتصال داریم

```
root@cangrow:/opt/abolfazl_eshghabadi_cangrow# ping master_cangrow
ping: master_cangrow: Name or service not known
root@cangrow:/opt/abolfazl_eshghabadi_cangrow# ./requirements.sh
all ips added to /etc/hosts
root@cangrow:/opt/abolfazl_eshghabadi_cangrow# ping master_cangrow
PING master_cangrow (172.28.0.2) 56(84) bytes of data.
64 bytes from master_cangrow (172.28.0.2): icmp_seq=1 ttl=64 time=0.064 ms
64 bytes from master_cangrow (172.28.0.2): icmp_seq=2 ttl=64 time=0.119 ms
64 bytes from master_cangrow (172.28.0.2): icmp_seq=3 ttl=64 time=0.054 ms
64 bytes from master_cangrow (172.28.0.2): icmp_seq=4 ttl=64 time=0.082 ms
^C
--- master_cangrow ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3030ms
rtt min/avg/max/mdev = 0.054/0.079/0.119/0.024 ms
```

فاز سوم

۱. دریافت ایميج

ابتدا ما نیاز به یک ایميج proxysql داریم تا بار دیتا را بین دو سرور master و replica تقسیم کنیم

```
docker pull proxysql/proxysql
```

۲. docker compose

سپس باید فایل docker compose را تغییرات دهیم

```

proxysql:
  image: proxysql/proxysql
  container_name: proxysql_cangrow
  ports:
    - "6032:6032"
    - "6033:6033"
  environment:
    MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    MYSQL_DATABASE: ${MYSQL_DATABASE}
    MYSQL_USER: ${MYSQL_USER}
    MYSQL_PASSWORD: ${MYSQL_PASSWORD}
  volumes:
    - ./proxysql/proxysql.cnf:/etc/proxysql/proxysql.cnf
    - ./proxysql/initial.sh:/etc/proxysql/initial.sh
  depends_on:
    - mariadb_master
    - mariadb_replica
    - nginx
  networks:
    - cangrownet

```

ابتدا نام را برای سرویس انتخاب میکنیم و پورت ها را expose میکنیم برای اتصال دیتابیس ها به سرور

بعد متغیر های محیطی را از فایل env. میخوانیم سپس دو volume اد میکنیم که باید کانفیگ ها را درون آنها قرار دهیم

سپس تعیین میکنیم که این سرویس آخرین سرویس بالا بیاید زیرا nginx آخرین سرویس بوده است پس با تعریف آن سرویس در depends-on باعث میشود تا این سرویس به صورت آخرین سرویس بالا بیاید

در نهایت نیز نتورک را برایش مشخص کرده ایم

۳. ساخت فایل کانفیگ دیتابیس

فایل proxysql.cnf را در مسیر /proxysql ایجاد میکنیم

برای نحوه کانفیگ فایل هم میتوانید از سایت اصلی آن کمک بگیرید : <https://proxysql.com/documentation>

هم میتوانید از هوش مصنوعی استفاده کنید که میتواند به شما کمک زیادی بکند

من در اینجا بخشی از آن را برایتان توضیح داده ام

```
[mysql]
datadir=/var/lib/proxysql

admin_variables=
{
    admin_credentials="admin:admin;cluster:password"
    mysql_ifaces="0.0.0.0:6032"
    refresh_interval=2000
}
```

ابتدا در بخش اول ما رمز و پسورد یوزر ادمین را استفاده کرده ایم که بعدا برای اتصال به آن نیاز داریم

```
mysql_variables=
{
    threads=4
    max_connections=2048
    default_query_delay=0
    default_query_timeout=36000000
    have_compress=true
    poll_timeout=2000
    interfaces="0.0.0.0:6033;/tmp/proxysql.sock"
    default_schema="information_schema"
    stacksize=1048576
    server_version="5.5.30"
    connect_timeout_server=3000
    monitor_username="monitor"
    monitor_password="monitor"
    monitor_history=600000
    monitor_connect_interval=60000
    monitor_ping_interval=10000
    ping_interval_server_msec=120000
    ping_timeout_server=500
    commands_stats=true
    sessions_sort=true
    monitor_reader_only_nodes=true
}
```

در این بخش نیز ما یوزر مانیتور را تعریف کرده که یوزر و پسورد او را برابر همان دیفالت گذاشته ایم برای امنیت بالاتر میتوانید او را تغییر دهید


```
# Define hostgroups
mysql_servers =
(
    { address="master_cangrow" , port=3306 , hostgroup=10 , max_connections=100 },
    { address="replica_cangrow" , port=3306 , hostgroup=20 , max_connections=100 }
)

# Define users
mysql_users =
(
    { username = "cangrow", password = "123", default_hostgroup = 10, active = 1 }
)

# Define query rules
mysql_query_rules =
(
    { rule_id=100, active=1, match_pattern="^SELECT", destination_hostgroup=20, apply=1 },
    { rule_id=200, active=1, match_pattern="^INSERT|^UPDATE|^DELETE", destination_hostgroup=10, apply=1 }
)

# Load to runtime
LOAD MYSQL SERVERS TO RUNTIME;
LOAD MYSQL USERS TO RUNTIME;
LOAD MYSQL QUERY RULES TO RUNTIME;

# Save to disk
SAVE MYSQL SERVERS TO DISK;
SAVE MYSQL USERS TO DISK;
SAVE MYSQL QUERY RULES TO DISK;
```

این نیز ادامه آن است که در بخش بعد آنها را توضیح خواهم داد

۴. ساخت فایل اسکریپت

سپس در همان مسیر قبلی یک فایل با نام initial.sh میسازیم و دسترسی لازم را به او میدهیم

سپس به شکل زیر کاملش میکنیم

```
#!/bin/bash

sleep 10

mysql -u admin -padmin -h 127.0.0.1 -P 6032 --execute="
INSERT INTO mysql_servers(hostgroup_id, hostname, port) VALUES (10, 'master_cangrow', 3306);
INSERT INTO mysql_servers(hostgroup_id, hostname, port) VALUES (20, 'replica_cangrow', 3306);
INSERT INTO mysql_users(username, password, default_hostgroup) VALUES ('cangrow', '123', 10);
INSERT INTO mysql_users(username, password, active, use_ssl, default_hostgroup, max_connections) VALUES ('monitor', 'monitor', 1, 0, 10, 10000);
INSERT INTO mysql_query_rules(rule_id, active, match_pattern, destination_hostgroup, apply) VALUES (100, 1, '^SELECT', 20, 1);
INSERT INTO mysql_query_rules(rule_id, active, match_pattern, destination_hostgroup, apply) VALUES (200, 1, '^INSERT|^UPDATE|^DELETE', 10, 1);
LOAD MYSQL SERVERS TO RUNTIME; SAVE MYSQL SERVERS TO DISK;
LOAD MYSQL USERS TO RUNTIME; SAVE MYSQL USERS TO DISK;
LOAD MYSQL QUERY RULES TO RUNTIME; SAVE MYSQL QUERY RULES TO DISK;
"

echo "config proxysql complete"
```

در بخش اول کد :

```
mysql -u admin -padmin -h 127.0.0.1 -P 6032 --execute="
```

ما به یوزر ادمین در proxysql متصل میشویم (برای اتصال از سرور خود نیز میتوانیم اقدام کنیم اما باید دسترسی را باز کنیم که این کار امنیت را کاهش میدهد)

```
INSERT INTO mysql_servers(hostgroup_id, hostname, port) VALUES (10, 'master_cangrow', 3306);
INSERT INTO mysql_servers(hostgroup_id, hostname, port) VALUES (20, 'replica_cangrow', 3306);
```

سپس در این بخش دو سرور Replcia و master را که قبلا ساخته ایم به سرور معرفی میکنیم تا بتواند آنها را بشناسد

```
INSERT INTO mysql_users(username, password, default_hostgroup)
VALUES ('cangrow', '123', 10);
INSERT INTO mysql_users(username, password, active, use_ssl, default_hostgroup, max_connections)
VALUES ('monitor', 'monitor', 1, 0, 10, 10000);
```

(برای خوانایی بیشتر در نقاط سبز رنگ کد به خط بعد رفته است اما در اصل کد باید در یک خط باشد)

در این بخش دو یوزر را به سرور معرفی میکنیم یک یوزر cangrow که برای دیتابیس ما است و یک یوزر monitor که در فایل کانفیگ نیز از او نام بردیم

```
INSERT INTO mysql_query_rules(rule_id, active, match_pattern,
destination_hostgroup, apply) VALUES (100, 1, '^SELECT', 20, 1);
INSERT INTO mysql_query_rules(rule_id, active, match_pattern,
destination_hostgroup, apply) VALUES (200, 1, '^INSERT|^UPDATE|^DELETE', 10, 1);
```

در این بخش از کد نیز که در هدف اصلی ما است به دیتابیس replica که به آن آیدی 20 را اختصاص داده ایم فقط رول select داده شده است و به دیتابیس master ما که آیدی آن 10 است 3 رول دیگر برای انجام تغییرات و اپدیت داده شده است

۵. اپدیت دو اسکرپت master و replica

حال اگر سرور ها را اجرا کنیم و لاگ های سرویس proxysql_cangrow را ببینیم متوجه میشویم که کاربر مانیتور نمیتواند به دو دیتابیس ما متصل شود :

```
2024-04-17 20:44:26 MySQL_Monitor.cpp:1328:monitor_connect_thread(): [ERROR]
Server replica_cangrow:3306 is returning "Access denied" for monitoring user
2024-04-17 20:44:27 MySQL_Monitor.cpp:1328:monitor_connect_thread(): [ERROR]
Server master_cangrow:3306 is returning "Access denied" for monitoring user
```

این به دلیل تعریف نشدن کاربر مانیتور در دو دیتابیس دیگر ما است برای رفع این مشکل باید این بخش کد را درون هر دو فایل کافینگ که برای دیتابیس ها ساختمون کنیم

```
CREATE USER 'monitor'@'%' IDENTIFIED BY 'monitor';
GRANT SELECT, PROCESS ON *.* TO 'monitor'@'%';
FLUSH PRIVILEGES;
```

به طور مثال در سرور master به این شکل است

```
CREATE USER 'rep_cangrow'@'%' IDENTIFIED BY 'cangrow';
GRANT REPLICATION SLAVE ON *.* TO 'rep_cangrow'@'%';

CREATE USER 'monitor'@'%' IDENTIFIED BY 'monitor';
GRANT SELECT, PROCESS ON *.* TO 'monitor'@'%';
FLUSH PRIVILEGES;

FLUSH PRIVILEGES;
SHOW MASTER STATUS;
```

۶. راه اندازی proxysql

حال کاری که باید بکنیم این است که فایلی که برای کانفیگ کاننتینر ساختیم را با دستوری اجرا کنیم که برای این کار روش های مختلفی وجود دارد ، روشی که ما تا به حال از آن استفاده میکردیم دستور “command” در docker compose بوده است اما این روش در این کاننتینر باعث ایجاد مشکلاتی میشود .

ما برای وصل شدن به ادمین باید دستور را از درون همان کاننتینر اجرا کنیم

اگر با استفاده از دستور command اسکریپت را ران کنیم به مشکل مواجه میشویم زیرا

```
root@cangrow:/opt/abolfazl_eshghabadi_cangrow# mysql -u admin -padmin -h proxy
sql_cangrow -P 6032 --prompt='ProxySQLAdmin> '
ERROR 1040 (42000): User 'admin' can only connect locally
```

میبینید که اگر این دستور را از طریق سرور خودمان اجرا کنیم به ما میگوید که فقط باید از درون سرور اجرا شود

حال اگر بخواهیم از command استفاده کنیم چون در حین بالا آمدن کاننتینر دستور اجرا میشود آیدی 127.0.0.1 که همان آیدی loopback ما است را به جای آیدی سرویس proxysql آیدی خودش در نظر میگیرد پس اتصال ما به مشکل میخورد

(با استفاده از دستوراتی مانند sleep و یا استفاده از حلقه while برای به تاخیر انداختن فرآیند باز هم مشکل حل نمیشود)

پس ما با استفاده از یک اسکریپت که در کنار فایل docker-compose.yml ما قرار میگیرد میتوانیم این مشکل را برطرف کنیم و از این فایل به عنوان یک فایل اجرا کننده پروژه استفاده کنیم

۷. نوشتن اسکریپت cangrow-web.sh

نکته : در پروژه در حال انجام ابتدا نام این اسکریپت requirements.sh بوده است !

ما در ادامه پروژه از این اسکریپت استفاده های زیادی میکنیم و مدام او را توسعه میدهیم فعلا برای رفع مشکل او را ساده مینویسیم :

```
# Rane kardan docker-compose
docker compose up -d

# entezar baraye etminan az bala amadan tamam container ha
echo "Waiting for proxysql_cangrow container running..."
while ! docker exec proxysql_cangrow /bin/bash -c "echo 'Proxysql is running'" &> /dev/null; do
    sleep 1
done

# Ejra kardan initial.sh dar container proxysql_cangrow
docker exec proxysql_cangrow /bin/bash /etc/proxysql/initial.sh

echo "service proxysql is runnig "
```

در اینجا ابتدا ما فایل docker compose خود را راه اندازی میکنیم و صبر میکنیم تا سرویس proxysql که در واقع آخرین سرویس ما است به طور کامل بالا بیاید و بعد از آن با دستور docker exec فایل اسکریپت خود را راه اندازی میکنیم

کارهای نهایی و تست سلامت

حال که سرویس خود را راه اندازی کرده ایم باید در فایل env. خود سایت های wordpress خود را به جای سرور master به سرور proxysql متصل کنیم تا این سرویس بار پخش کوئری را انجام دهد

```
WORDPRESS_DB_HOST=proxysql_cangrow:6033
```

حال ابتدا اسکریپت `cangrow-web.sh` را اجرا میکنیم .

سپس برای تست سلامت و اینکه آیا سرویس ما به درستی ران شده است باید این مراحل را انجام دهیم

ابتدا با دستور زیر به سرور `proxysql` متصل میشویم

```
docker exec -it proxysql_cangrow bash
```

بعد با دستور زیر به `ProxySQLAdmin` متصل میشویم

```
root@5fd34e15bc30:/# mysql -u admin -padmin -h 127.0.0.1 -P 6032 --prompt='ProxySQLAdmin> '
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.5.30 (ProxySQL Admin Module)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

ProxySQLAdmin> |
```

بعد از آن با استفاده از دستورات چک میکنیم که آیا سرور ها و یوزر های ما به درستی تعریف شده اند یا خیر

```
ProxySQLAdmin> SELECT * FROM mysql_users;
+-----+-----+-----+-----+-----+-----+
| username | password | active | use_ssl | default_hostgroup | default_schema |
| max_connections | attributes | comment |
+-----+-----+-----+-----+-----+-----+
| cangrow | 123 | 1 | 0 | 10 | NULL |
| 10000 | | | | |
| monitor | monitor | 1 | 0 | 10 | NULL |
| 10000 | | | | |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.001 sec)

ProxySQLAdmin> |
```

```
ProxySQLAdmin> SELECT * FROM mysql_servers;
+-----+-----+-----+-----+-----+-----+-----+-----+
| hostgroup_id | hostname          | port | gtid_port | status | weight | compression | max_connections |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 10           | master_cangrow    | 3306 | 0          | ONLINE | 1       | 0            | 1000             |
| 20           | replica_cangrow    | 3306 | 0          | ONLINE | 1       | 0            | 1000             |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.000 sec)
```

همانطور که میبینید هر دو سرور ما به صورت آنلاین هستند و کانفیگ ها به درستی انجام شده است

1. توسعه اسکریپت cangrow-web

حال که مراحل ابتدایی ما تمام شده است یک اسکریپت برای آسانی استفاده از سرویس خود مینویسیم و آن را توسعه میدهیم

ابتدا باید یک ورودی از کاربر بگیریم و یک منو از کارهایی که برنامه انجام میدهد را نمایش دهیم

```
echo -e "1.start cangrow-web \n2.stop cangrow-web \n3.WordPress user authentication \n4.clear cangrow-web \n5.help \n6.exit"
read -p "select the option: " VAR1
VAR1=$(echo $VAR1 | tr -d [:alpha:] | tr -d [:blank:])
```

در اینجا ابتدا یک منو از دستورات را با استفاده از echo نمایش داده ایم و سپس ورودی را از کاربر دریافت کرده ایم

در بخش آخر نیز متغیر را از فاصله ها خالی کردیم و حروف آن را حذف کرده ایم

```
15 clear
16 docker compose up -d
17 if [ "$startup_config" == "0" ]; then
18     echo "Waiting for proxysql_cangrow container running..."
19     while ! docker exec proxysql_cangrow /bin/bash -c "echo 'Proxysql is running'" &> /dev/null; do
20         sleep 1
21     done
22     docker exec proxysql_cangrow /bin/bash /etc/proxysql/initial.sh
23     declare -A services=(
24         [master_cangrow]="172.28.0.2"
25         [replica_cangrow]="172.28.0.3"
26         [wordpress1_cangrow]="172.28.0.4"
27         [wordpress2_cangrow]="172.28.0.5"
28         [nginx_cangrow]="172.28.0.6"
29         [proxysql_cangrow]="172.28.0.7"
30     )
31     for service in "${!services[@]}; do
32         if ! grep -q "${services[$service]} $service" "$HOSTS_FILE"; then
33             echo "${services[$service]} $service" | sudo tee -a "$HOSTS_FILE" > /dev/null
34         fi
35     done
```

در لاین 19 ابتدا چک میکنیم که سرور proxysql_cangrow ران شده باشد و اگر ران شده بود در خط 22 اسکریپت را درونش ران میکنیم

در خط 23 تا 30 نیز یک آرایه از سرویس ها میسازیم و سپس اگر وجود نداشتند آنها را اد میکنیم

در شرط if در خط 17 از کد، برای اینکه متوجه شویم تنظیمات اولیه تا به حال انجام شده است یا خیر یک متغیر به فایل .env. خود افزوده ایم که اگر برابر صفر باشد به معنی انجام نشدن تنظیمات اولیه است

با این دستور نیز میتوانیم فایل خود را به اسکریپت ادا کنیم

```
if [ "$?" = 0 ]; then
    source .env
fi
```

این بخش را نیز به فایل .env. می افزاییم

```
MYSQL_PASSWORD=123
startup_config=0
```

در لاین 38 نیز با دستور sed مقدار متغیر را به 1 تغییر میدهیم

```
36 echo "all ips added to $HOSTS_FILE"
37 echo "startup config complete"
38 sed -i 's/startup_config=0/startup_config=1/' .env
39 else
```

اگر کاربر عدد 2 را به ورودی بدهد دستور docker compose down اجرا میشود

```
2)
clear
docker compose down
echo "cangrow-web stopped..."
echo "Press Enter to continue..."
read -p ""
;;
```

این بخش مهم است !

وقتی در سایت وردپرسی خود وارد میشویم و مراحل نصب را انجام میدهیم و یوزر می سازیم یوزر ما دسترسی کامل برای مدیریت پنل را ندارد در این بخش ما پس از اینکه کاربر مراحل ثبت نام خود را انجام داده بود دسترسی کامل را به یوزر او میدهیم

```
53 3)
54 clear
55 mysql -u root -h master_cangrow -p$MYSQL_ROOT_PASSWORD -e "
56 USE $MYSQL_DATABASE;
57 INSERT INTO wp_usermeta (user_id, meta_key, meta_value) VALUES (1, 'wp_capabilities', 'a:1:{s:13:\"administrator\";b:1;});
58 INSERT INTO wp_usermeta (user_id, meta_key, meta_value) VALUES (1, 'wp_user_level', '10');
59 "
60 echo "You have access to make changes:)"
61 echo "Press Enter to continue..."
62 read -p ""
63 ;;
64 4)
```

اگر کاربر عدد 4 را به ورودی بدهد ابتدا از او سوال میشود و اگر تایید کردن

تمام volume های اپ حذف میشوند و مقدار متغییر ما برای تنظیمات استارتاپ نیز برابر با 0 قرار میگیرد و کاملاً سرویس داکری ما حذف و پاک میشود

```
64 4)
65 clear
66 echo "remove all volumes?(all volumes removed!) "
67 read -p "yes OR no: " confirmation
68 confirmation=$(echo $confirmation | tr '[:upper:]' '[:lower:]')
69 if [ "$confirmation" == "yes" ]; then
70     docker compose down
71     echo "cangrow-web stopped"
72     docker volume rm wordpress_data
73     docker volume rm proxysql_data
74     docker volume rm mariadb_master_data
75     docker volume rm mariadb_replica_data
76     sed -i 's/startup_config=1/startup_config=0/' .env
77
78     echo "All volumes removed."
79 else
80     echo "Volume removal cancelled."
81 fi
82 echo "Press Enter to continue..."
83 read -p ""
84 ;;
```

در بقیه کد نیز فرمان های دیگر طراحی شده اند

```
85 5)
86 clear
87 echo -e "1:started and run startup config for cangrow-web \n2.stopped cangrow
88 echo "Press Enter to continue..."
89 read -p ""
90 ;;
91 6)
92 echo "Good luck ..."
93 sleep 2
94 clear
95 break
96 ;;
97
98 *)
99 echo "Invalid option, please try again."
100 sleep 1
101 ;;
102 esac
103 done
104
```

ما تمام کد را در یک حلقه while true گذاشتیم تا برنامه پس از اجرای دستور بسته نشود

```

3 #masir host ro behesh midim
4 HOSTS_FILE="/etc/hosts"
5
6 while true; do
7     source .env
8     clear
9     echo -e "1.start cangrow-web \n2.stop
10 read -p "select the option: " VAR1

```

4. فاز چهارم

برای انجام فرآیند ci/cd ما از github و action در آن استفاده میکنیم

ما از ssh-deploy برای این کار استفاده میکنیم

<https://github.com/marketplace/actions/ssh-deploy>

این سایت آن است و میتوانید از آموزش های آن استفاده کنید

۱. ساخت کلید ssh

ابتدا باید یک جفت کلید ssh برای اتصال بسازیم

وارد کامند لاین میشویم و دستور `ssh-keygen -m PEM -t rsa -b 4096` وارد میکنیم

```

root@cangrow:/home/debian# ssh-keygen -m PEM -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): /home/debian/docker/key-ssh
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/debian/docker/key-ssh
Your public key has been saved in /home/debian/docker/key-ssh.pub
The key fingerprint is:
SHA256:TQ9bWWJkxlovdljFR/nX8PxmK92to29VlFnionxczz0 root@cangrow
The key's randomart image is:
+---[RSA 4096]---+
|                |
|  o* o+O|       |
| +o=o+=+|       |
| oo++o=+|       |
| +.B+oooO|      |
| S =.+o E%|     |
| . oX|         |
| =.|          |
| + |          |
| .+..|         |
+-----[SHA256]-----+
root@cangrow:/home/debian#

```

پس از اجرای دستور باید مسیری که میخواهیم کلید ها درون آن ذخیره شوند را وارد کنیم

سپس رمزی را برای ssh تعریف میکنیم (اگر اینتر بزنید بدون رمز ثبت میشود)


```
-rw-r--r-- 1 root root 738 Apr 21 12:51 key-ssh.pub
-rw----- 1 root root 3243 Apr 21 12:51 key-ssh
```

اگر به همان مسیر برویم میبینیم دو تا فایل ذخیره شده است

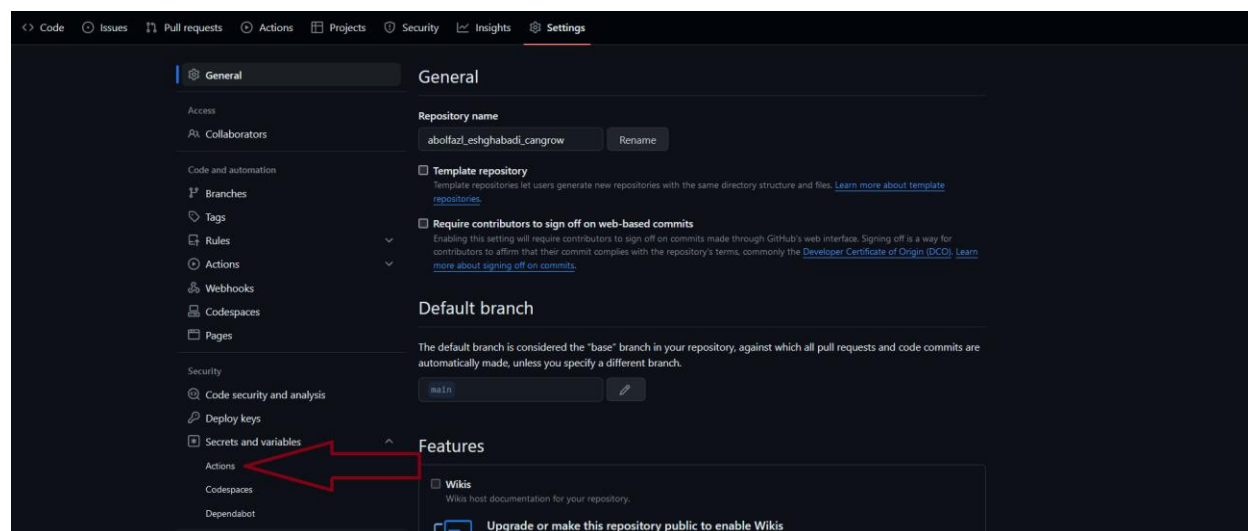
فایلی که پسوند pub. دارد کلید پابلیک ما است و فایل دیگر کلید پرایوت است

```
root@cangrow:/home/debian# cd /home/debian/.ssh/
root@cangrow:/home/debian/.ssh# nano
authorized_keys id_rsa id_rsa.pub
root@cangrow:/home/debian/.ssh# nano authorized_keys |
```

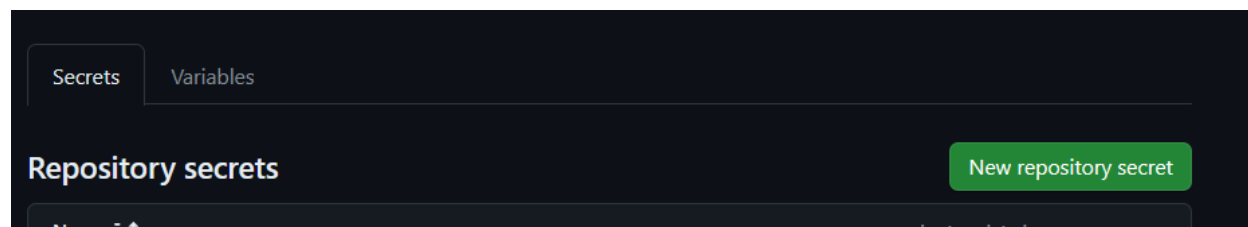
وارد مسیر /home/youruser/.ssh/ میشویم و مانند تصویر فایل را باز میکنیم

و محتوای کلید پابلیک را درون آن کپی میکنیم

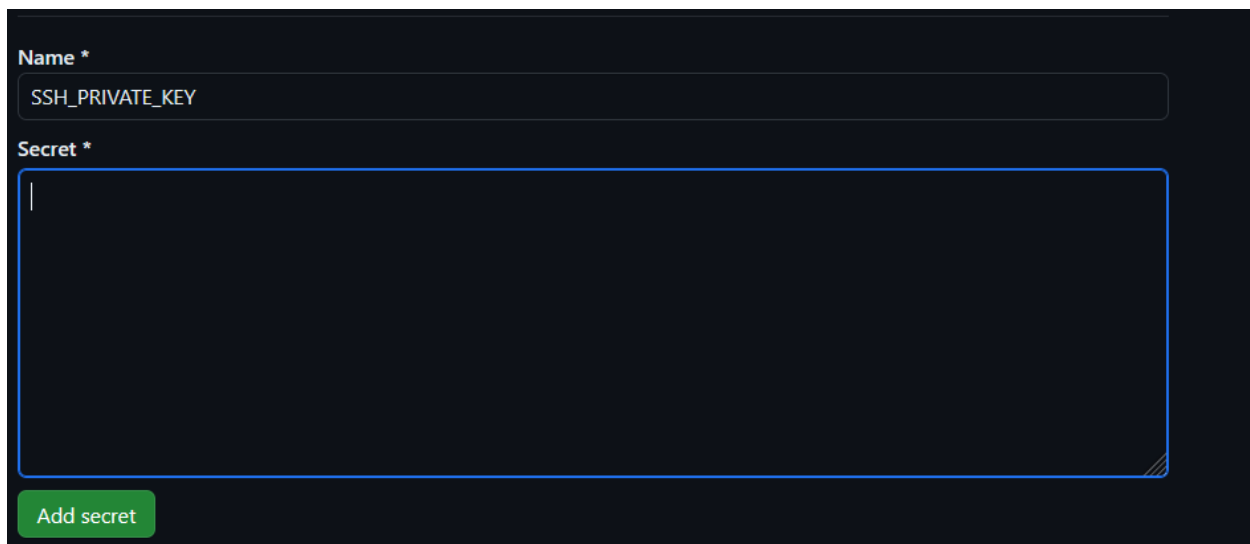
سپس وارد صفحه گیت هاب خود شد



وارد این بخش شده



روی new repository کلیک کرده



Name *

SSH_PRIVATE_KEY

Secret *

Add secret

در بخش name مانند تصویر یک نام وارد کنید

در بخش secret کلید پرایوت خود را وارد کنید

۲. ایجاد و تنظیم فایل workflow

ابتدا باید در دایرکتوری اصلی یک دایرکتوری به شکل زیر ایجاد کنیم

```
root@cangrow:/opt/abolfazl_eshghabadi_cangrow# tree .github
.github
├── workflows
│   └── theme_deploy.yml
```

سپس یک فایل با نام دلخواه و پسوند .yml ایجاد میکنیم

```

1 name: Deploy WordPress Theme
2
3 on:
4   push:
5     branches:
6       - main
7     paths:
8       - 'themes/**'
9
10  jobs:
11    deploy:
12      runs-on: ubuntu-latest
13      steps:
14        - name: Checkout repository
15          uses: actions/checkout@v2
16
17        - name: Sync theme directory to server
18          uses: easingthemes/ssh-deploy@v3
19          env:
20            SSH_PRIVATE_KEY: ${ secrets.SSH_PRIVATE_KEY }
21            REMOTE_HOST: ${ secrets.REMOTE_HOST }
22            REMOTE_PORT: ${ secrets.REMOTE_PORT }
23            REMOTE_USER: ${ secrets.REMOTE_USER }
24            ARGS: '-avzr --delete --rsync-path="sudo rsync"'
25            SOURCE: 'themes/'
26            TARGET: ${ secrets.TARGET }
27
28

```

در این کد گفته ایم که در صورتی که در پوشه themes در ریپازیتوری ما تغییری ایجاد شد action شروع به کار کند

نکته : وقتی اینگونه ست کنیم از انجام ci/cd های اضافی جلوگیری میشود

در خط 20 SSH_PRIVATE_KEY را اد میکنیم که در مرحله قبل ایجاد کرده بودیم

Line21 : سپس در بخش های بعد هاستی که میخواهیم به آن وصل شویم که همان آیپی سرور ما است

Line22 : پورت ssh را قرار میدهیم که

Line 23 : یوزری که به آن وصل میشویم که همان Debian است

Line26 : مسیری که فایل باید در انجا منتقل شود

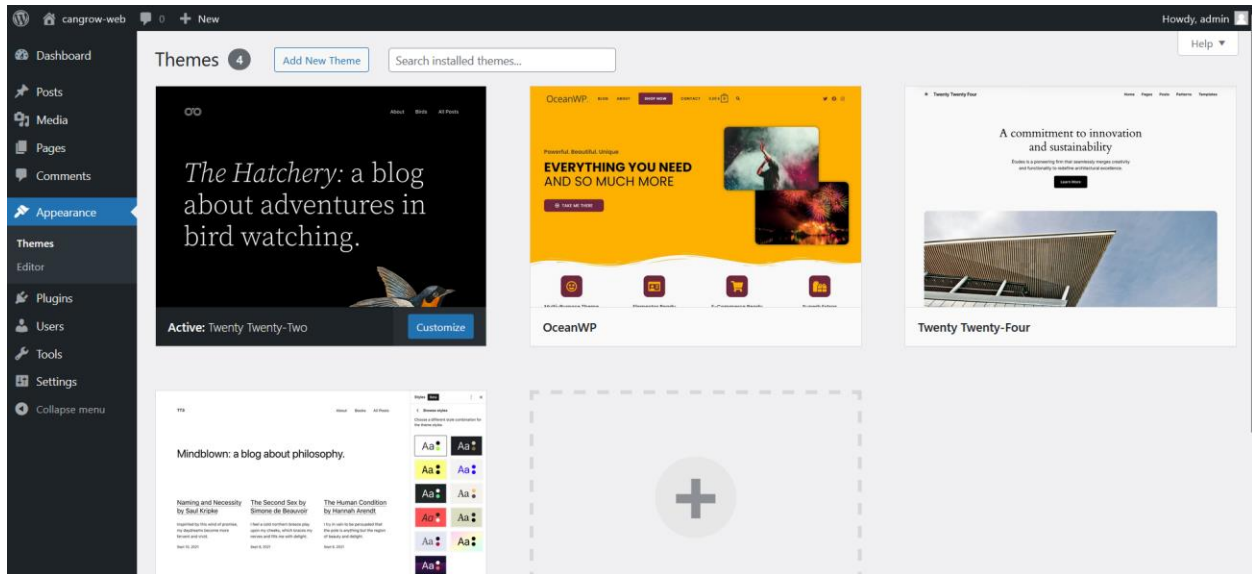
همه ی این متغیر ها را مانند اد کردن private key در گیت هاب ، به گیت هاب خود اضافه کنید

۳. ایجاد پوشه برای themes

در مسیر `/opt/yourname` یک دایرکتوری با نام themes میسازیم

و تمی که میخواهیم روی وردپرس اعمال شود را درون آن کپی میکنیم

۴. تست ci/cd



به طور مثال من این تم را روی وردپرس اعمال کرده ام

```
root@cangrow:/opt/abolfazl_eshghabadi_cangrow/themes/twentytwentythree# cd templates/
root@cangrow:/opt/abolfazl_eshghabadi_cangrow/themes/twentytwentythree/templates# ls
404.html blank.html home.html page.html single.html
archive.html blog-alternative.html index.html search.html
root@cangrow:/opt/abolfazl_eshghabadi_cangrow/themes/twentytwentythree/templates# rm index.html
-bash: rm: command not found
root@cangrow:/opt/abolfazl_eshghabadi_cangrow/themes/twentytwentythree/templates# cd /opt/abolfazl_eshghabadi_cangrow/
root@cangrow:/opt/abolfazl_eshghabadi_cangrow# git add -A
root@cangrow:/opt/abolfazl_eshghabadi_cangrow# git commit -m "faz4:2 test2 final ci/cd"
[main e4b1237] faz4:2 test2 final ci/cd
2 files changed, 1 insertion(+), 28 deletions(-)
delete mode 100644 themes/twentytwentythree/templates/index.html
root@cangrow:/opt/abolfazl_eshghabadi_cangrow# git push -u origin main
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (8/8), 650 bytes | 650.00 KiB/s, done.
Total 8 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), completed with 5 local objects.
To https://github.com/Sobanggg/abolfazl_eshghabadi_cangrow
8feald3..e4b1237 main -> main
branch 'main' set up to track 'origin/main'.
root@cangrow:/opt/abolfazl_eshghabadi_cangrow#
```

```
page-large-header.html
page-no-separators.html
search.html
single.html
single-no-separators.html
theme.json

178 directories, 1312 files
root@cangrow:/var/lib/docker/volumes/wordpress_data/_data/wp-content/themes# ls
index.php oceanwp twentytwentyfour twentytwentythree twentytwentytwo
root@cangrow:/var/lib/docker/volumes/wordpress_data/_data/wp-content/themes# cd twentytwentythree
root@cangrow:/var/lib/docker/volumes/wordpress_data/_data/wp-content/themes/twentytwentythree# cd templates/
root@cangrow:/var/lib/docker/volumes/wordpress_data/_data/wp-content/themes/twentytwentythree/templates# ls -la
total 40
drwxr-xr-x 2 1001 docker 4096 Apr 22 09:04 .
drwxr-xr-x 7 1001 docker 4096 Apr 22 09:04 ..
-rw-r--r-- 1 1001 docker 318 Apr 22 09:04 404.html
-rw-r--r-- 1 1001 docker 1673 Apr 22 09:04 archive.html
-rw-r--r-- 1 1001 docker 60 Apr 22 09:04 blank.html
-rw-r--r-- 1 1001 docker 1502 Apr 22 09:04 blog-alternative.html
-rw-r--r-- 1 1001 docker 2063 Apr 22 09:04 home.html
-rw-r--r-- 1 1001 docker 890 Apr 22 09:04 page.html
-rw-r--r-- 1 1001 docker 1815 Apr 22 09:04 search.html
-rw-r--r-- 1 1001 docker 939 Apr 22 09:04 single.html
root@cangrow:/var/lib/docker/volumes/wordpress_data/_data/wp-content/themes/twentytwentythree/templates#
```

همانطور که میبینید وقتی یک فایل `index.html` را حذف کرده ایم در آن مسیر نیز اعمال شده است

2. توسعه اسکریپت cangrow-web

حال باید اسکریپت خود را کامل تر کنیم تا بخش های مورد نیاز مانند نصب نیازمندی ها را به طور کامل انجام دهد

```

17     echo "installing docker "
18     sudo apt-get update
19     sudo apt-get install ca-certificates curl
20     sudo install -m 0755 -d /etc/apt/keyrings
21     sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
22     sudo chmod a+r /etc/apt/keyrings/docker.asc
23     echo \
24     "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
25     $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
26     sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
27     sudo apt-get update
28     sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
29
30     # اجرای تست hello-world
31     if sudo docker run hello-world; then
32         echo "Docker installed and tested successfully."
33     else
34         echo "error installing docker "
35         read -p ""
36         exit 1
37     fi

```

در ابتدا برای نصب داکر این بخش را به کد اضافه میکنیم و در صورتی که درست نشود برنامه را میبندیم تا کاربر مشکل را برطرف کند

```

38     sudo apt-get install docker-compose
39
40     # Docker Compose
41     if docker-compose --version; then
42         echo "Docker Compose installed successfully."
43     else
44         echo "error installing docker compose"
45         read -p ""
46         exit 1
47     fi

```

در این بخش نیز داکر کمپوس را نصب میکنیم

```

48     # نصب MySQL Client
49     sudo apt-get install mysql-client
50
51     # تست نصب MySQL Client
52     if mysql --version; then
53         echo "MySQL Client has been installed successfully."
54     else
55         echo "There was an issue installing MySQL Client."
56         read -p ""
57         exit 1
58     fi

```

در این بخش نصب کلاینت mysql را انجام میدهم برای اجرای دستورات و تست سلامت دیتابیس ها نیاز است

```

59 LINE='debian ALL=(ALL) NOPASSWD:ALL'
60
61 # add sudo permission
62 if sudo grep -Fxq "$LINE" /etc/sudoers; then
63     echo "There has been permission"
64 else
65     echo "permission sudo add"
66     echo "$LINE" | sudo EDITOR='tee -a' visudo
67 fi
68

```

در این بخش از کد نیز ما دسترسی اجرای `sudo` بدون نیاز به رمز را به کاربر `Debian` داده ایم تا در صورت اجرای `ci/cd` دسترسی را داشته باشد و بتواند تغییرات را اعمال کند

نکته : میتوانید به جای این کار رمز را درون `Secret` قرار داده و در فایل `workflow` استفاده کنید اما این روش امنیت را بالاتر میبرد

```

146 6)
147 clear
148 mysql -u root -h replica_cangrow -p123 -e "SHOW SLAVE STATUS\G"
149 read -p ""
150 ;;
151
152
153
154 7)
155 clear
156 mysql -u root -h master_cangrow -p123 -e "SHOW MASTER STATUS;"
157 read -p ""
158 ;;
159
160 8)
161 clear
162 docker exec -it proxysql_cangrow bash -c "
163 mysql -u admin -padmin -h 127.0.0.1 -P 6032 --prompt='ProxySQLAdmin> ' -e '
164 SELECT * FROM mysql_users;
165 SELECT * FROM mysql_servers;
166 '"
167 read -p ""
168 ;;
169

```

سپس در این بخش با وارد کردن هر عدد یک دیتابیس را چک میکند و نتیجه را نمایش میدهد اینگونه کاربر راحت تر میتواند از صحت دیتابیس خود مطمئن شود