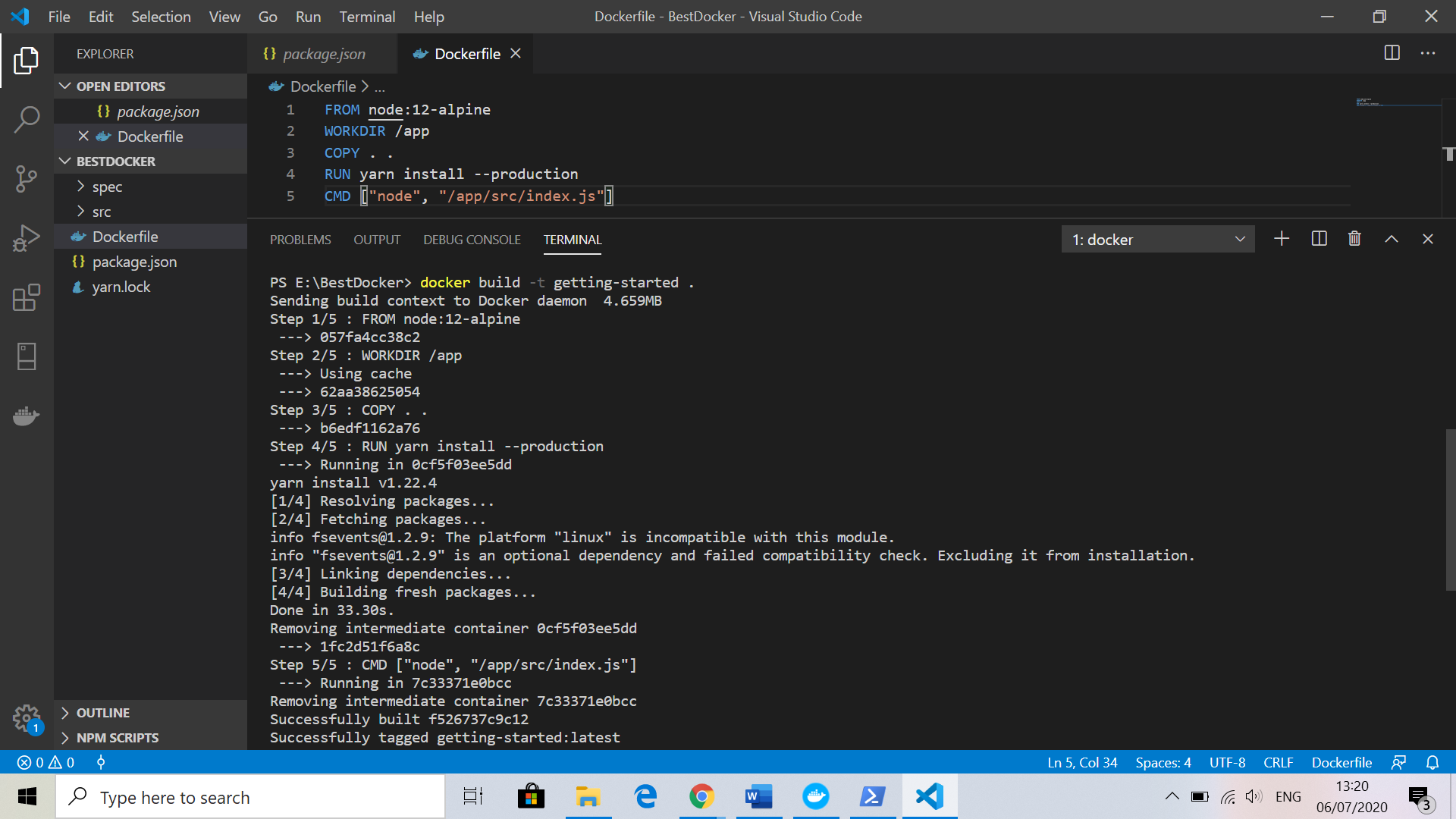
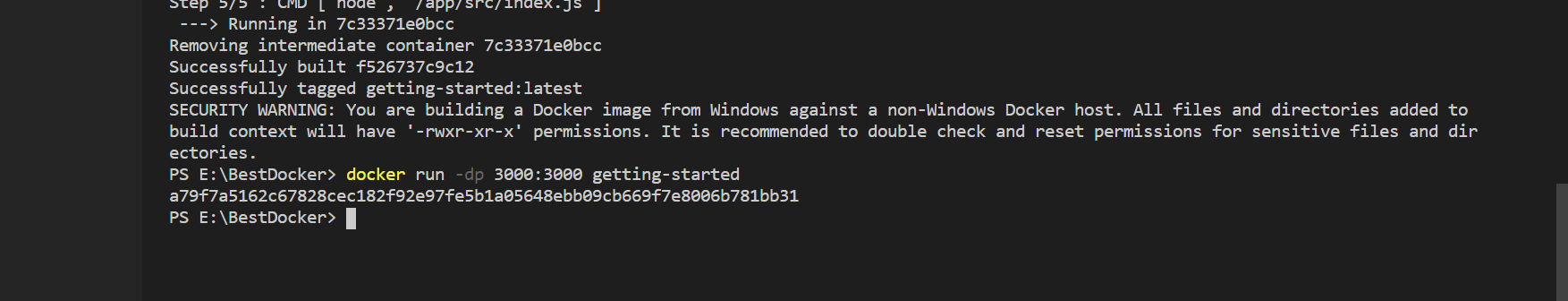


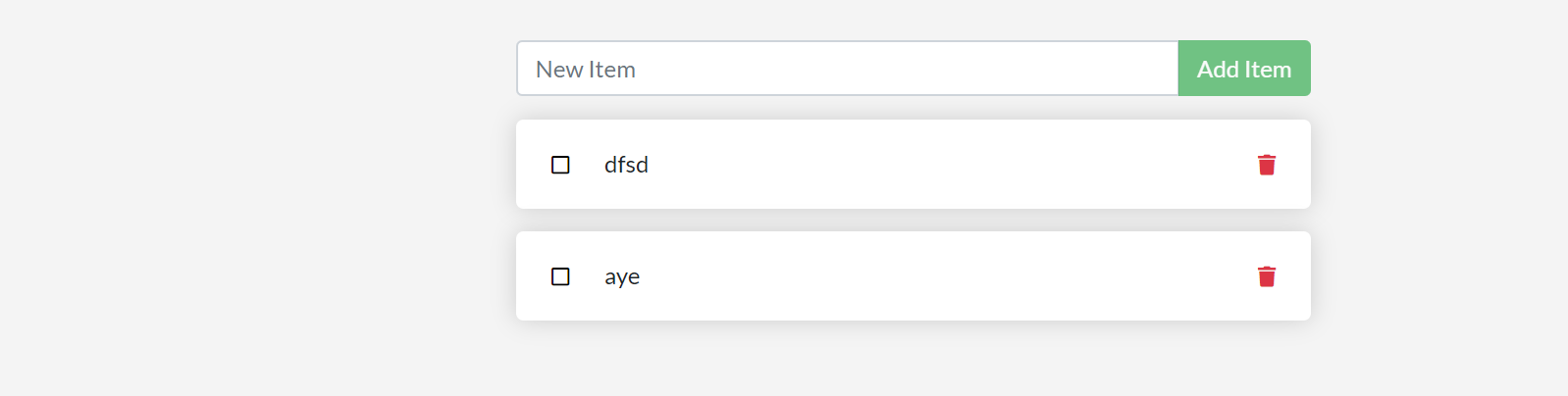
Localhost:80

Create Images: Buat file dengan nama dockerfile dengan content file dibawah ini. Lalu bangun Images container dengan menggunakan docker build. docker build -t getting-started .

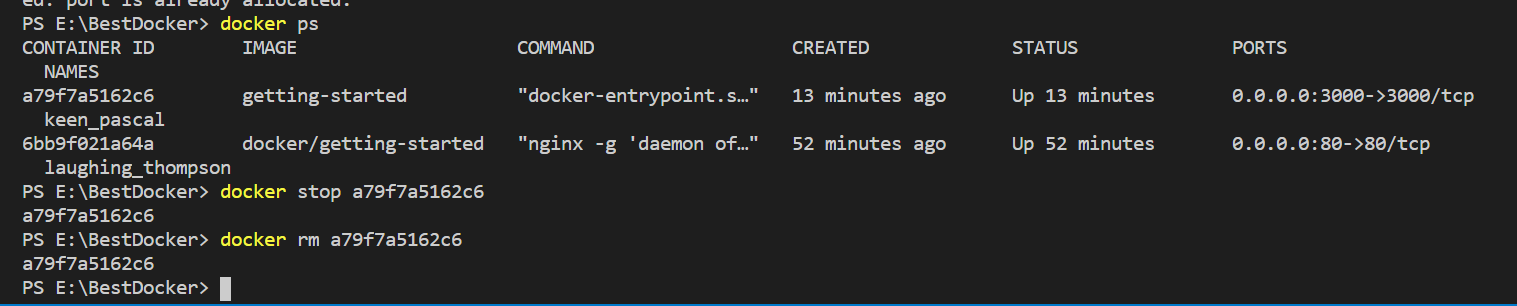




Buka Localhost:3000



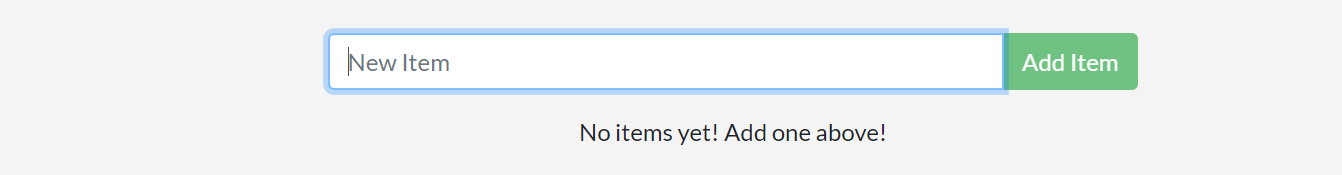
**Hapus Container**



**Update aplikasi container**

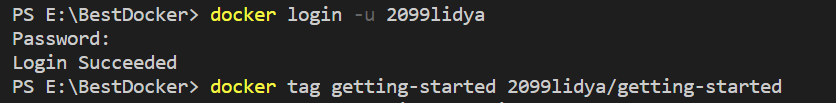
docker run -dp 3000:3000 getting-started

kemudian jalanin lagi <https://localhost:3000>

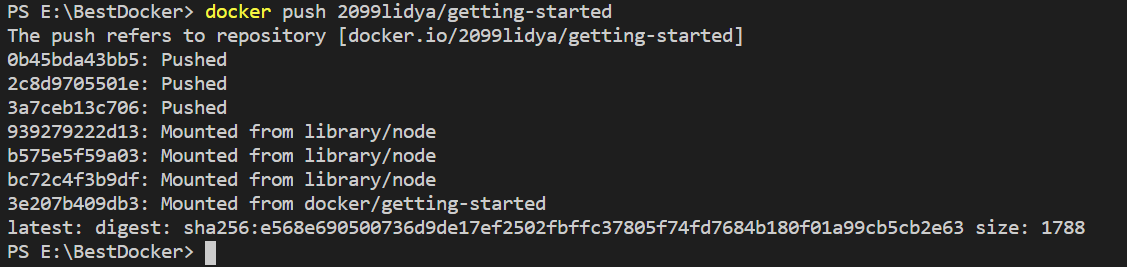


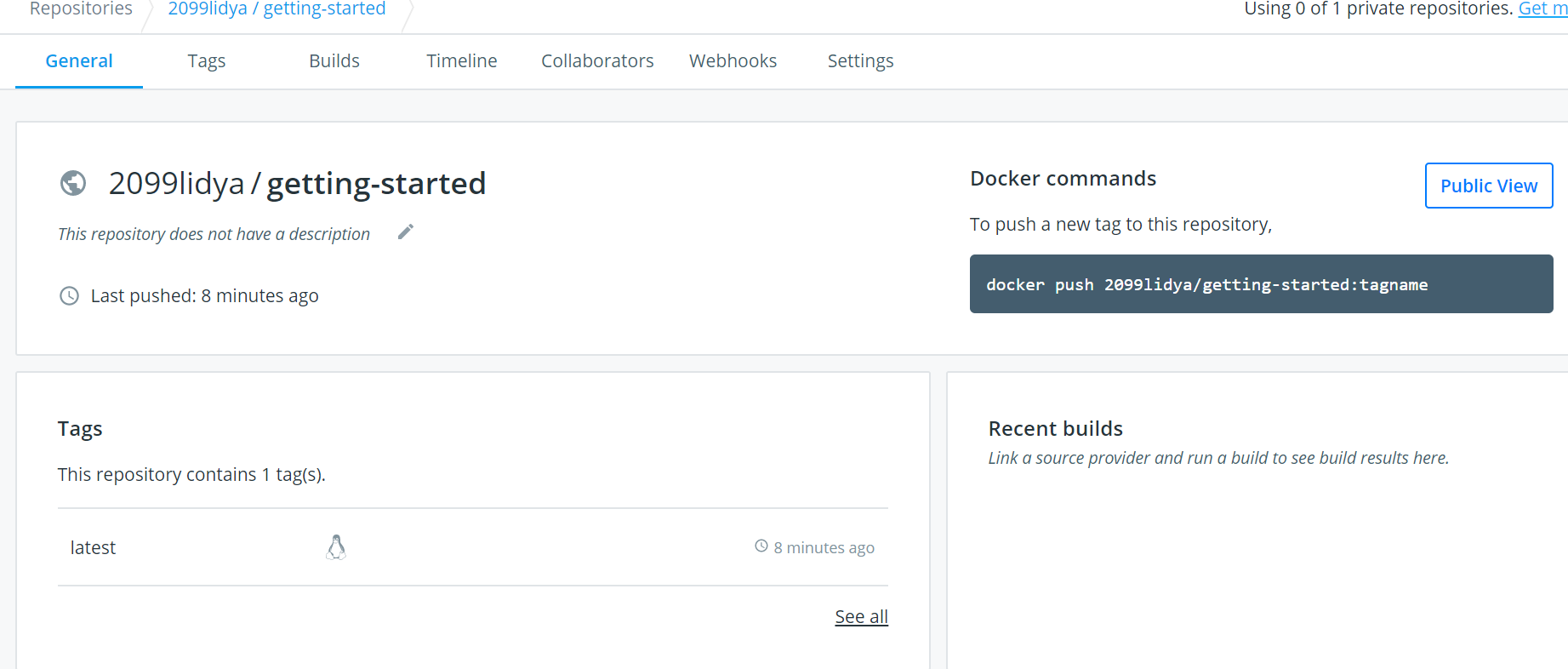
**Sharing APP**

1. **Create repository**
2. **Login to Docker hub**



1. Gunakan docker tag perintah untuk memberi getting-started images nama baru
2. Jika tidak menentukan tag, Docker akan menggunakan tag yang disebut latest. Untuk push



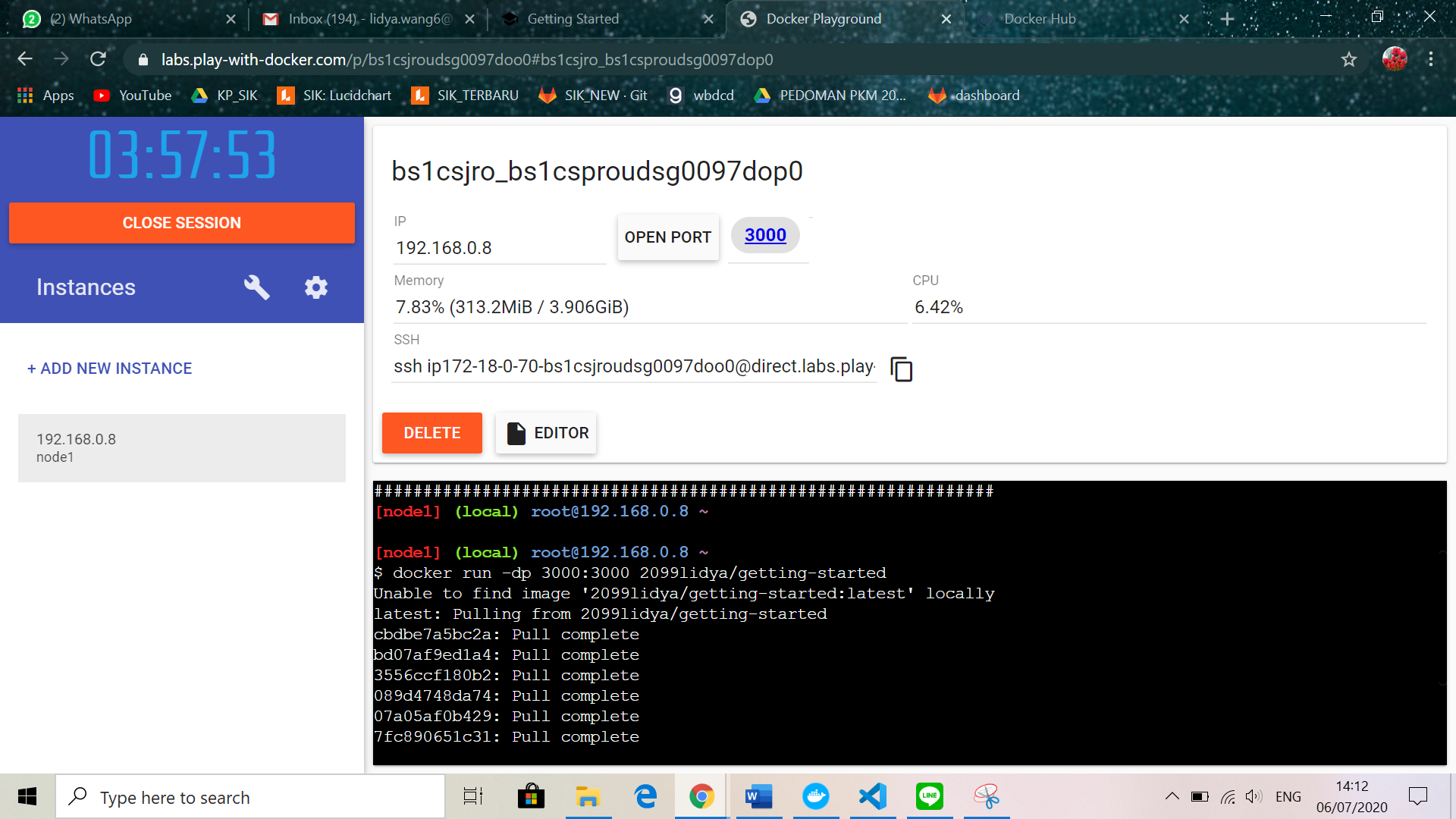


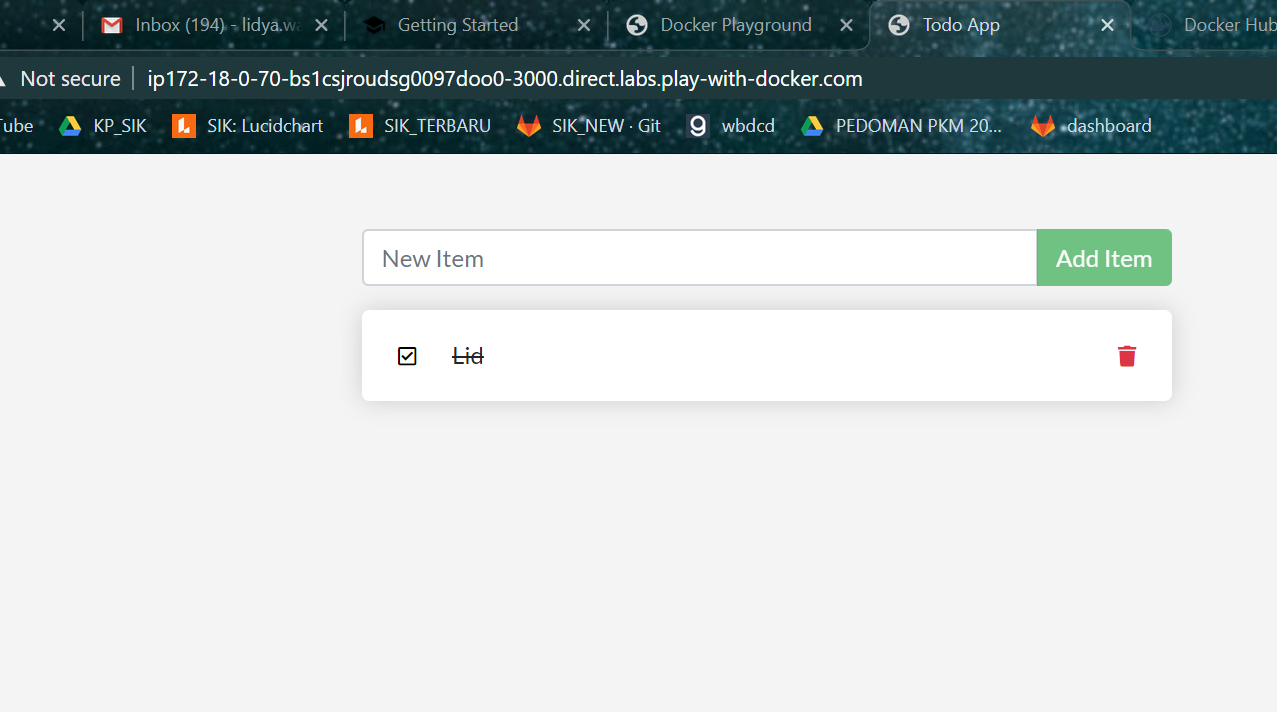
Running our Image on a New Instance

1. **Buka web** [**http://play-with-docker.com/**](http://play-with-docker.com/)
2. **Klik login dan add new instance**
3. **Lalu masukan ke terminal :**

docker run -dp 3000:3000 YOUR-USER-NAME/getting-started

1. **Klik 3000**



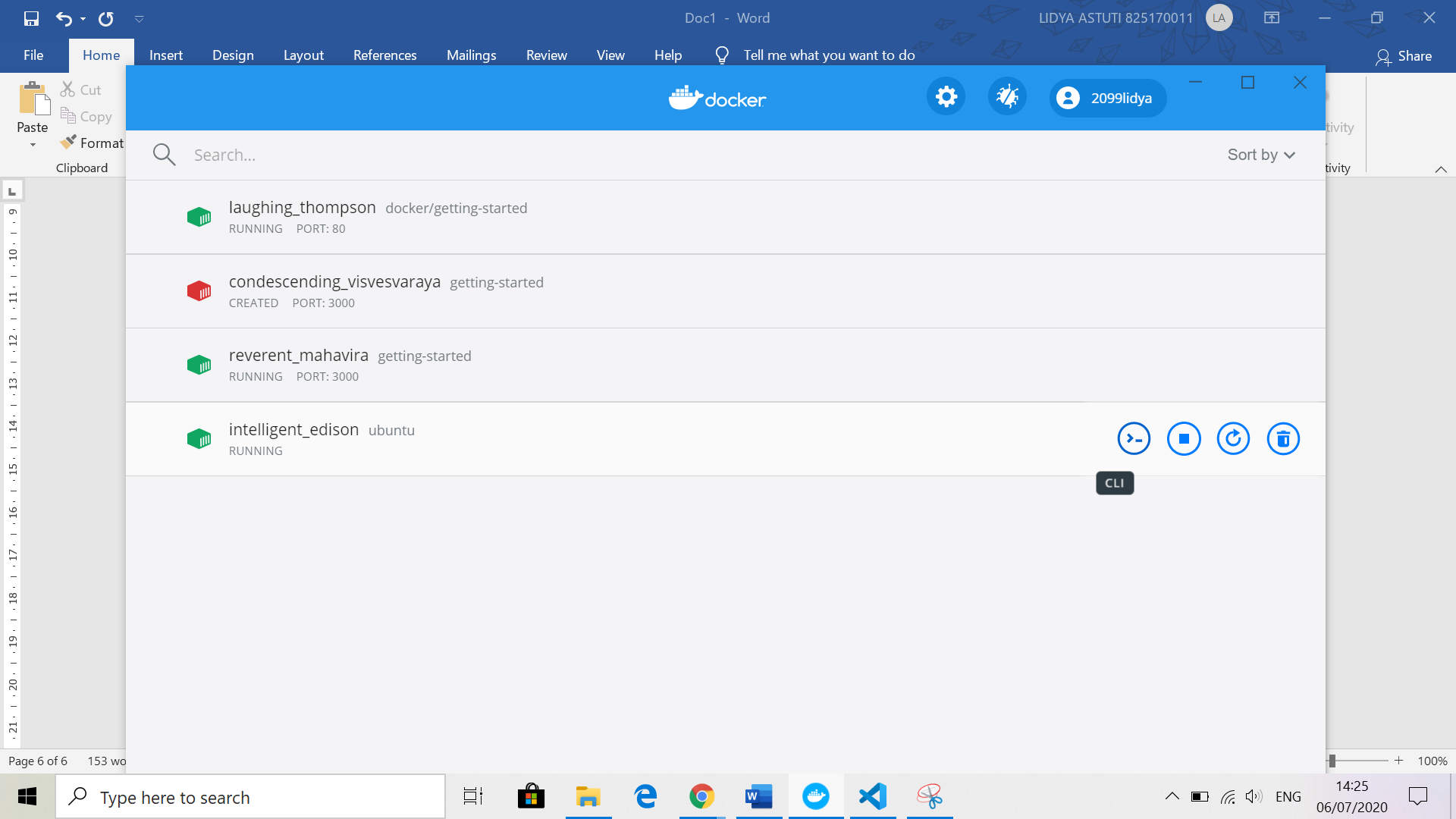


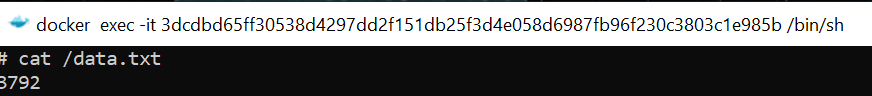
# **Persisting our DB**

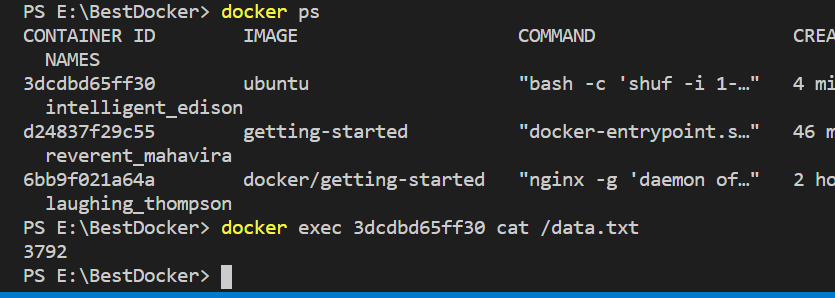
Untuk melihat ini dalam tindakan, kita akan mulai dua container dan membuat file di masing-masing. Yang akan Anda lihat adalah bahwa file yang dibuat dalam satu container tidak tersedia di container lain

1. Mulai ubuntu container yang akan membuat file bernama /data.txt dengan angka acak antara 1 dan 10.000.

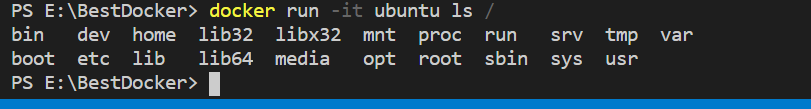
docker run -d ubuntu bash -c "shuf -i 1-10000 -n 1 -o /data.txt && tail -f /dev/null"







**Docker run -it ubuntu ls / : Untuk melihat apakah ada file data.txt . Tidak ada karena ditulis di ruang awal /scratch space hanya untuk container pertama**



**Volume Container**

<http://localhost/tutorial/persisting-our-data/>

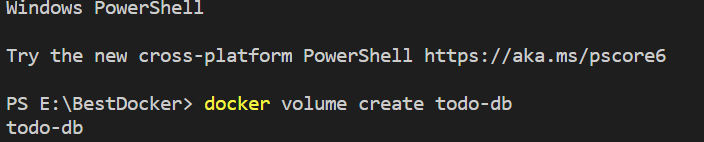
Dengan percobaan sebelumnya, kami melihat bahwa setiap wadah dimulai dari definisi gambar setiap kali dimulai. Sementara wadah dapat membuat, memperbarui, dan menghapus file, perubahan itu hilang ketika wadah dihapus dan semua perubahan diisolasi ke wadah itu. Dengan volume, kita dapat mengubah semua ini.

[Volume](https://docs.docker.com/storage/volumes/) menyediakan kemampuan untuk menghubungkan jalur sistem file tertentu dari wadah kembali ke mesin host. Jika direktori dalam wadah sudah terpasang, perubahan dalam direktori itu juga terlihat pada mesin host. Jika kita memasang direktori yang sama di wadah restart, kita akan melihat file yang sama.

**Named Volume**

Step:

1. Buat volume dengan menggunakan docker volume create



1. Hentikan container aplikasi todo sekali lagi di Dasbor (atau dengan docker rm -f <id>), karena masih berjalan tanpa menggunakan volume persisten.



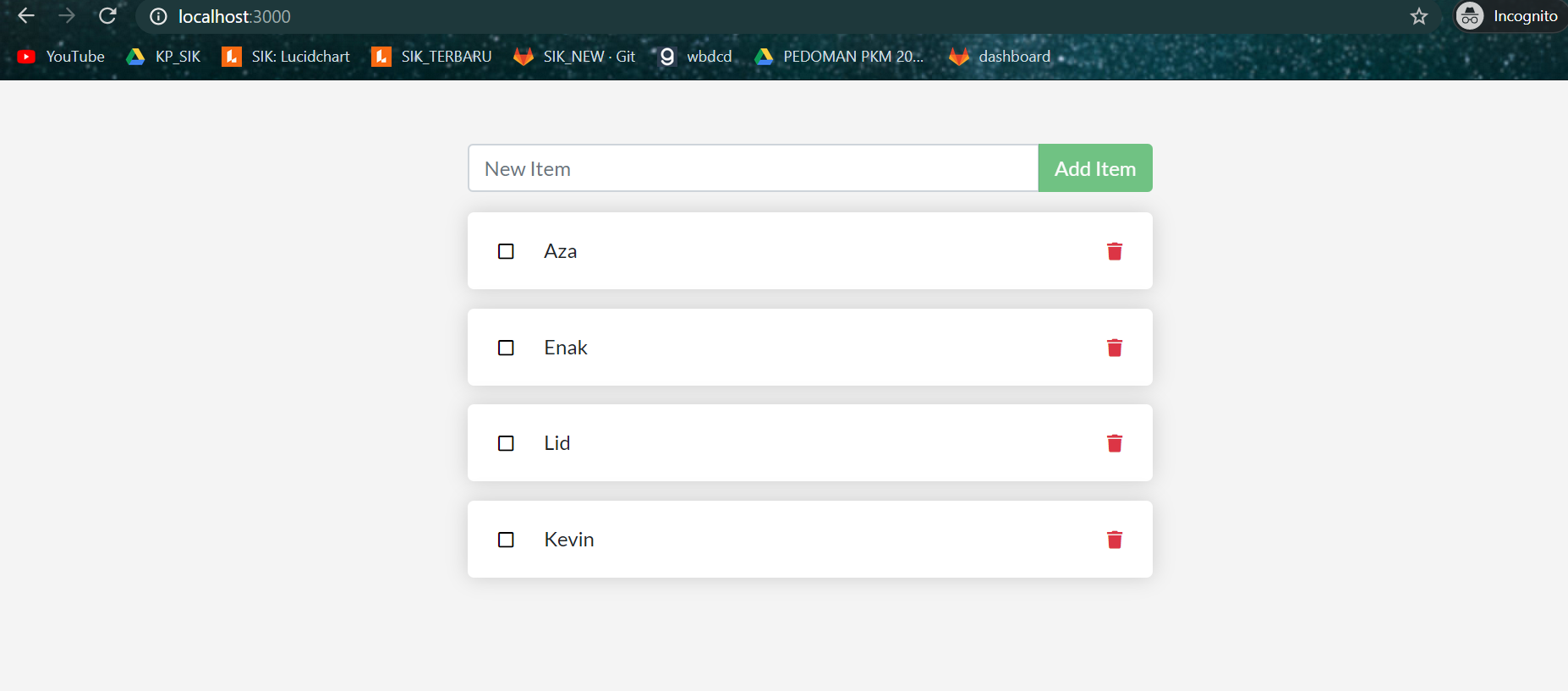
1. Mulai Container aplikasi todo, tetapi tambahkan -v bendera untuk menentukan volume mount. Kami akan menggunakan volume bernama dan mount ke /etc/todos, yang akan menangkap semua file yang dibuat di jalur.

C

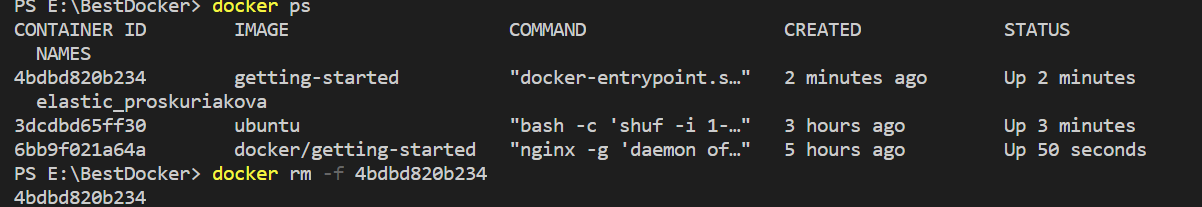
ddocker run -dp 3000:3000 -v todo-db:/etc/todos getting-started



1. Buka aplikasi dan tambahkan banyak item



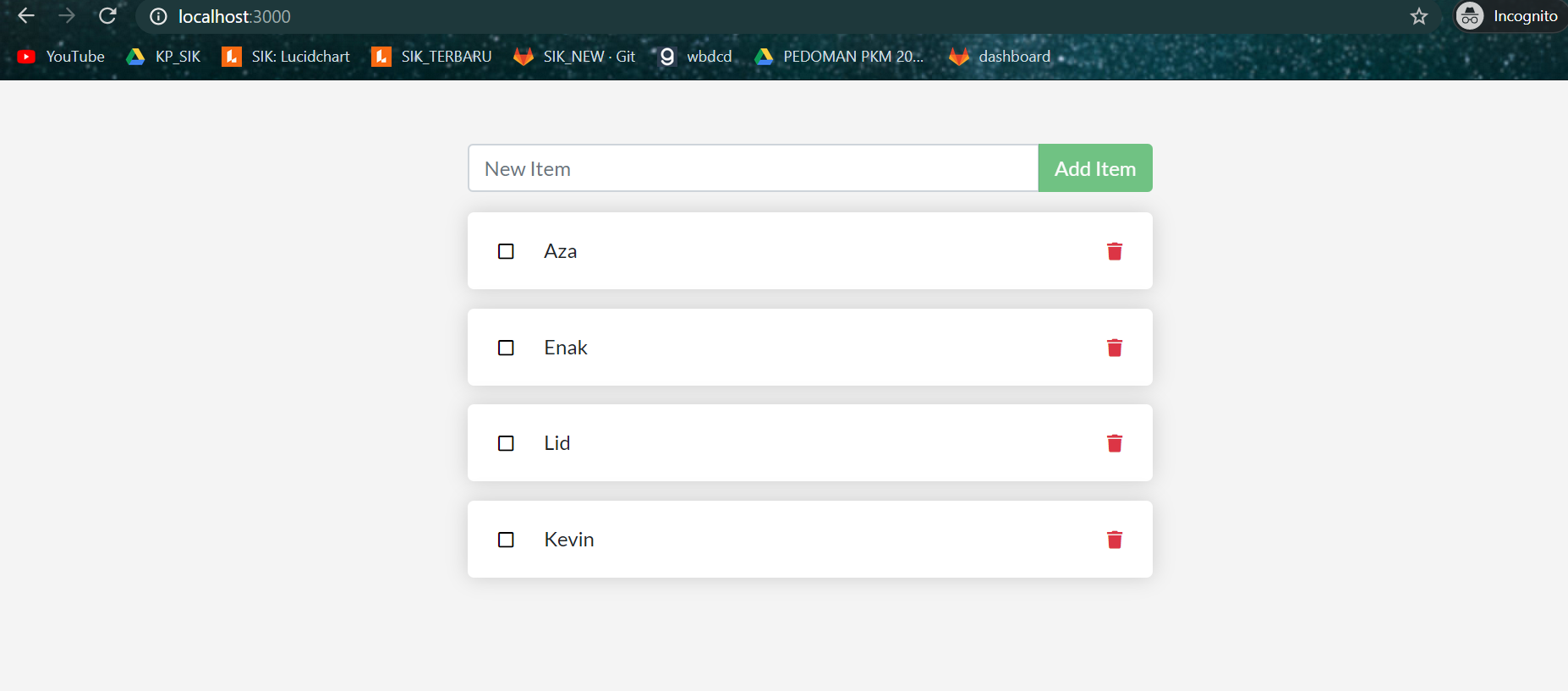
1. Hapus Container untuk aplikasi todo. Gunakan Dasbor atau docker psuntuk mendapatkan ID dan kemudian docker rm -f <id>untuk menghapusnya.



1. Mulai container baru

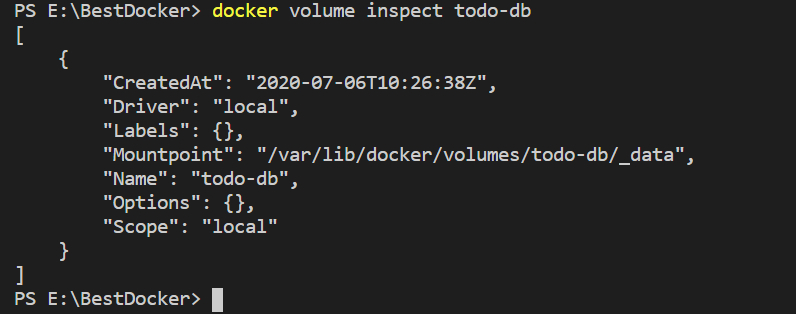


1. Buka aplikasi, maka items-item masih tersimpan



## Diving into our Volume

1. Cek Docker simpan data menggunakan named volume



Mountpoint adalah lokasi aktual pada disk tempat data disimpan.

# **Using Bind Mounts**

Dengan **bind mounts** , mengontrol titik mount yang tepat pada host. Kita bisa menggunakan ini untuk bertahan data, tetapi sering digunakan untuk memberikan data tambahan ke dalam Container. Saat mengerjakan aplikasi, kita bisa menggunakan bind mount untuk memasang kode sumber kita ke dalam Container untuk membiarkannya melihat perubahan kode, merespons, dan biarkan kita melihat perubahannya segera

## Starting a Dev-Mode Container

* Pasang kode sumber kami ke dalam wadah
* Instal semua dependensi, termasuk dependensi "dev"
* Mulai nodemon untuk melihat perubahan sistem file

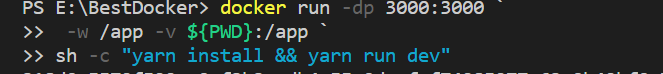
1. Pastikan Anda tidak memiliki getting-startedwadah sebelumnya yang berjalan.
2. Jalankan perintah berikut (ganti \karakter dengan `di Windows PowerShell). Kami akan menjelaskan apa yang terjadi setelahnya:

docker run -dp 3000:3000 \

-w /app -v ${PWD}:/app \

node:12-alpine \

sh -c "yarn install && yarn run dev"



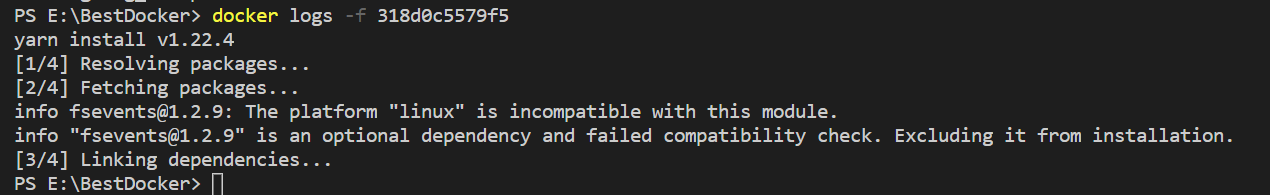
\*nb:w/app directory kerja

-v ${PWD}:/app- bind mount direktori saat ini dari host dalam Container ke /appdirektori

node:12-alpine- Images yang akan digunakan

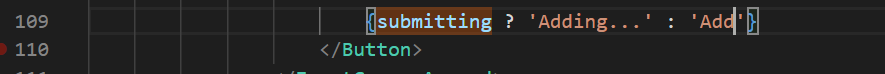
yarn install untuk menginstal semua dependensi dan kemudian berjalan yarn run dev

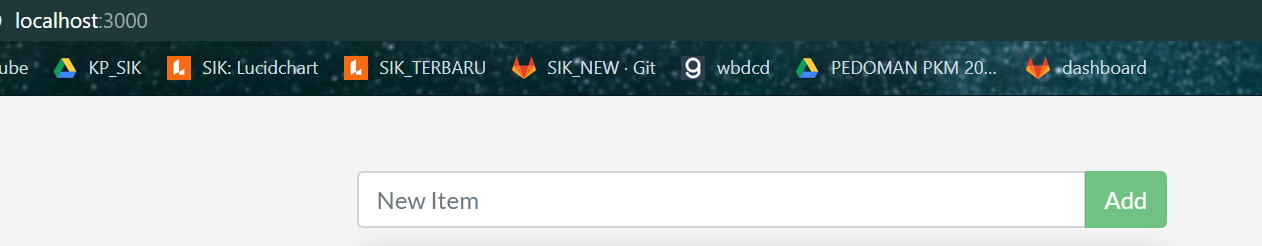
1. menonton log menggunakan docker logs -f <container-id>



Untuk keluar log ctrl+c

1. Sekarang, mari kita buat perubahan ke aplikasi. Dalam src/static/js/app.jsfile tersebut, mari kita ubah tombol "Add Item" menjadi “Add”.





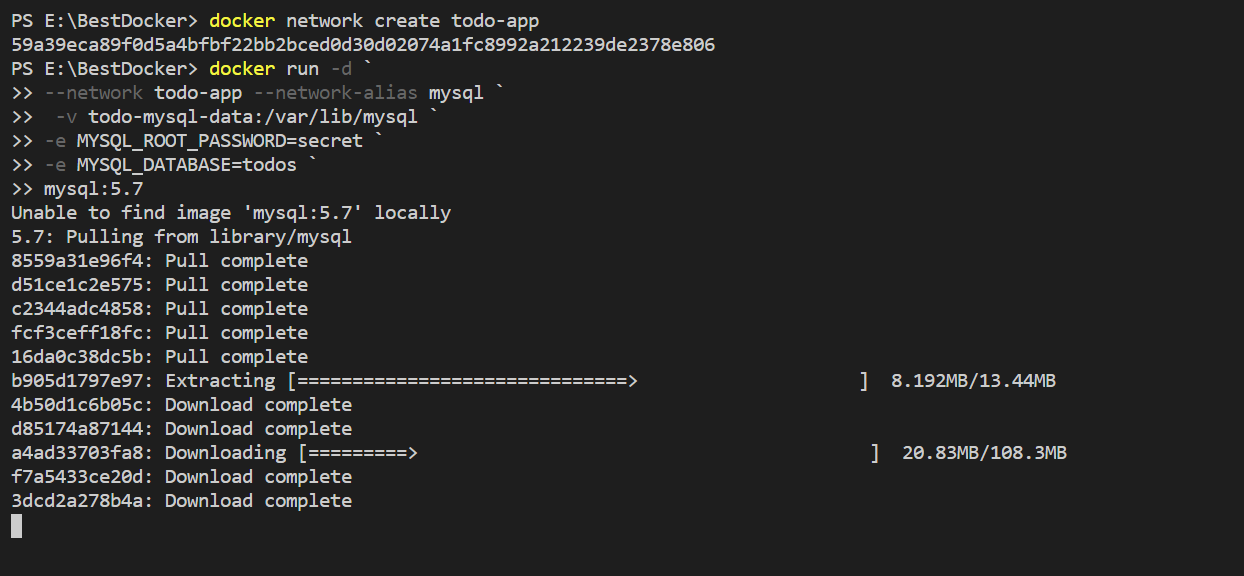
Container Networking

Jika dua container berada dijaringan yang sama, maka dapat saling berbicara. Jika tidak, tidak bisa.

Step:

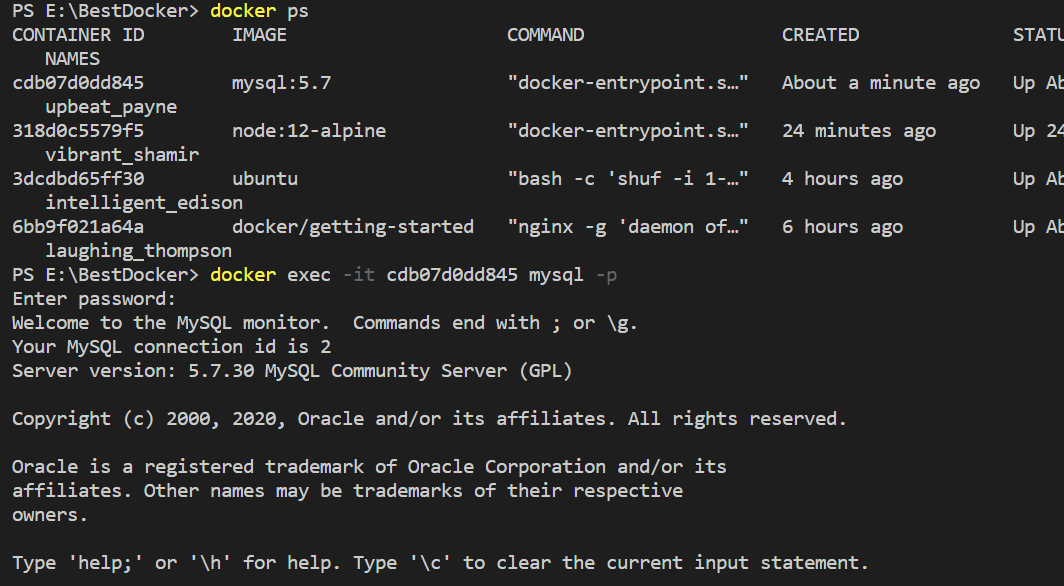
* 1. Create networking: docker network create todo-app
  2. Mulai container Mysql dan pasang networking :
* docker run -d \
* --network todo-app --network-alias mysql \
* -v todo-mysql-data:/var/lib/mysql \
* -e MYSQL\_ROOT\_PASSWORD=secret \
* -e MYSQL\_DATABASE=todos \
* mysql:5.7

Ganti \ jadi `



3.Untuk konfirmasi database telah ada dan berjalan, sambungkan ke basisdata dan verifikasi terhubung

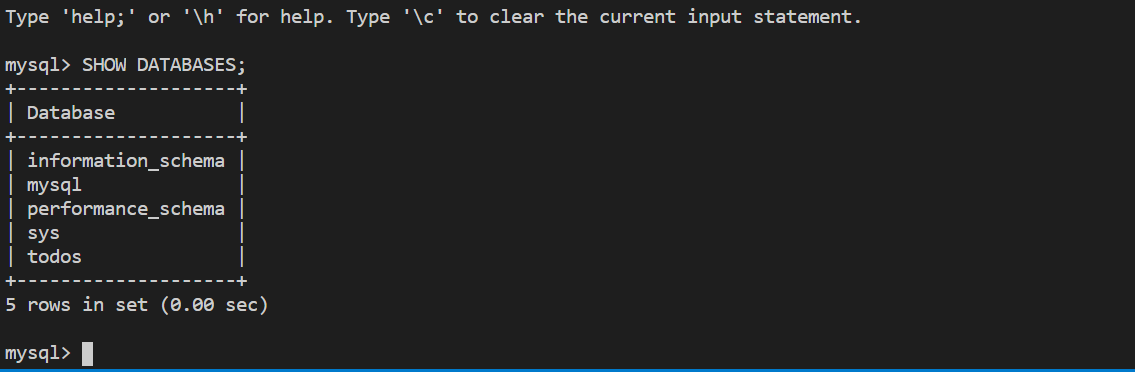
docker exec -it <mysql-container-id> mysql -p



Password: secret

4.Di shell Mysql, lihat database dan verifikasi untuk lihat todos database

mysql> SHOW DATABASES;



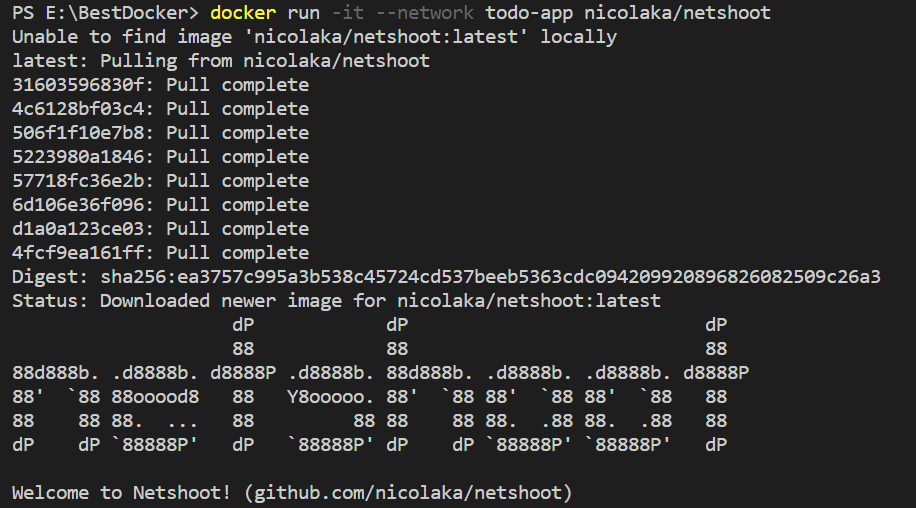
Connecting to mysql

Jika kita menjalankan Container lain di jaringan yang sama, bagaimana kita menemukan Container (ingat setiap Container memiliki alamat IP sendiri)?

Step:

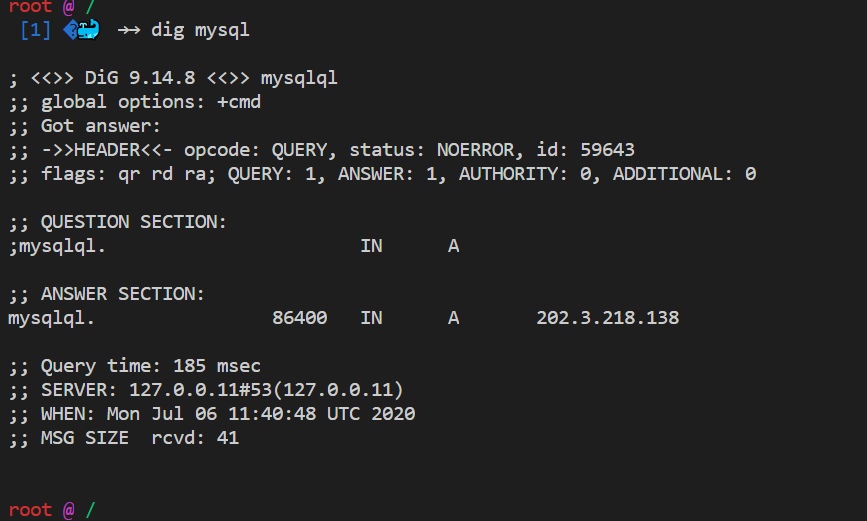
* 1. Mulai Container baru menggunakan Imagesnicolaka / netshoot. Pastikan untuk menghubungkannya ke jaringan yang sama.

docker run -it --network todo-app nicolaka/netshoot

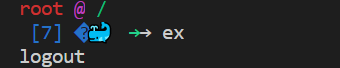


* 1. Di dalam containernya, gunakan perintah dig untuk sebagai alat DNS . Disini akan ada alamat host mysql untuk alamat IP

dig mysql



Dalam "**Answer Section**", Anda akan melihat A catatan untuk mysqlyang diputuskan 202.3.218. Meskipun mysqlbiasanya bukan nama host yang valid, Docker dapat menyelesaikannya ke alamat IP wadah yang memiliki alias jaringan itu (ingat --network-aliasbendera yang kami gunakan sebelumnya?)



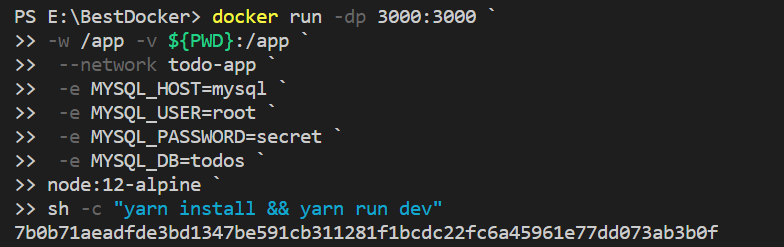
Exit untuk logout

**Running Todo Aplikasi with MYSQL**

**STEP:**

**Pastikan yang container lama yang node:12-alpine distop running**

1. Kami akan menentukan masing-masing variabel lingkungan di atas, serta menghubungkan Container ke jaringan aplikasi kami (ganti \karakter dengan `di Windows PowerShell).



docker run -dp 3000:3000 \

-w /app -v ${PWD}:/app \

--network todo-app \

-e MYSQL\_HOST=mysql \

-e MYSQL\_USER=root \

-e MYSQL\_PASSWORD=secret \

-e MYSQL\_DB=todos \

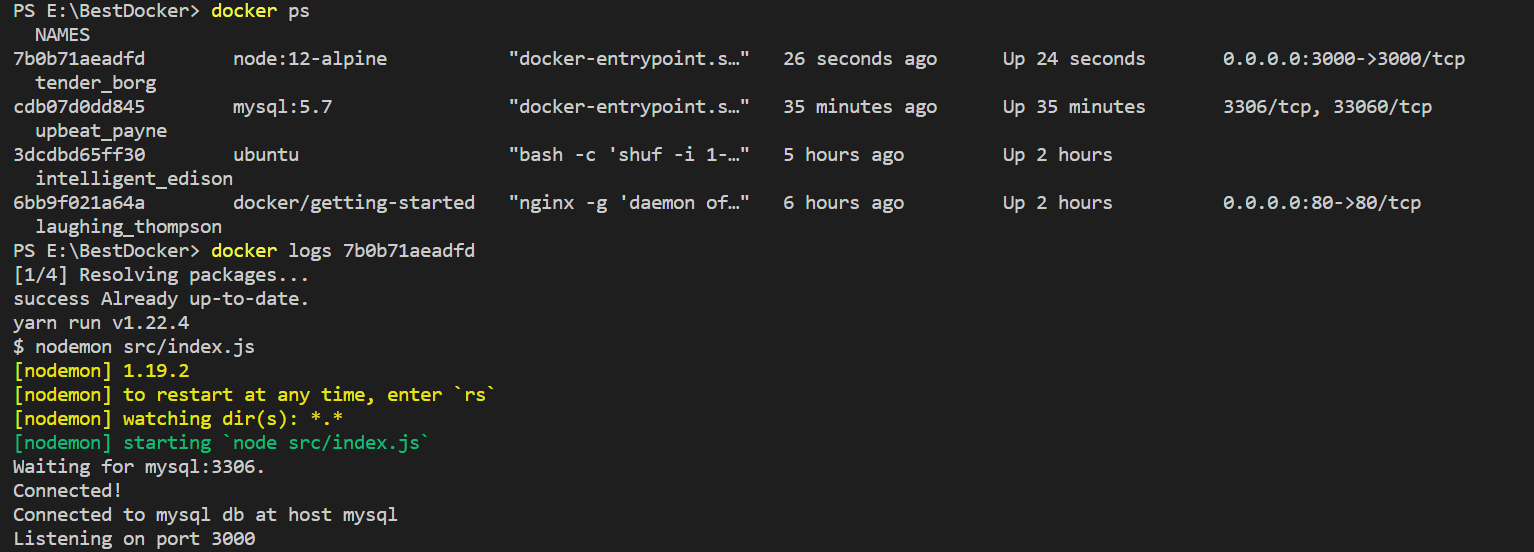
node:12-alpine \

sh -c "yarn install && yarn run dev"

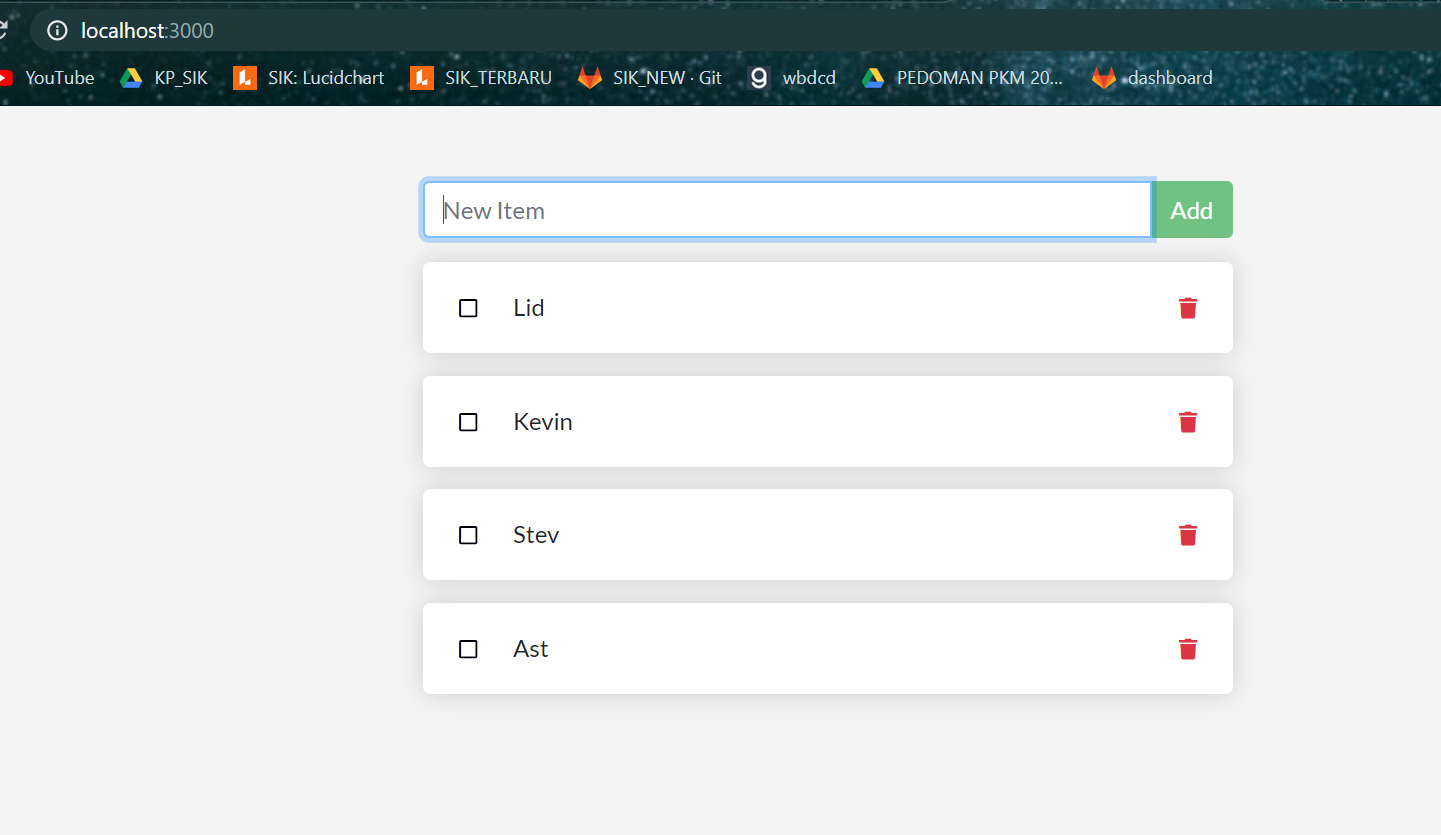
**ganti \ jadi `**

1. **Untuk lihat log penampung :**

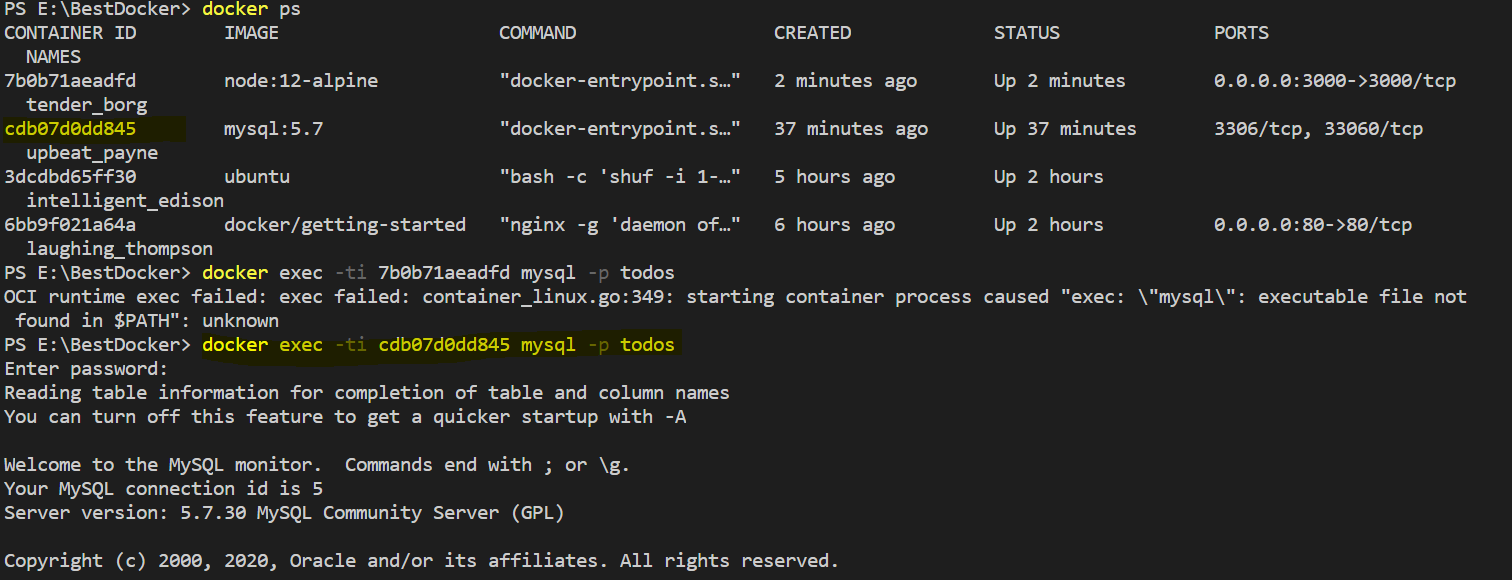
**docker logs idcontainer**



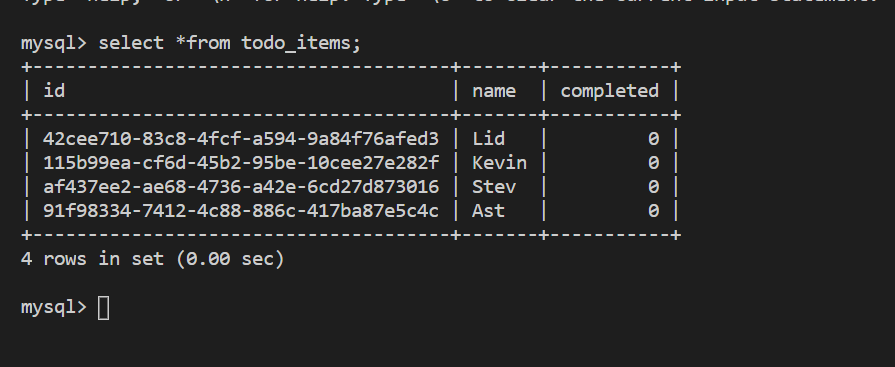
1. Buka aplikasi di browser Anda dan tambahkan beberapa item ke daftar todo Anda.



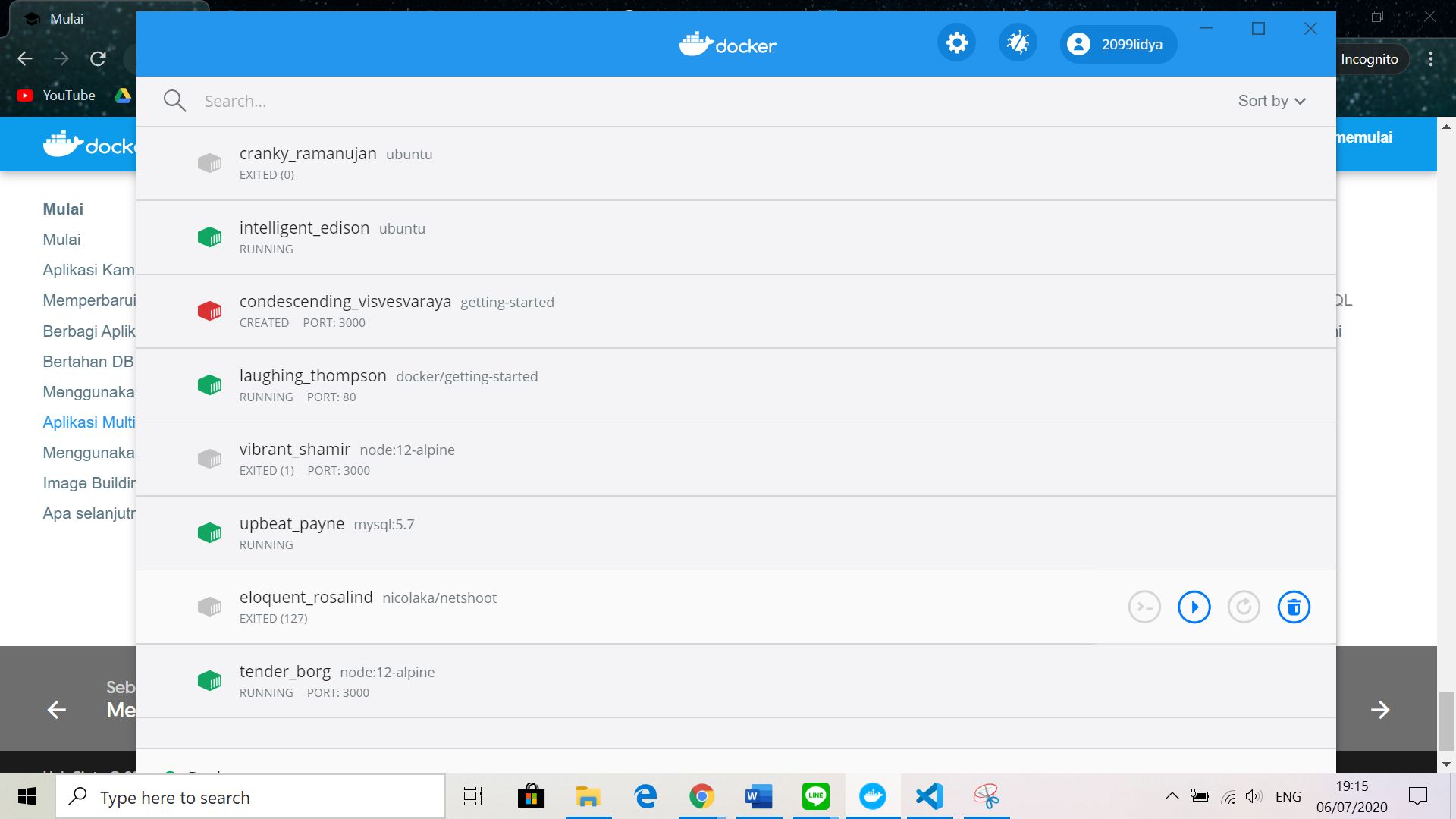
1. **Hubungkan ke database mysql dan buktikan item ada tertulis didatabase. Password:secret**



1. **Lalu jalankan select \*from todo\_items;**

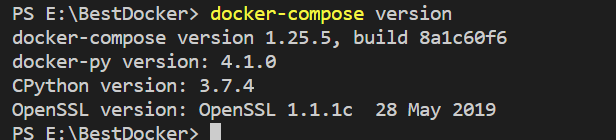


**Hasil seperti diatas**



**CEK DOCKER COMPOSE VERSION**

docker-compose version



**Defining App Service**

docker run -dp 3000:3000 \

-w /app -v ${PWD}:/app \

--network todo-app \

-e MYSQL\_HOST=mysql \

-e MYSQL\_USER=root \

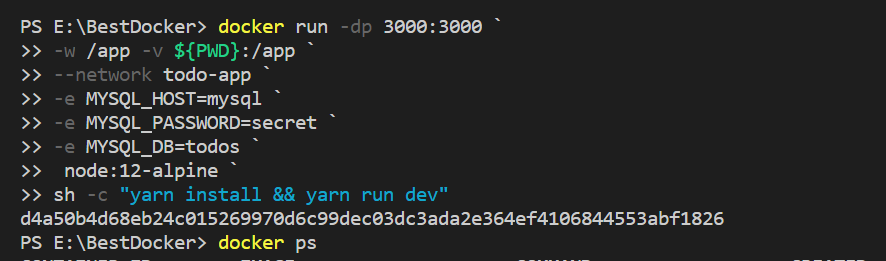
-e MYSQL\_PASSWORD=secret \

-e MYSQL\_DB=todos \

node:12-alpine \

sh -c "yarn install && yarn run dev"

\ ganti jadi `



**Defining the Mysql Services**

docker run -d \

--network todo-app --network-alias mysql \

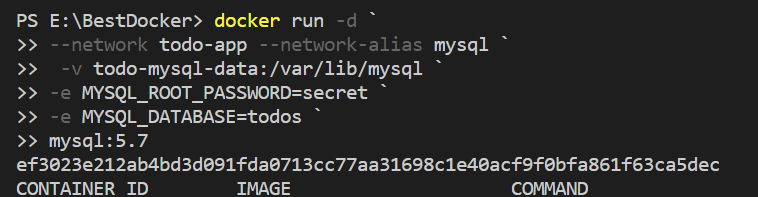
-v todo-mysql-data:/var/lib/mysql \

-e MYSQL\_ROOT\_PASSWORD=secret \

-e MYSQL\_DATABASE=todos \

mysql:5.7

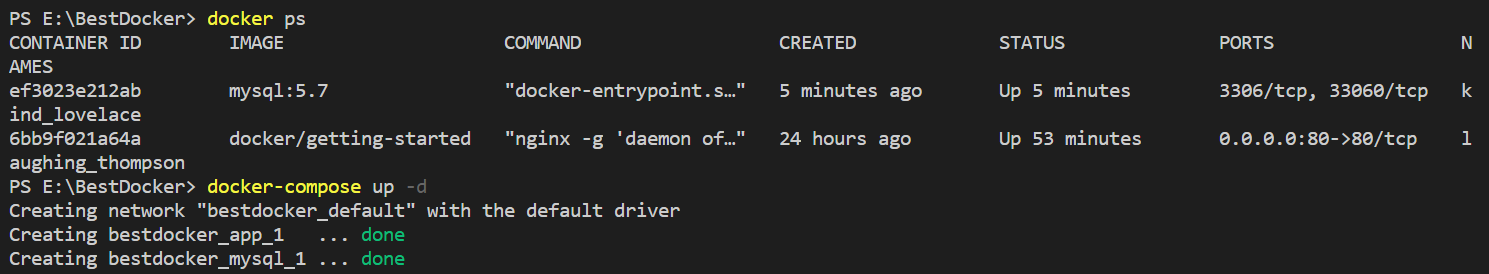
ganti \ jadi `



**Running Application Stack**

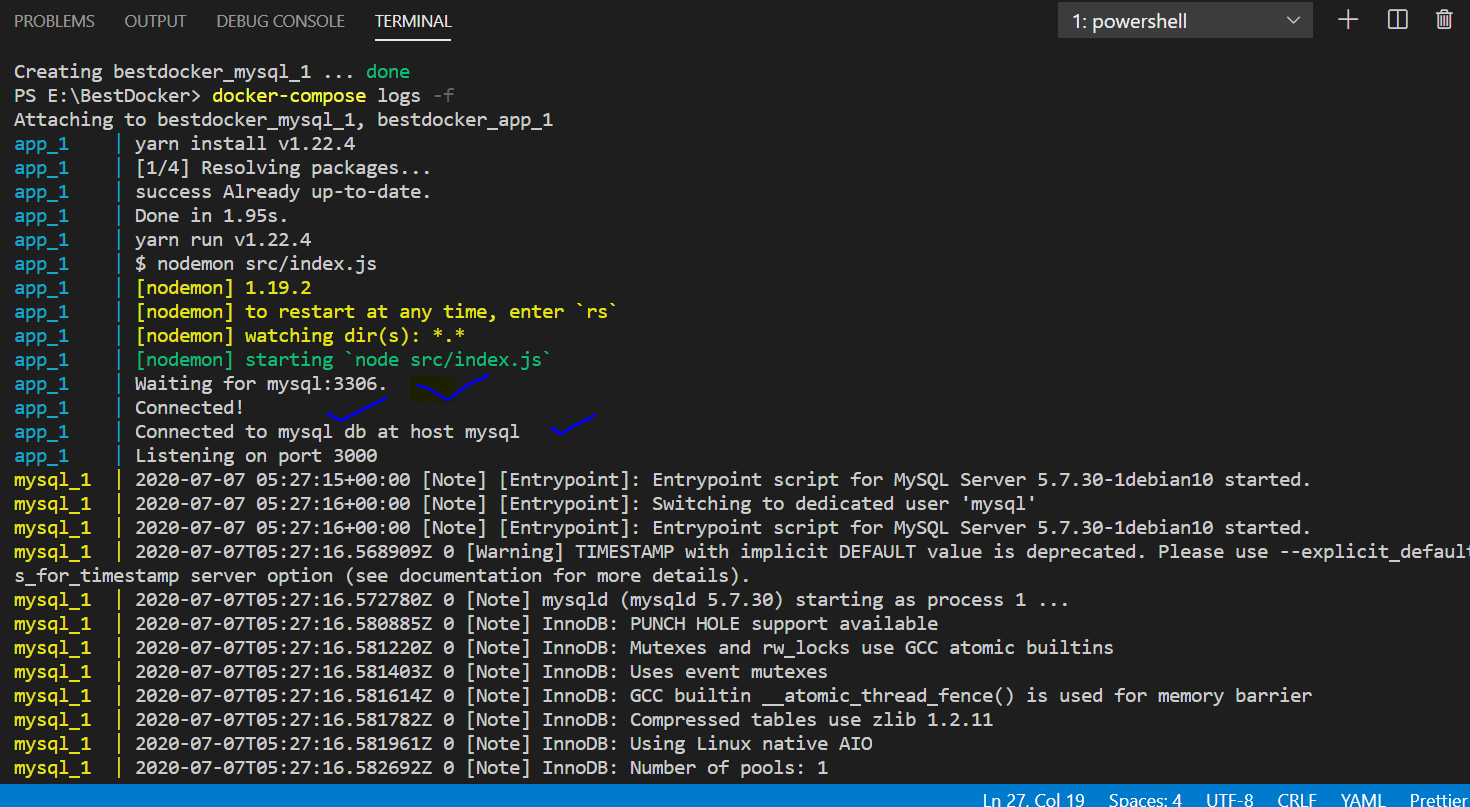
1. Pastikan tidak ada salinan lain dari aplikasi / db berjalan terlebih dahulu ( docker ps dan docker rm -f <ids>).
2. Untuk menggabungkan stack/tumpuan aplikasi, maka menggunakan perintah:

docker-compose up -d



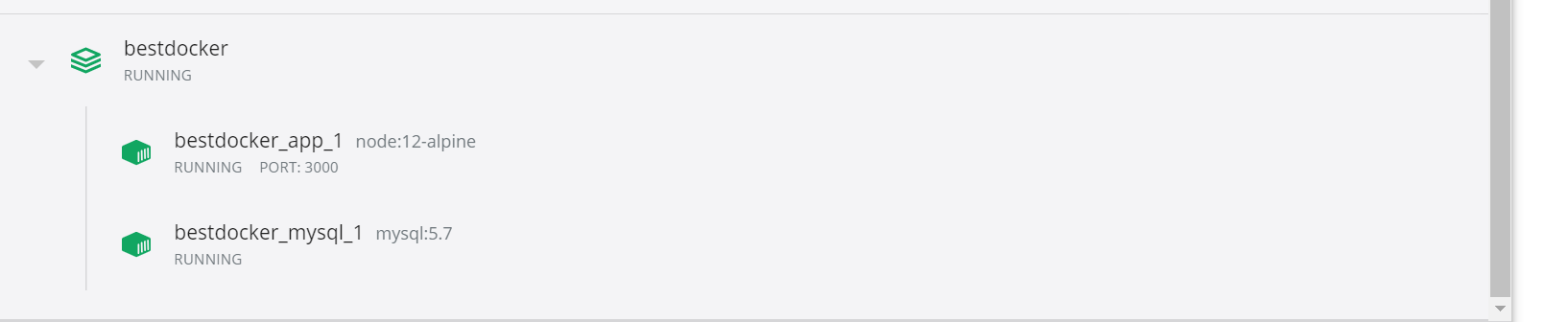
**Volume diatas telah dibuat di network.**

1. Dapat melihat log menggunakan docker-compose logs -f perintah. Anda akan melihat log dari masing-masing layanan disatukan menjadi satu aliran.





1. **Melihat aplikasi stack di Docker dashboard**

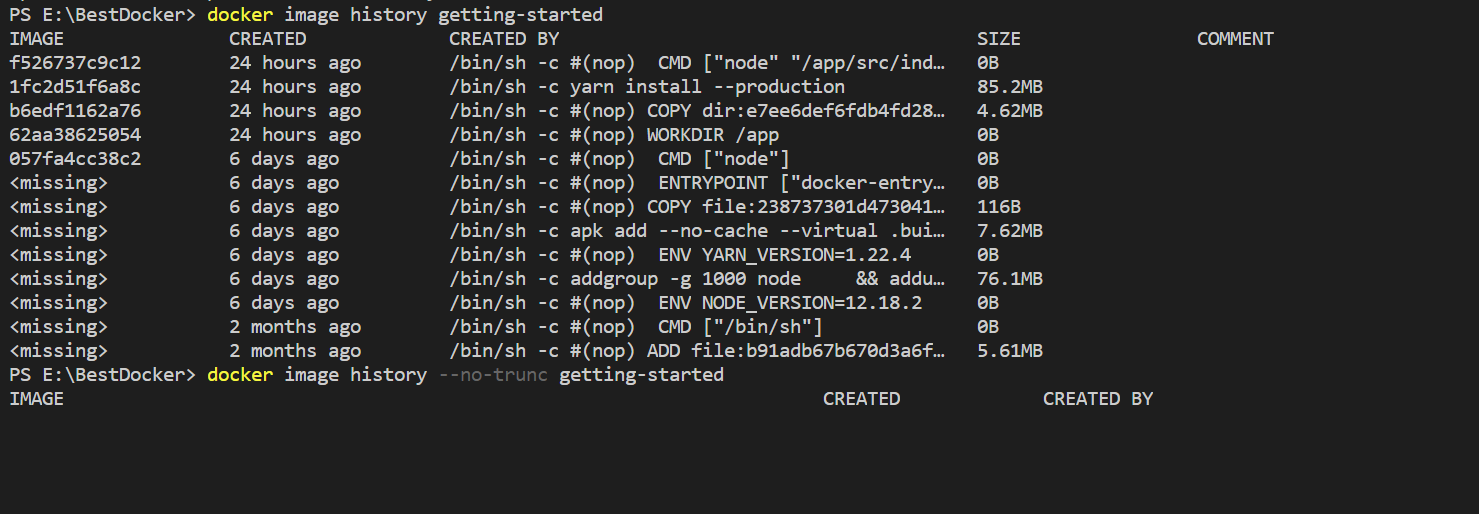


## Jika ingin Tearing it All Down

docker-compose down

Container berhenti dan network akan dihapus

Untuk lihat history Images



**Untuk download images: docker pull …:versi**

**Docker container ls utk lihat daftar container sedang running**

**Semua:docker container ls –all**

**Jalankan container: docker container start namacontainer**

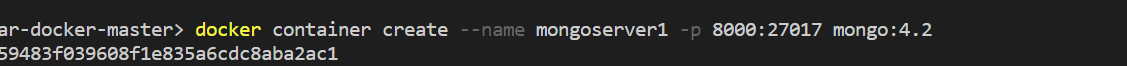
**Container dengan port :**

**Docker container create –name … -p 8080:29107 nama\_images**

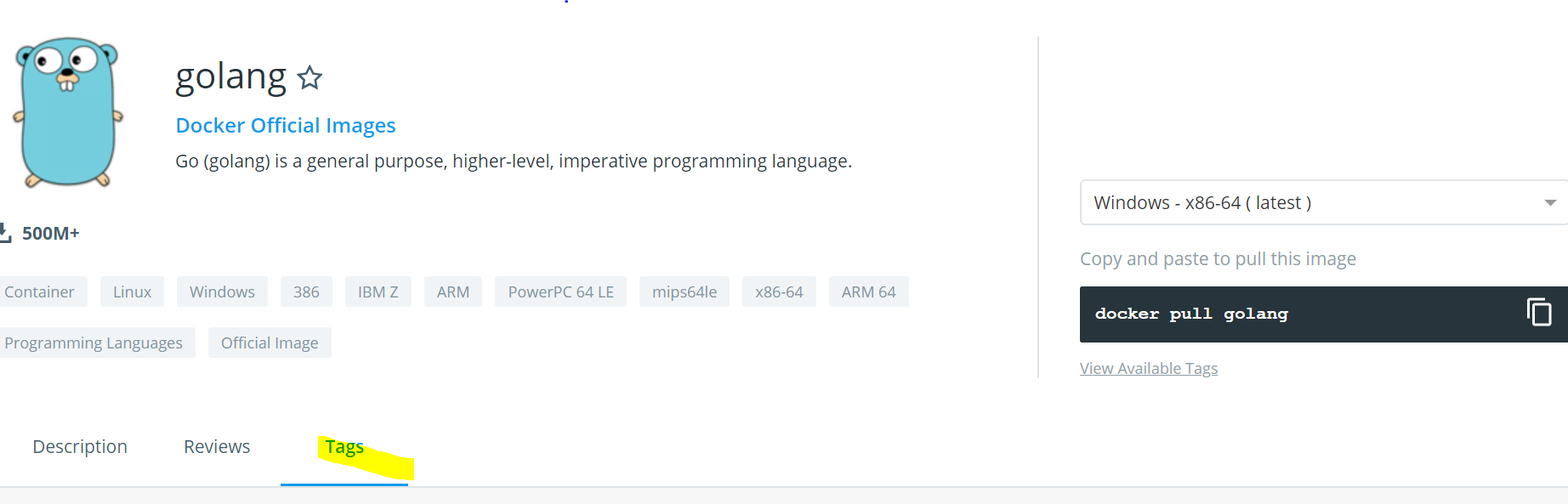
**Hapus images:**

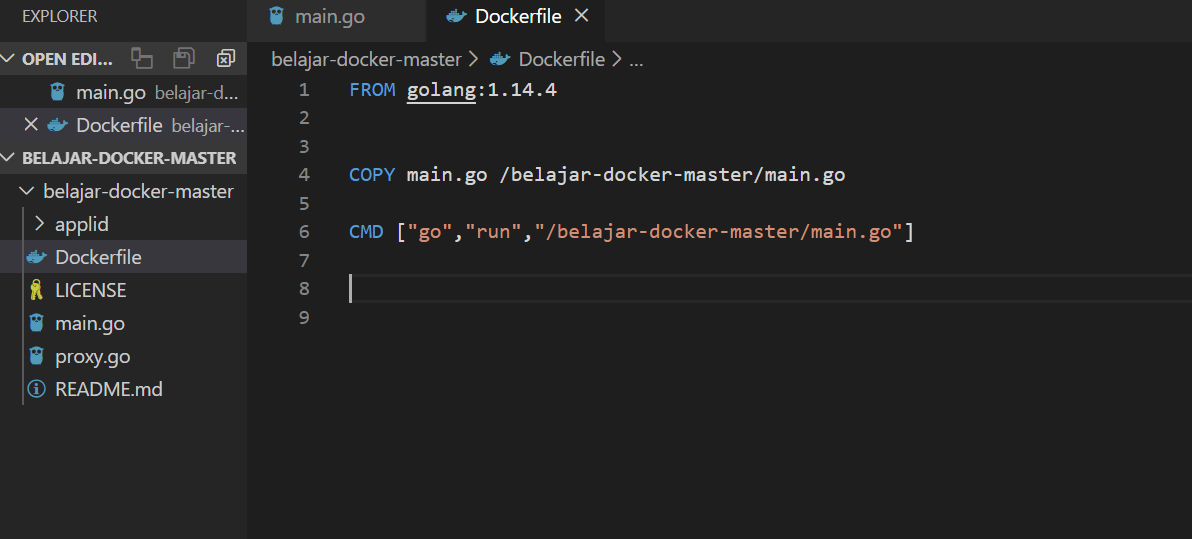
**Docker image rm nama\_images**

**\*Kalau hapus image,containernya jg harus dihapus**

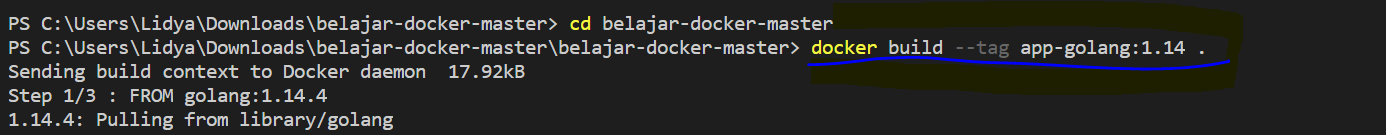


**CARA DEPLOY GOLANG DI DOCKER HUB**

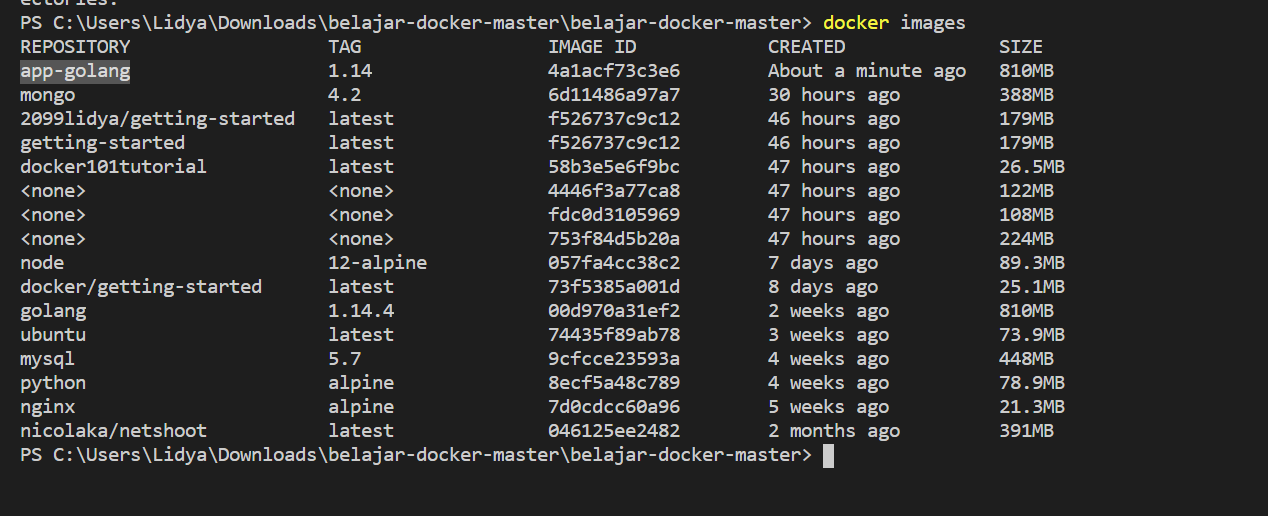
**CARI TAG YANG DIINGINKAN MISAL DI DOCKERFILE TULIS VERSI**



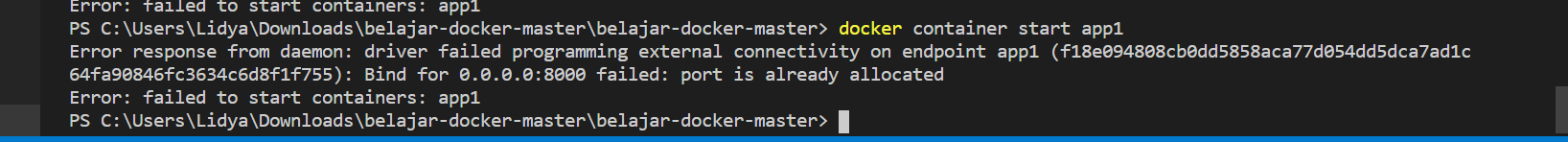
**LALU DI BUILD DENGAN NAMETAG GOLANG**

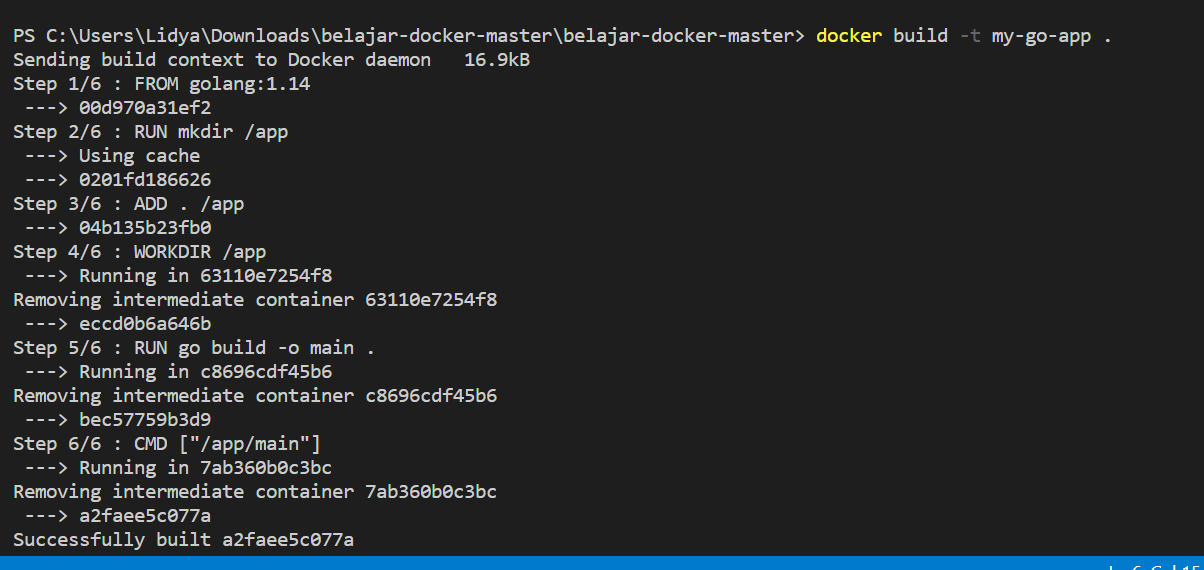


**Golang sudah ada**



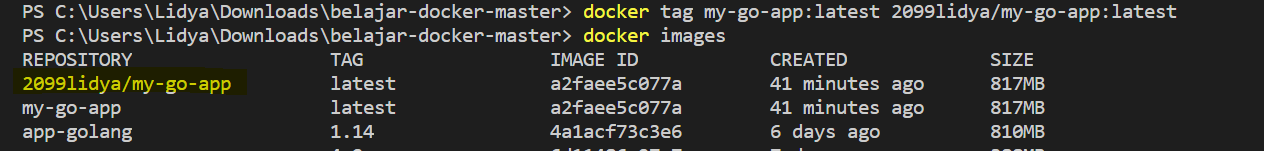
**Ini masih salah pas mau jalanin golang.**





Buka localhost:8080 berhasil

Buat images bisa digunakan taruh di dockerhub: (local ke registry)



Kemudian: - docker login

* Lalu push: docker push 2099lidya/my-go-app:latest –example
* Jika mau ambil imagesnya tinggal docker pull