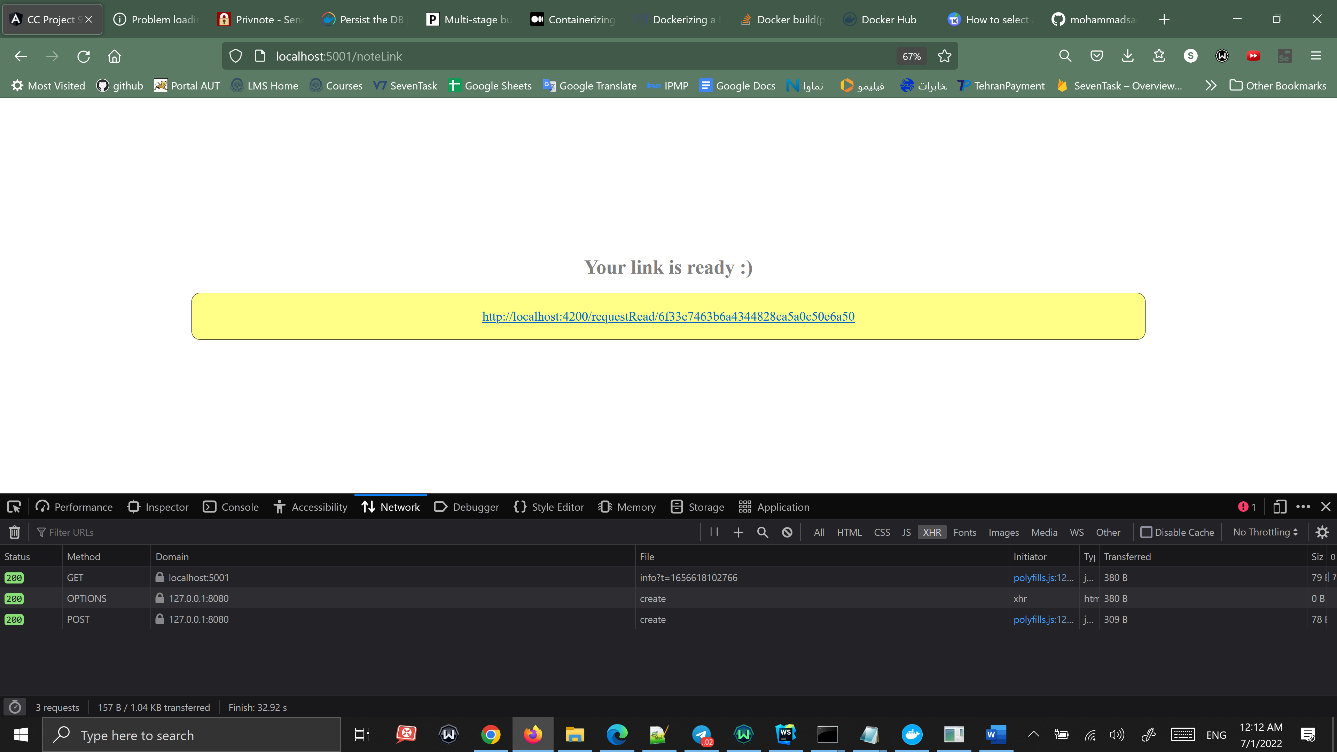
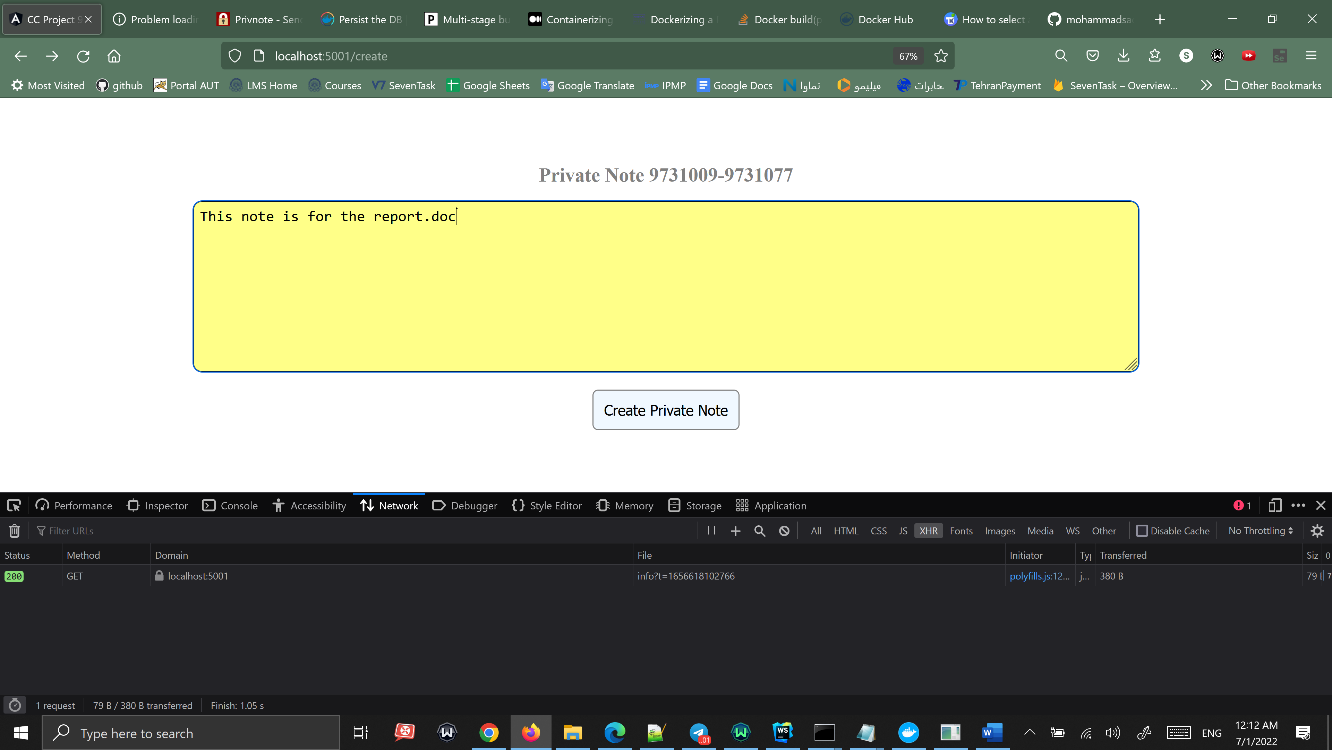
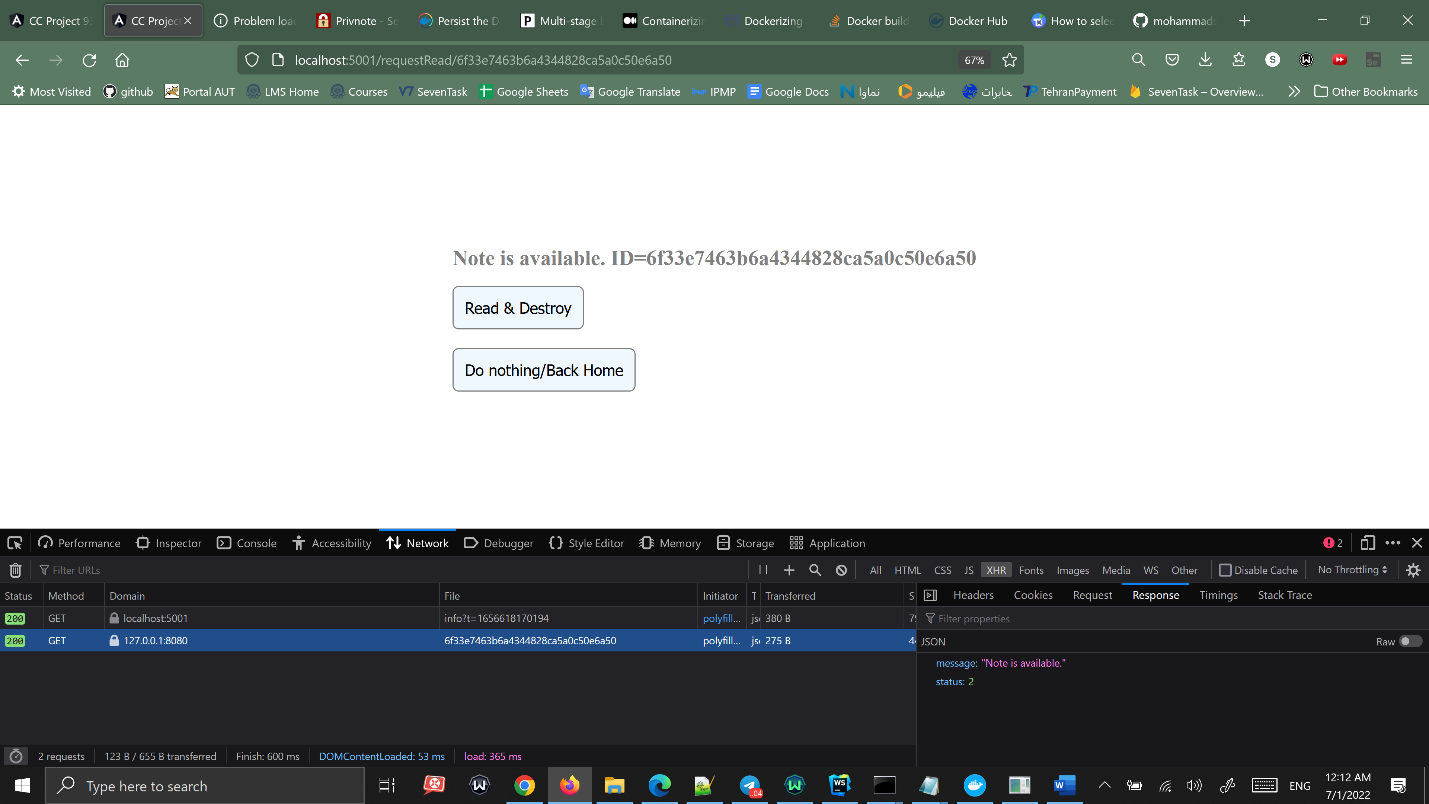
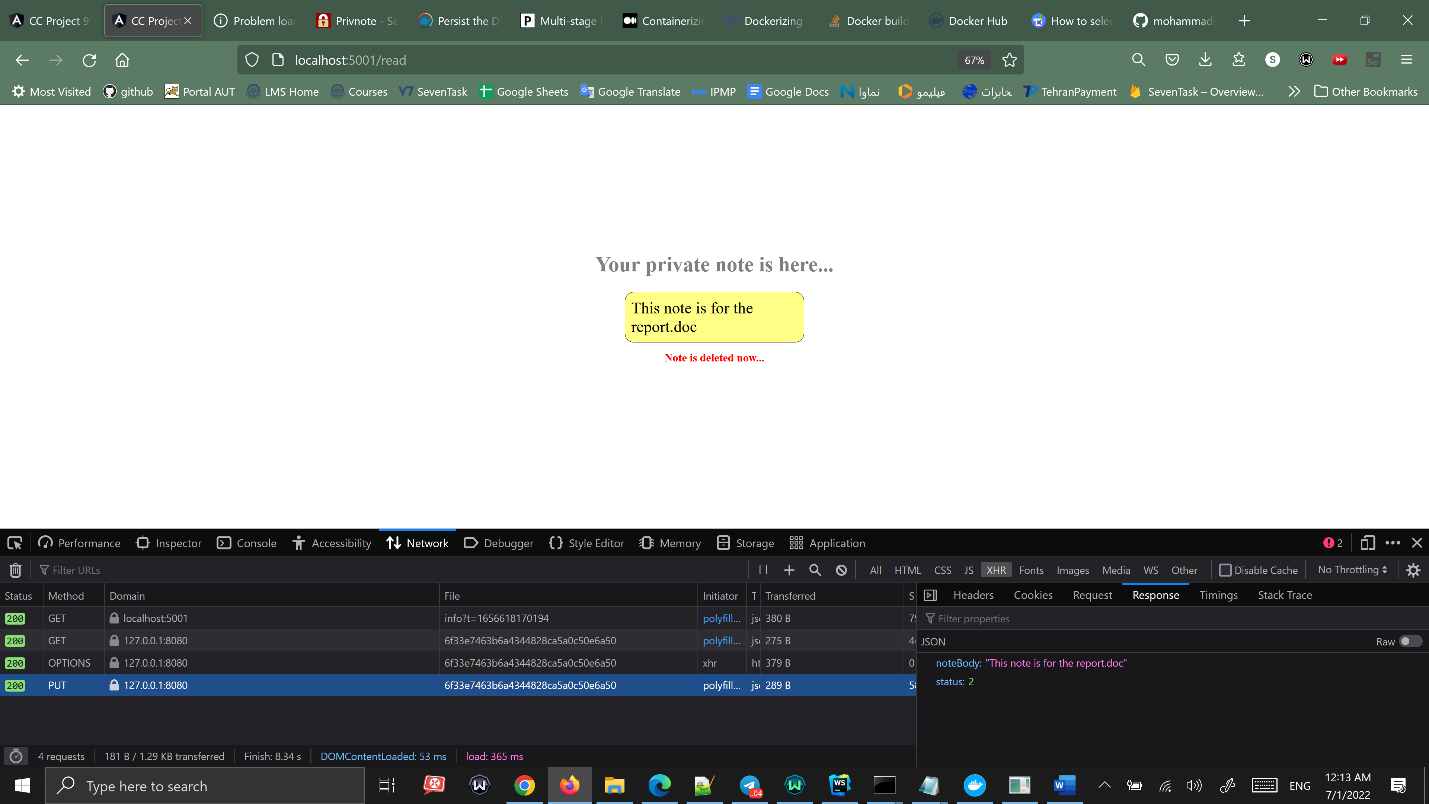
**گزارش پروژه‌ی رایانش ابری**

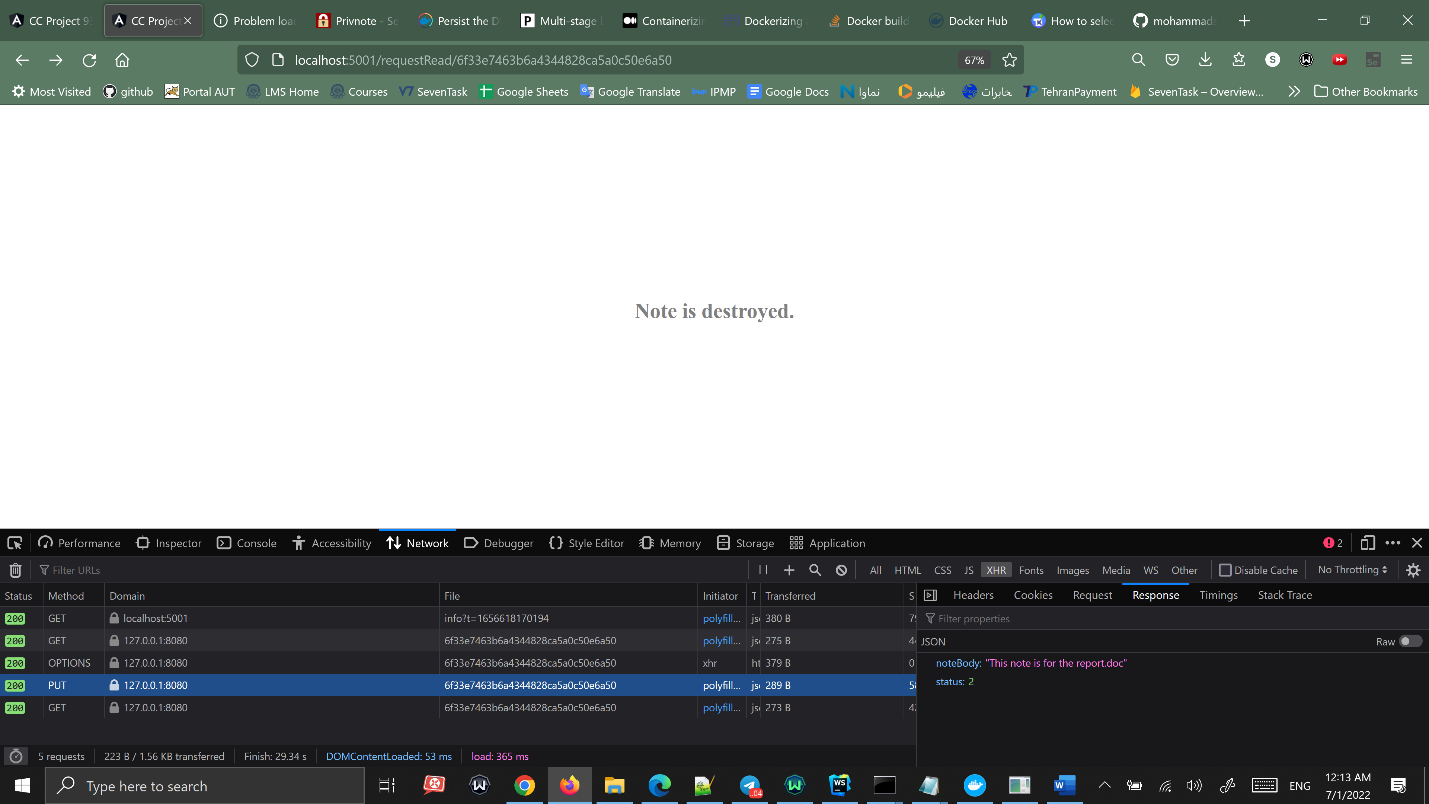
**گام اول**

بکند با استفاده از پایتون و MySQL

فرانت انگولار

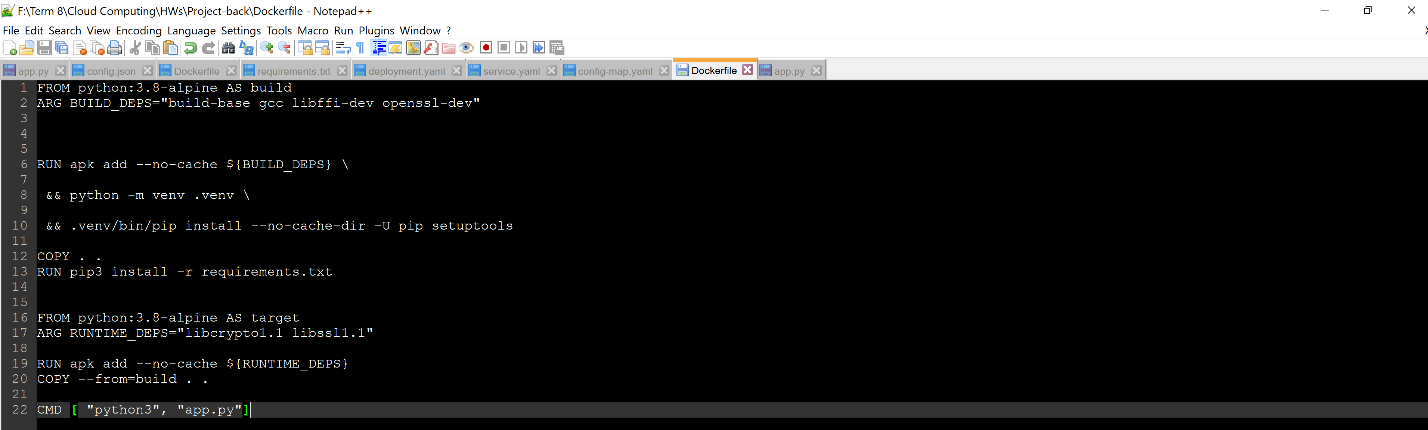






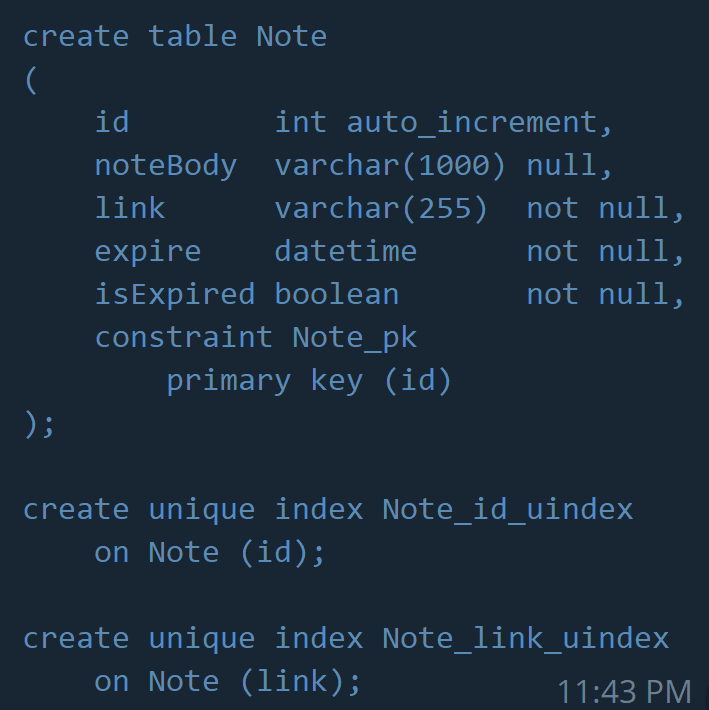
**گام دوم**

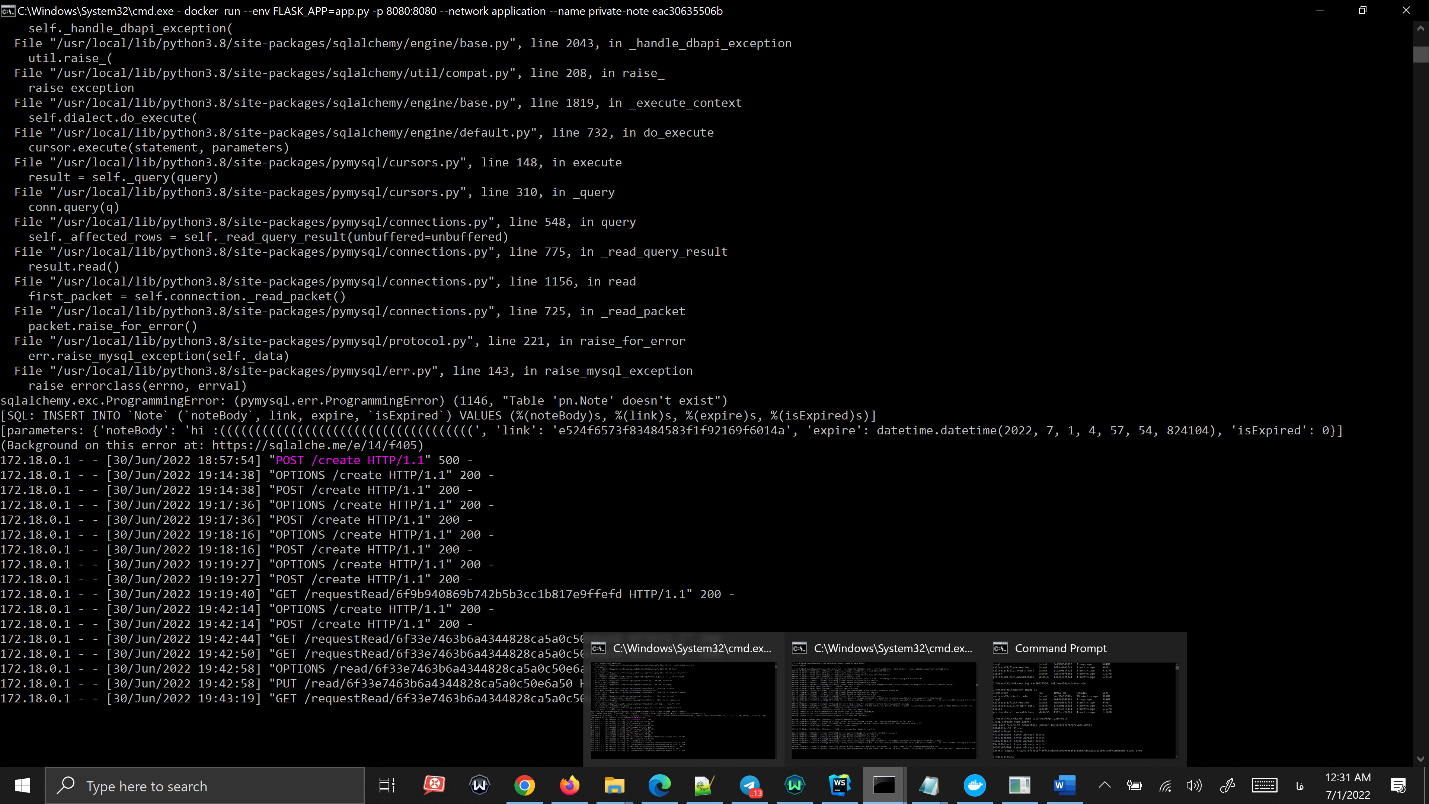
ابتدا داکرفایل را به روش multistage می‌نویسیم که چون Python زبان مفسری است خیلی در کم کردن حجم تاثیر ندارد.

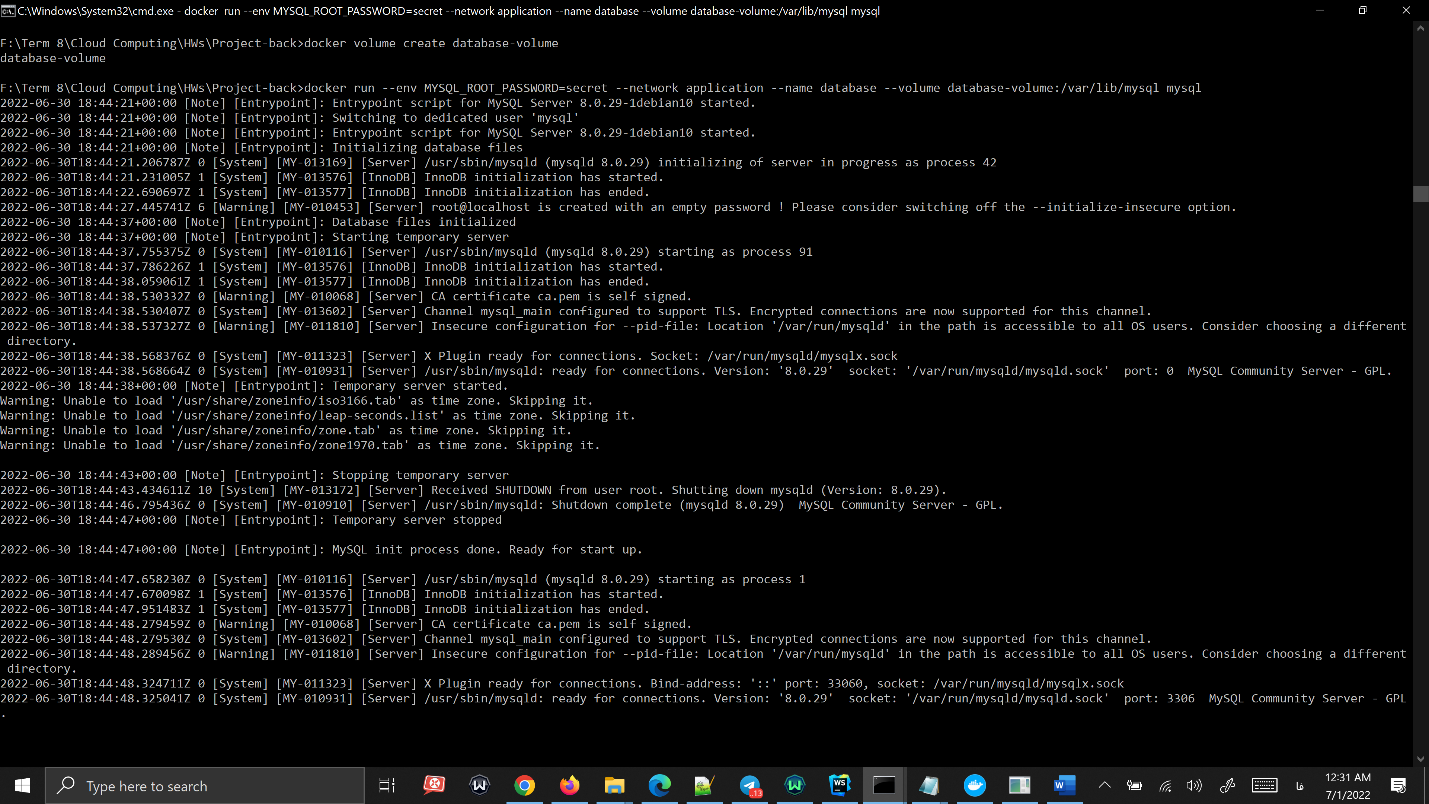
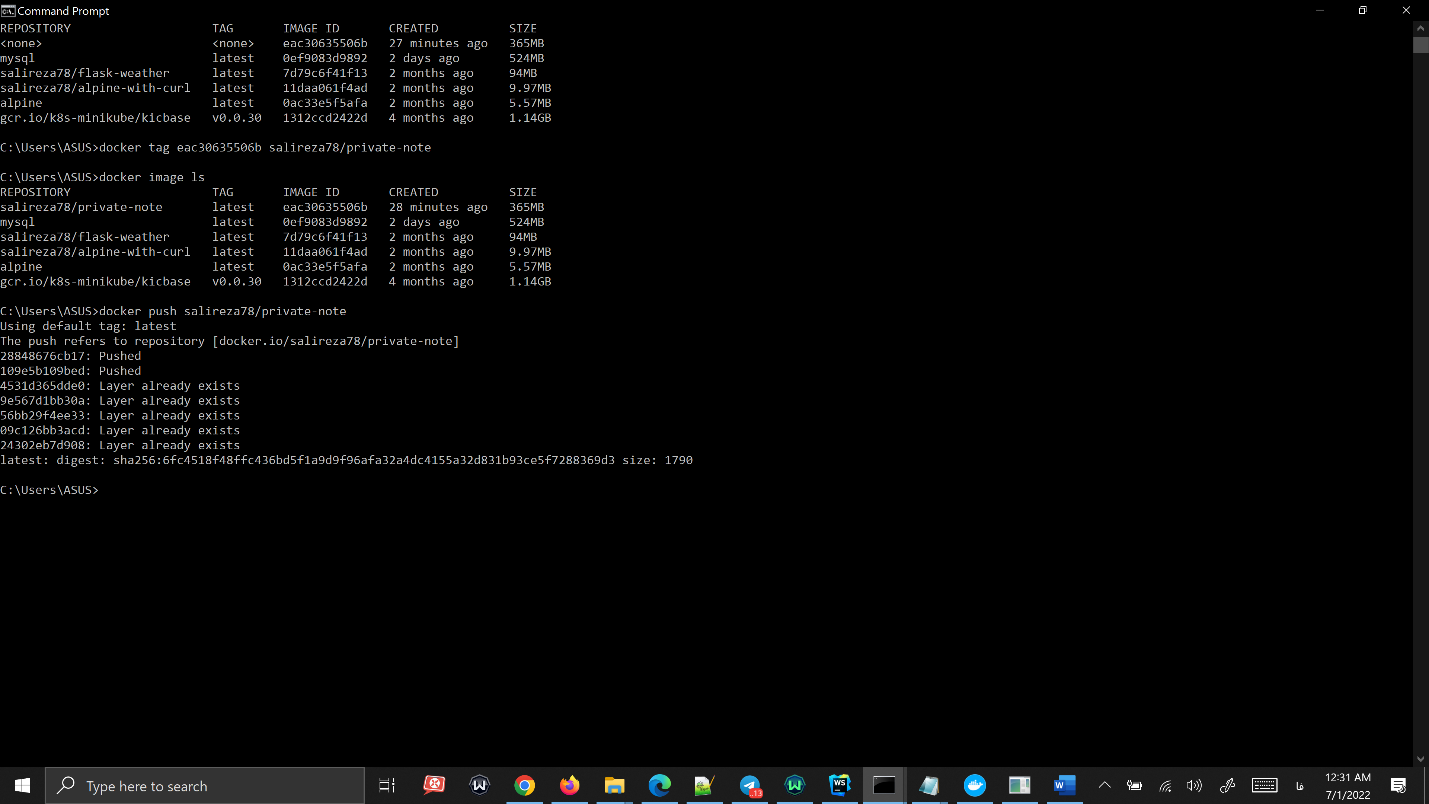


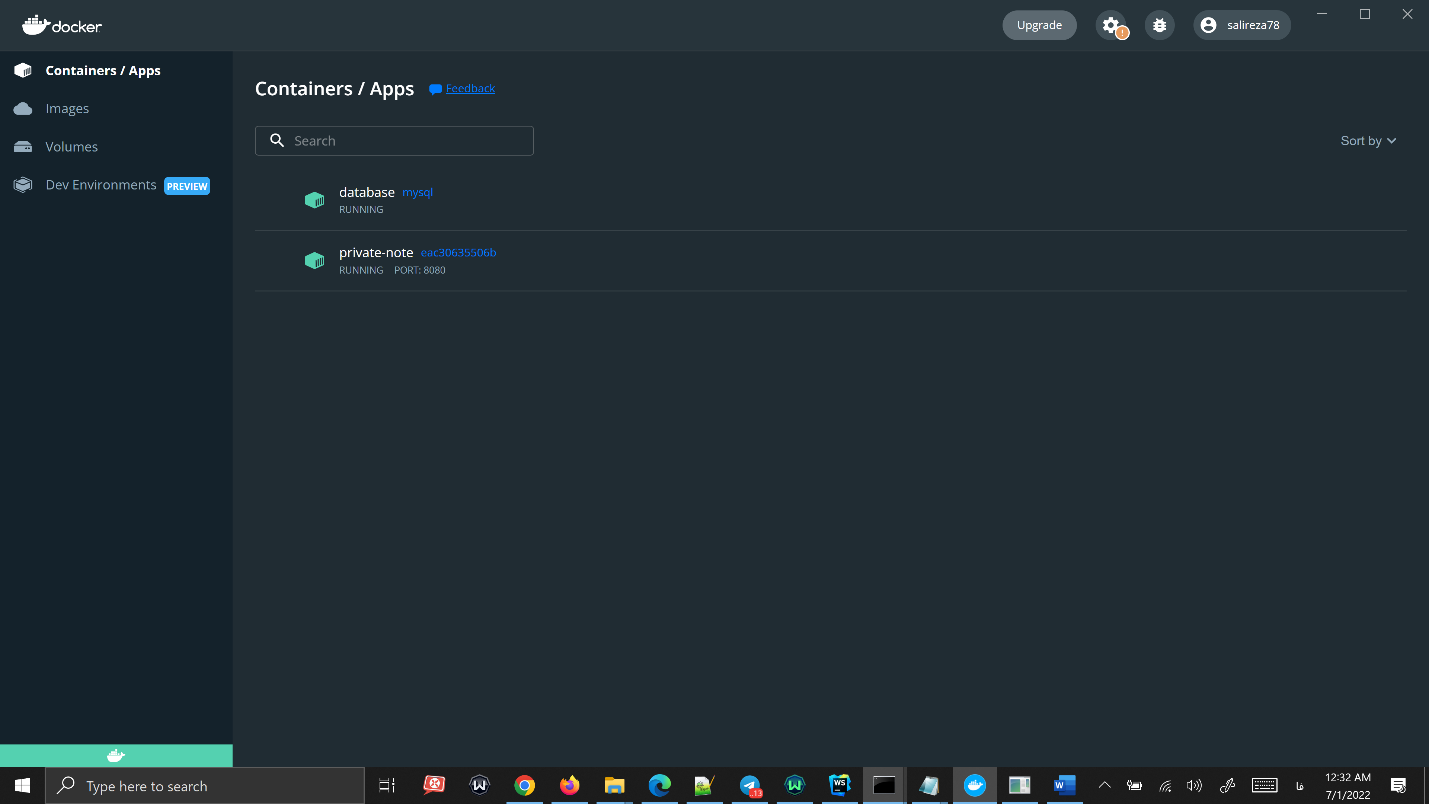
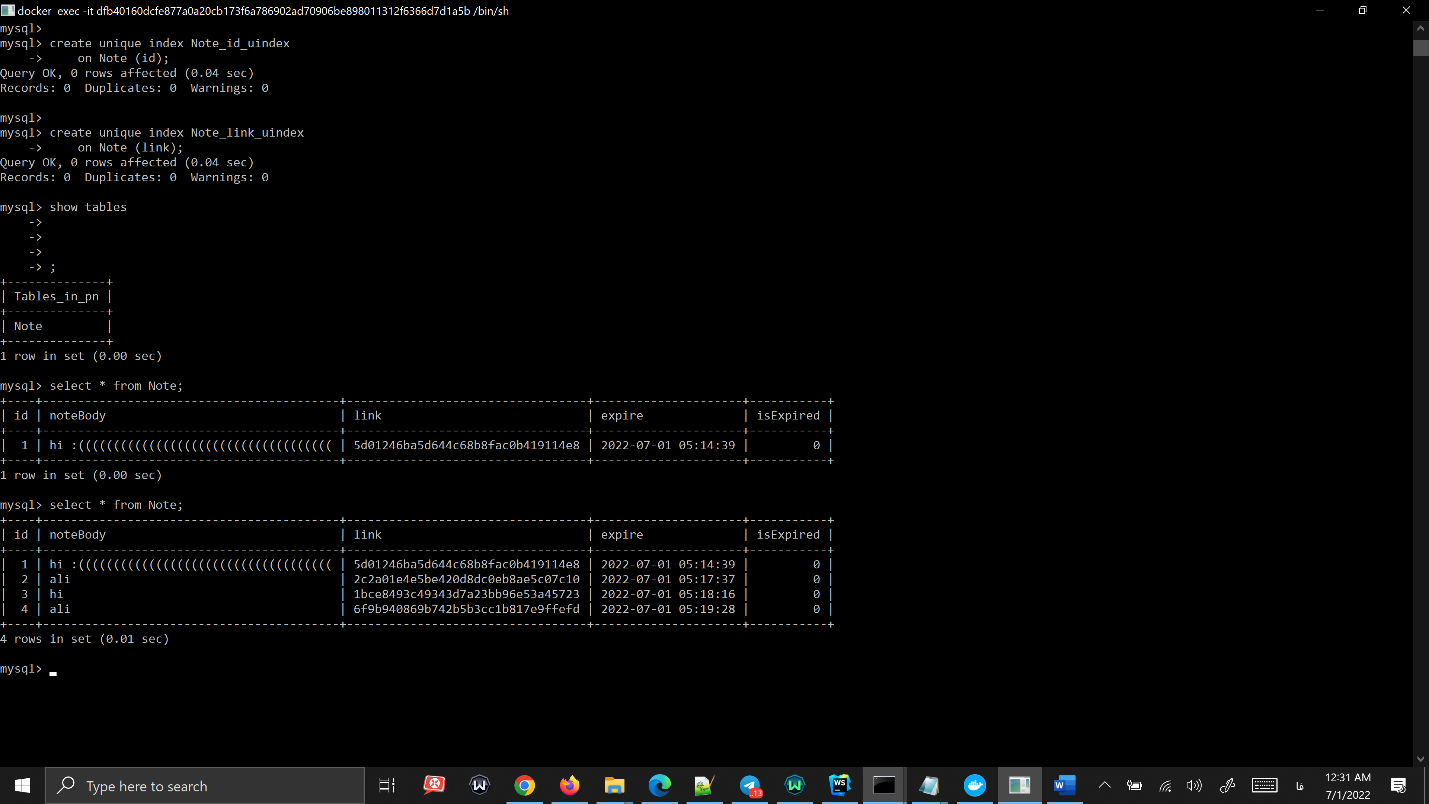
* No-catch فایل‌های build بدردنخور را کش نمی‌کند
* دوتا From داریم (multistage)
* ARG RUNTIME\_DEPS="libcrypto1.1 libssl1.1" برای runtime لازم است

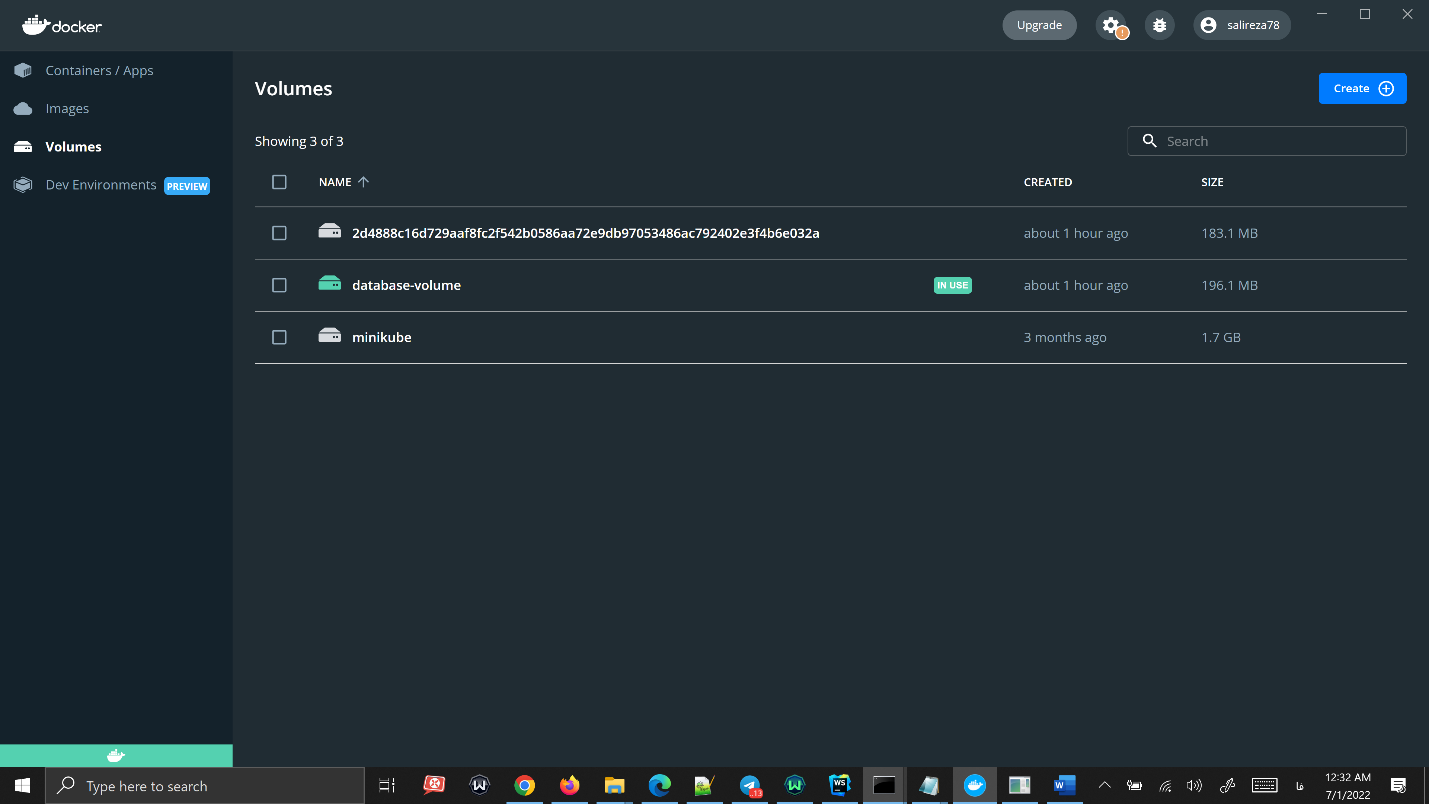
سپس

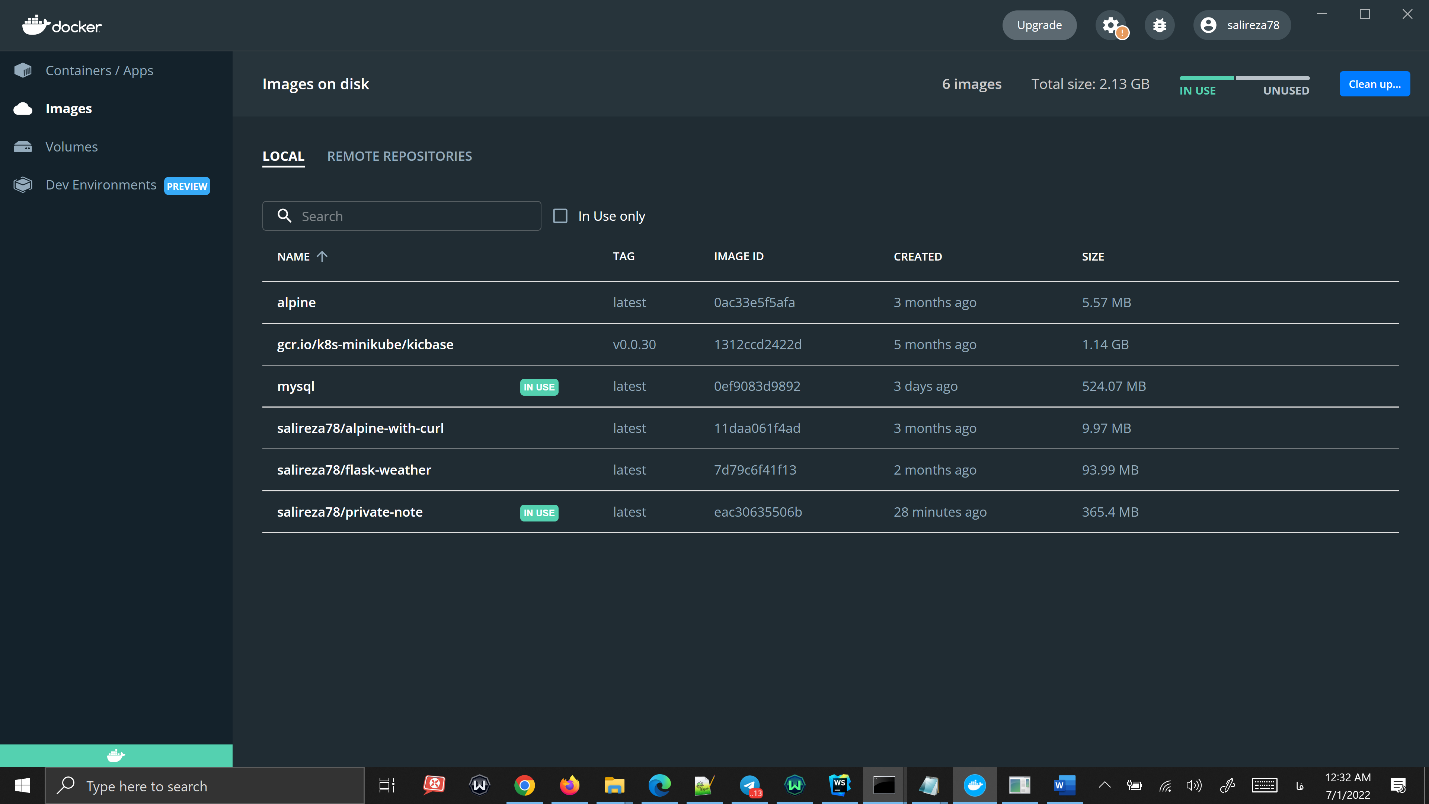
* Docker build .
* Docker tag xxxxxxx salireza78/private-note
* Docker push salireza78/private-note
* docker network create application
* دستور قبل برای ارتباط دیتابیس و سرور در یک شبکه لازم است و باید شبکه ایجاد کنیم
* docker run --env FLASK\_APP=app.py -p 8080:8080 --network application --name private-note salireza78/private-note
* docker pull mysql
* docker volume create database-volume
* دستور قبل برای ایجاد volume
* docker run --env MYSQL\_ROOT\_PASSWORD=secret --network application --name database --volume database-volume:/var/lib/mysql mysql
* mysql -p
* وارد کردن پسورد و ساخت دیتابیس و جدول‌هایش
  + Create datebase pn;
  + Use pn;
  + 











**گام سوم**

With current approach we would have to share both the docker commands, i.e docker command for web application and the docker command for database, with other developers for them to run the application in a Docker setup.  
Consider the scenario where there are more services involved, say Redis, Rabbitmq etc. This approach would very soon become unmanageable.

Ideally there should be a single command to get this entire infrastructure up. That's where Docker compose comes into picture.

Docker compose is a single file similar to Dockerfile. It is written in yaml.

* دستور docker compose up را پس از ساختن فایل docker-compose.yaml می‌زنیم

