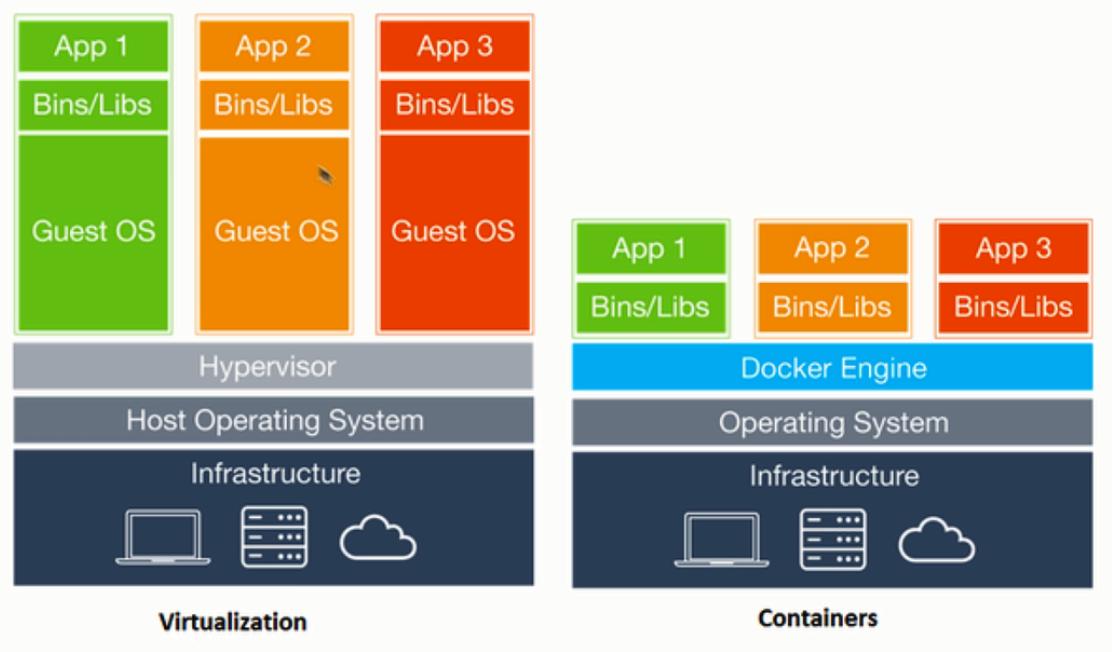
1. Bruno's Request

3. Docker Containers



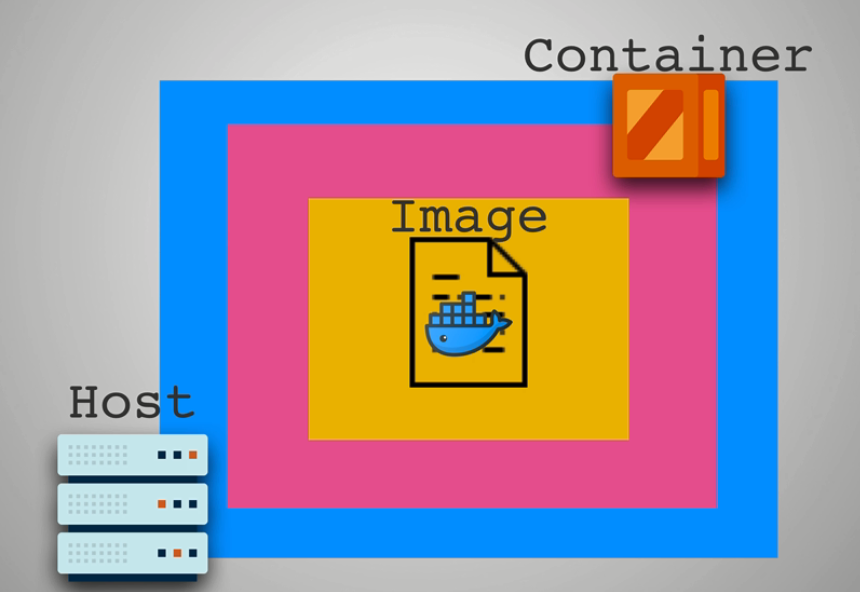
Trước khi có docker -> dùng virtualize machine ( VM ware, virtual box) : trên một computer có các ENV on top -> Một hệ thống riêng biệt chạy hệ điều hành riêng và có các thư viện trên đó -> để chạy application sẽ mất time ( few minutes)

Docker: gói gọn tất cả phần mềm cần thiết trên một file hệ thống <-> virtual machine : code, system tools, system libraries,… -> container

Container: user the host operating system ( an **operating system** used in a virtualized server ) -> less time ( few seconds )

* Docker easy access to run single app in it

Container:



We first have a host ( machine or computer will host our container = our PC)

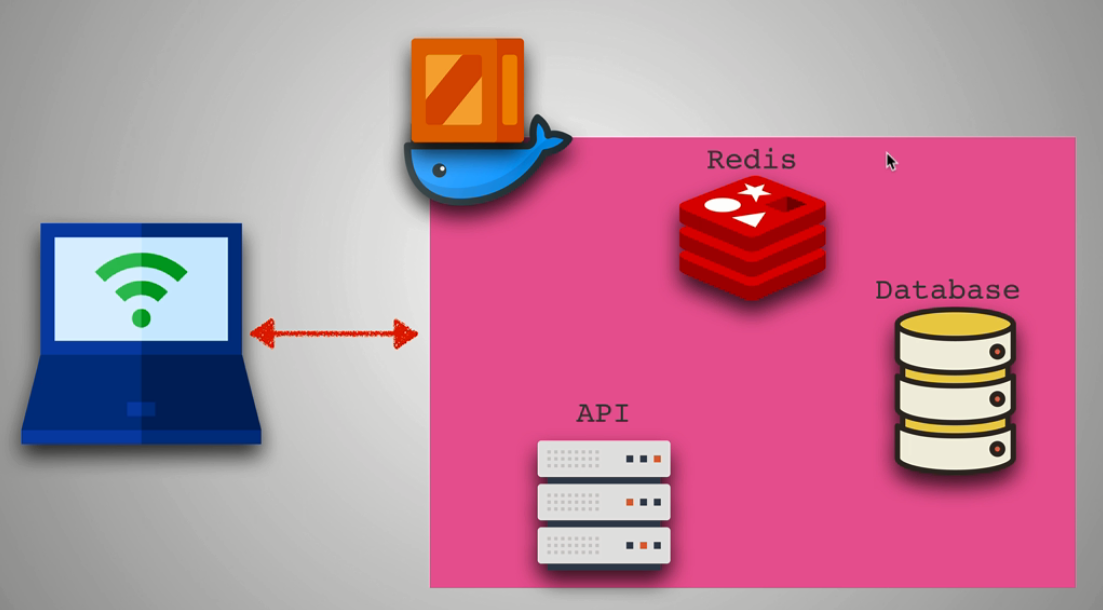
On it we create container with docker

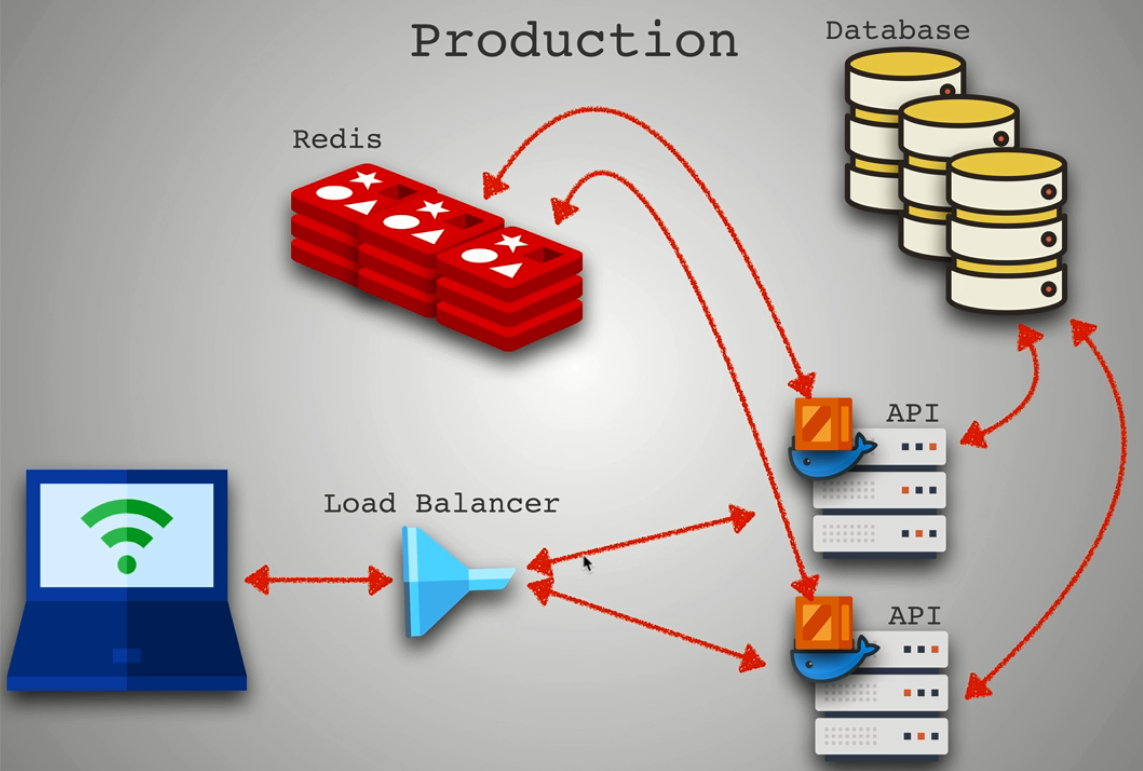
Inside container we have images: standalone package need for app like node server, postgres, redis

Env inside container is completely isolated from that virtual machine

* We can have all and just libraries and dependencies that we want for our app to work

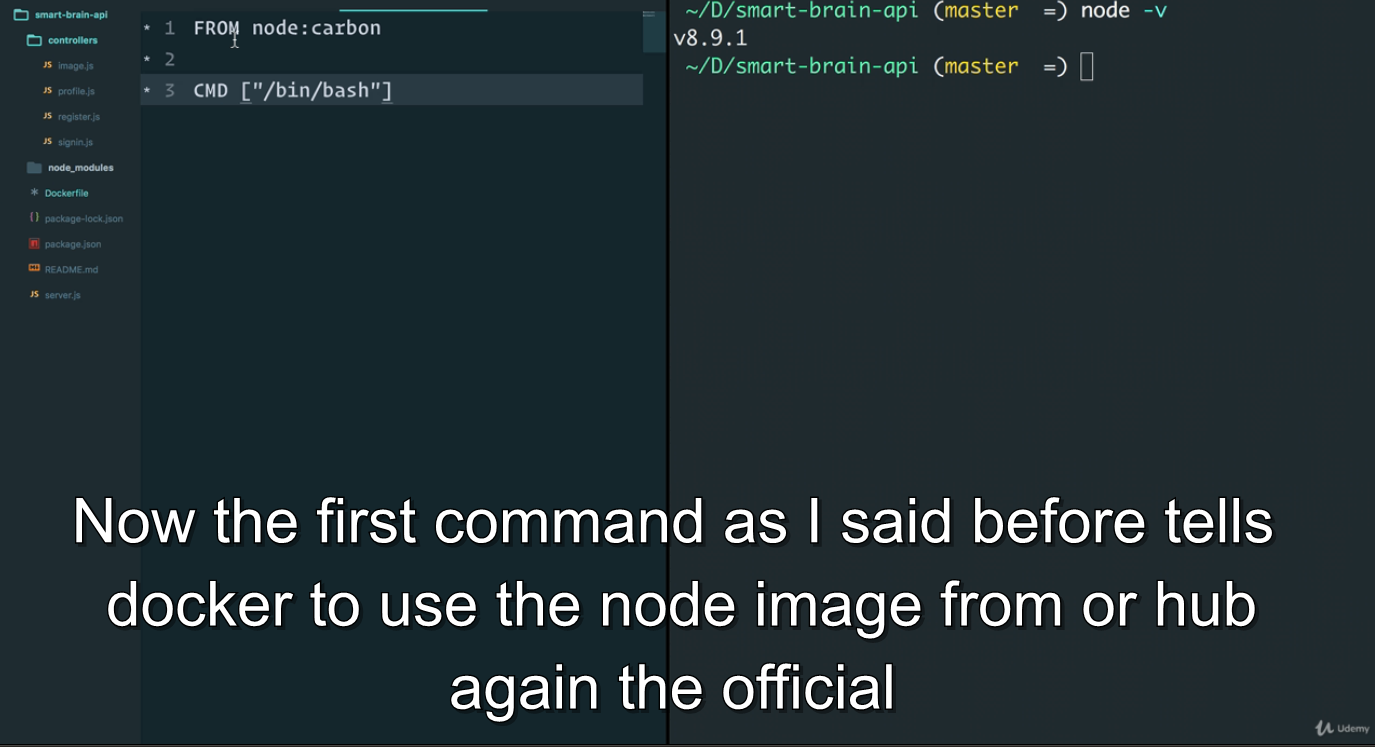
DockerHub: ( giống kiểu NPM ) search and download image

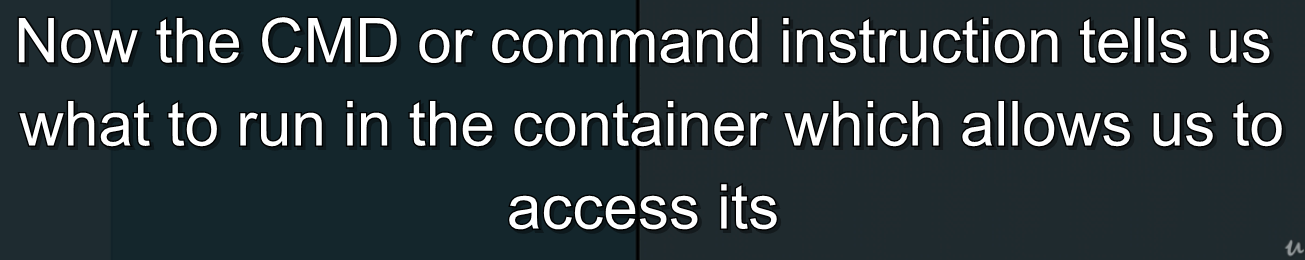




8. Docker Commands

<https://kapeli.com/cheat_sheets/Dockerfile.docset/Contents/Resources/Documents/index>

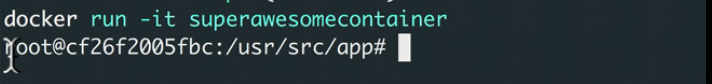




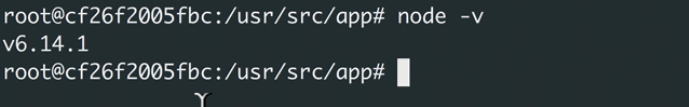
Cái lệnh sẽ chạy trong cli của container ngay khi start container



Chạy lần sau nhanh hơn vì đã get về rồi



We inside teminal of container

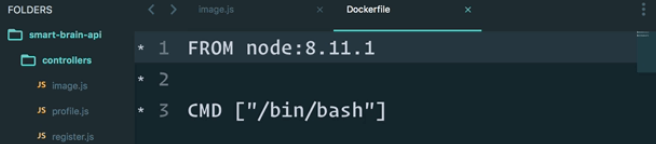


Ví dụ gõ trong teminal node –v -> thấy được version của node



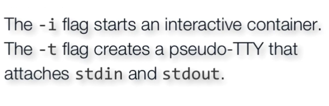
Khi thay đổi file docker thì run lại docker build

Ví dụ check version xong thấy sai version



Đổi tag lại rồi run lại lệnh build

-it : xem file sumary cũ nhé





-d để run container in background



Xem các container đang run ( không thấy mấy thằng stop)

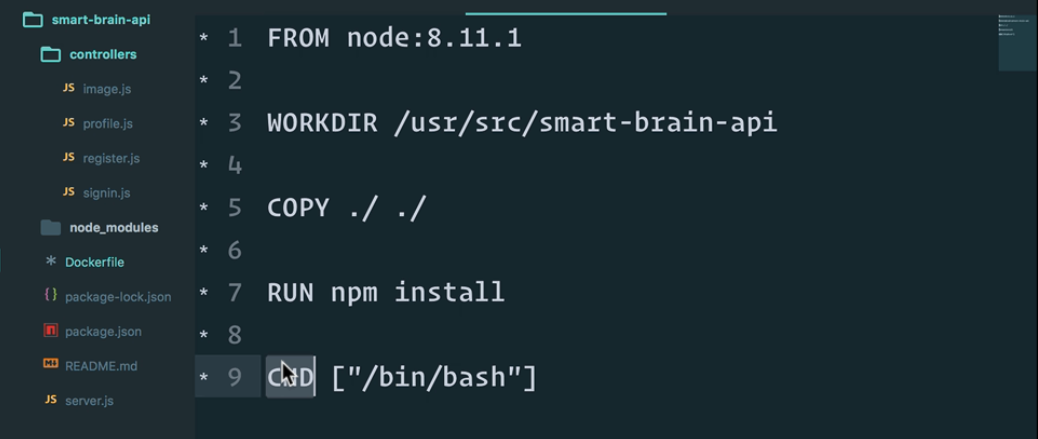


Nhảy vào container



Stop container

9. Dockerfile 2



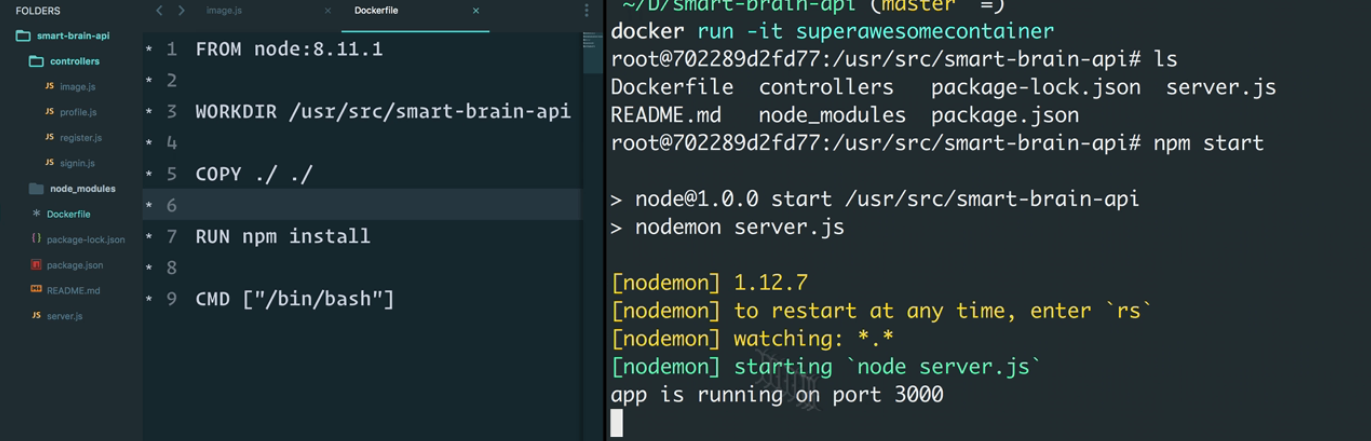
Workdir dễ nè

Copy dễ nè

Run sẽ chạy comman trong cli của container

Mỗi lệnh run sẽ chạy một layer mới, layer sau sẽ dùng layer trước để chạy tiếp

Cmd là lệnh run sau khi build thôi, như gõ một command trên cli bình thường, ko build layer image



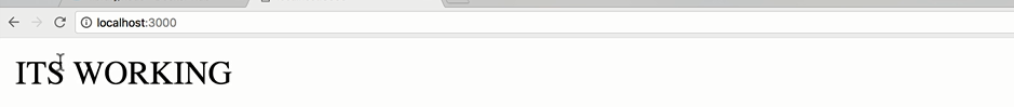
Có ngay quả server chạy trong này

( node đó để chạy npm nhé)

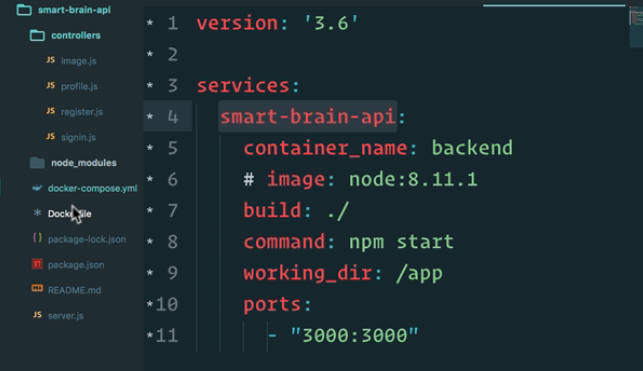


Thêm port để map post của container -> post của host (machine)

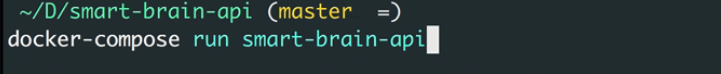
Khi đó, sau khi chạy npm start trong container -> vào đk web đó



12. Docker Compose

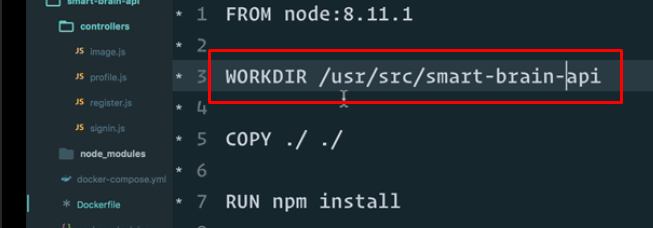


Build sẽ chạy lại từ dockerfile luôn



Run container

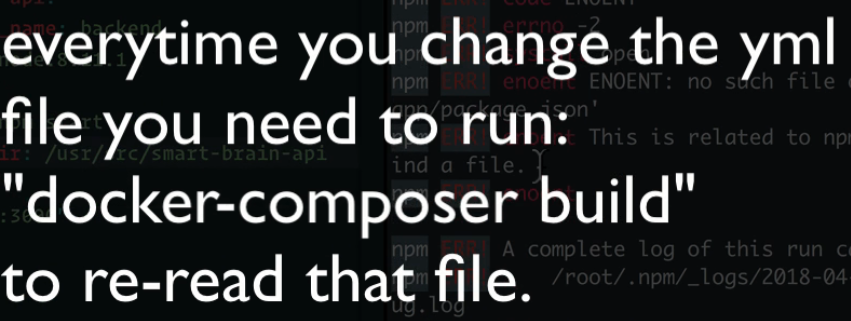




Work dir trên docker file

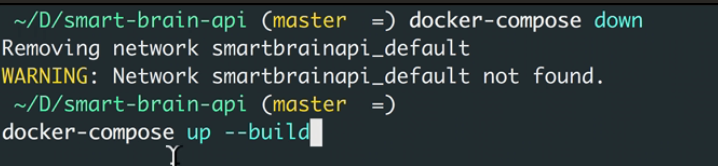


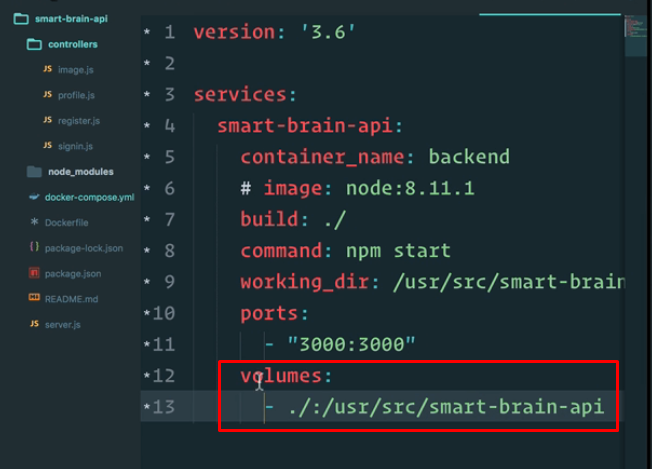
* Đổi thành giống dockerfile

 mỗi lần thay đổi trong file cần phải build lại



Chạy lại -> ok





Thêm volumes: map với dir của container

Khi file change ko cần build lại

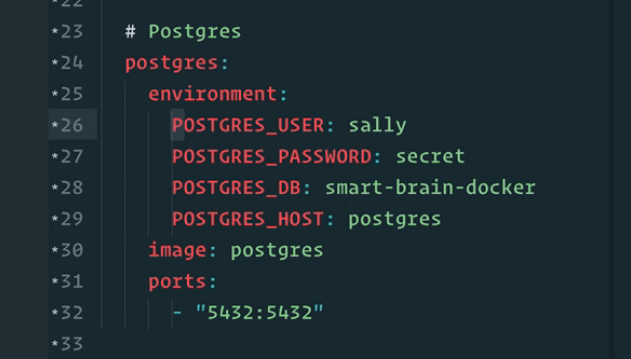
Change trong compose thì cần up lại

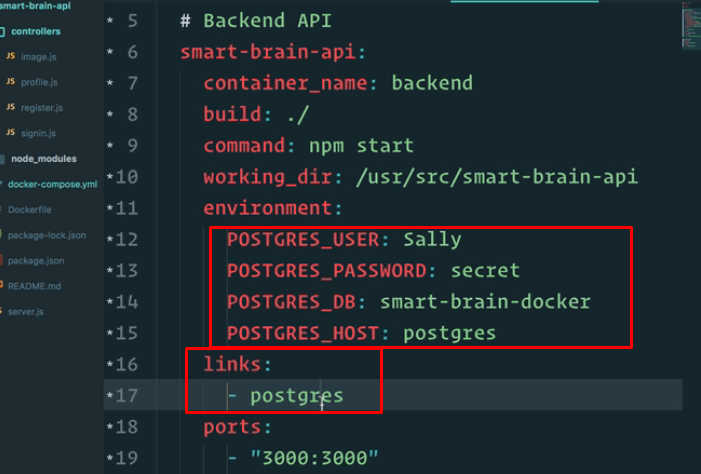
Change trong dockerfile thì mới cần build lại



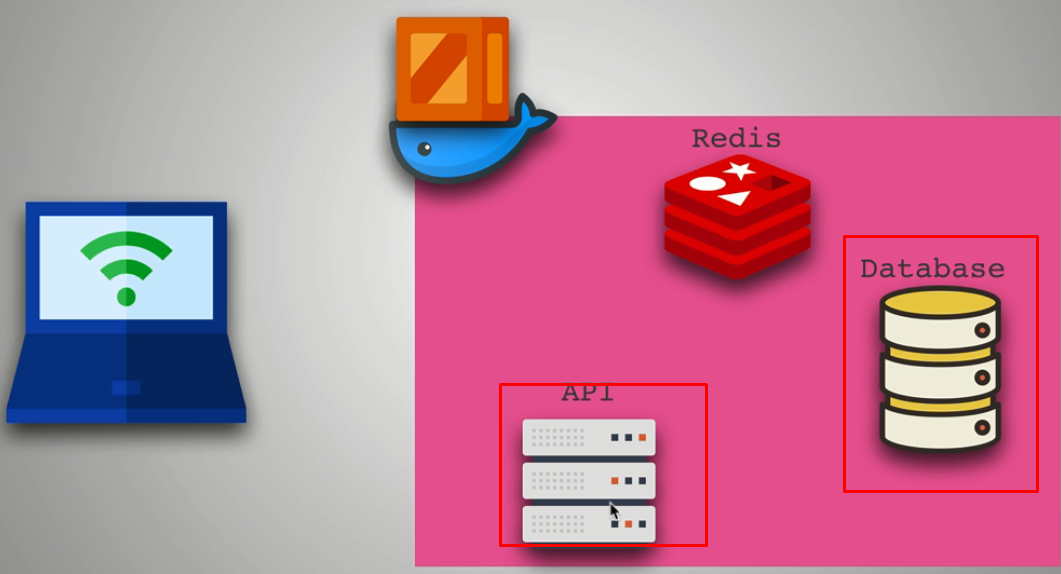
Execute container và vào cli của nó

Cái container đó phải đang run (up ) thì mới exec được

  
Tạo các giá trị env cho postgres, cái này là khởi tạo

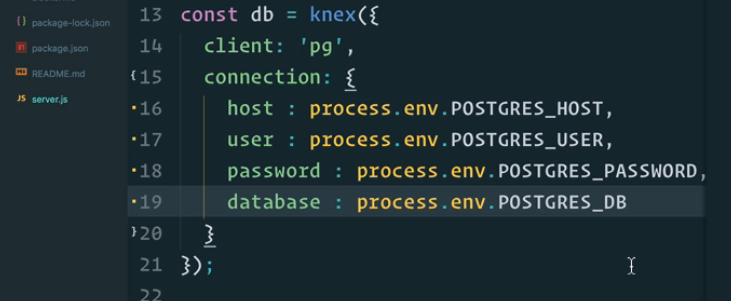


Đặt các biến môi trường và link app to postgres

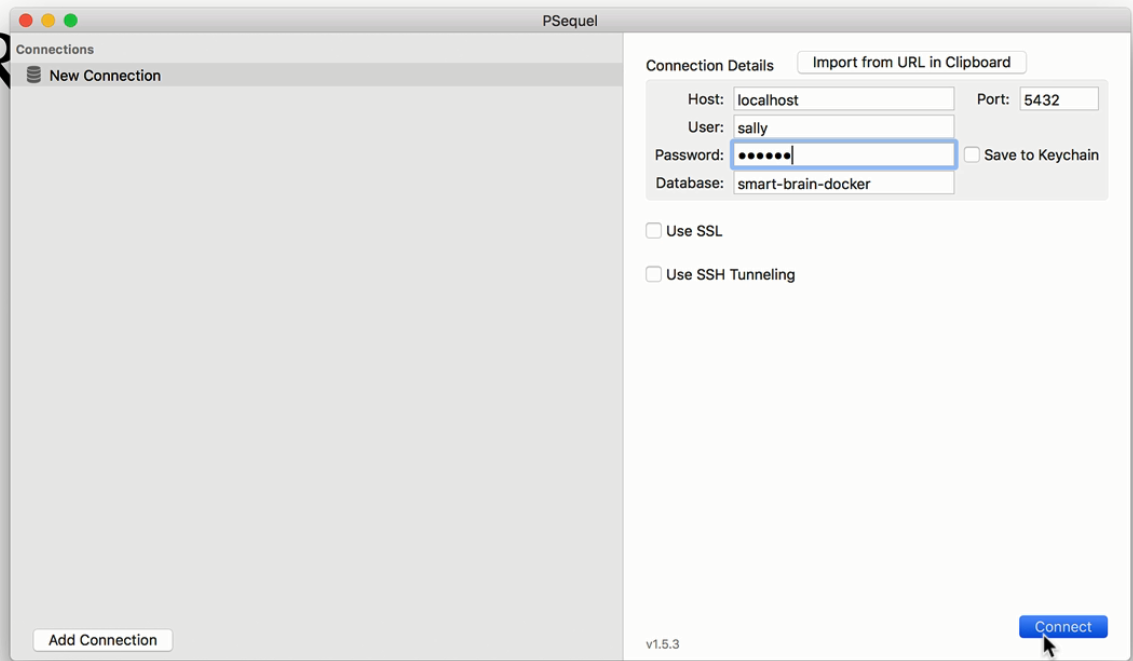


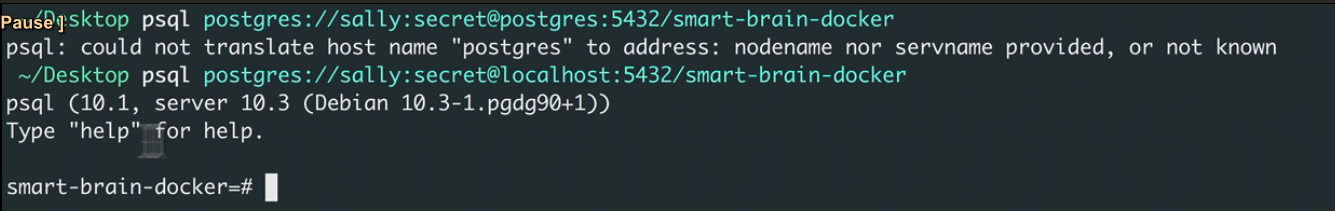
2 cái này tồn tại độc lập không hề biết sự tồn tại của nhau.

Link api vs db để api có thể access db

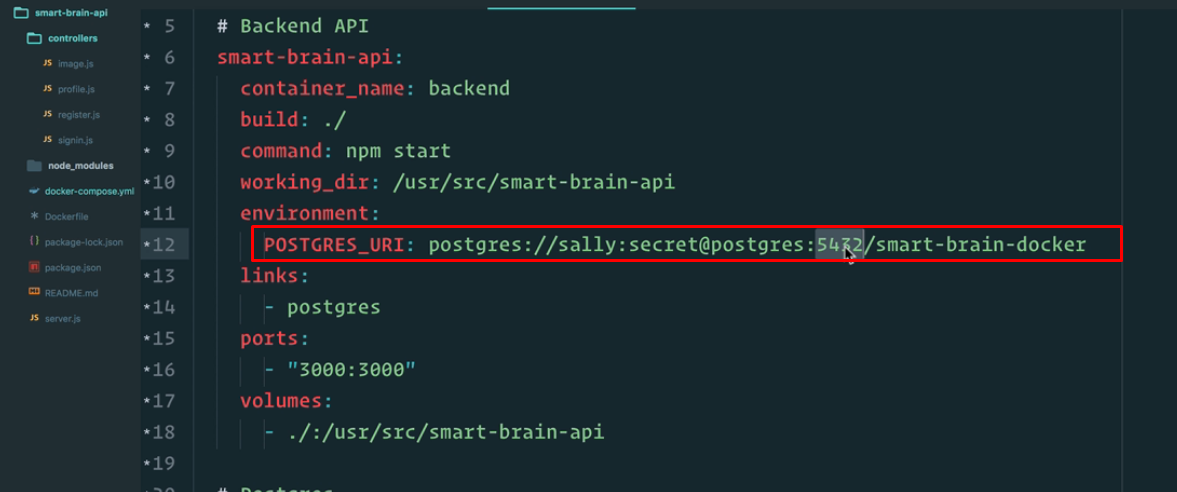


Access env variable để config db connection

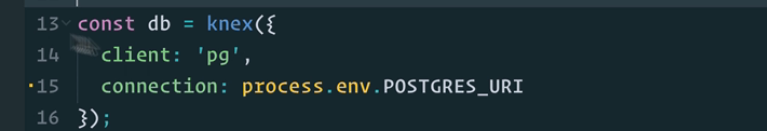


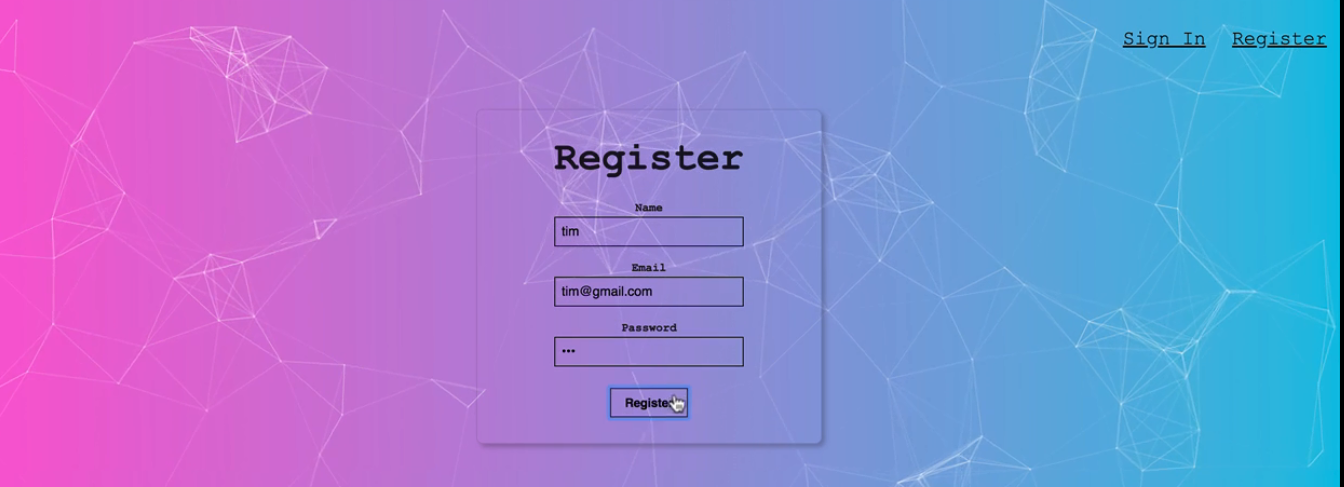


Connect to postgres in command line ( hoặc có thể dùng tool như trên)

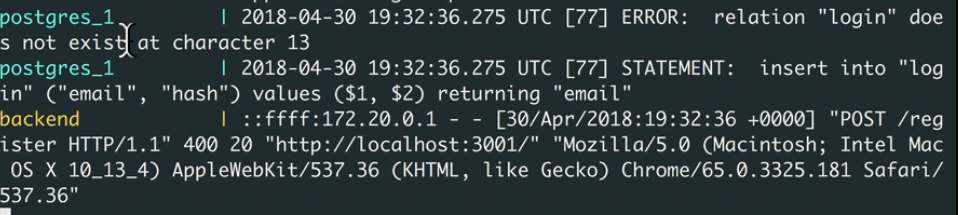


Đổi thành ntn cho dễ xài

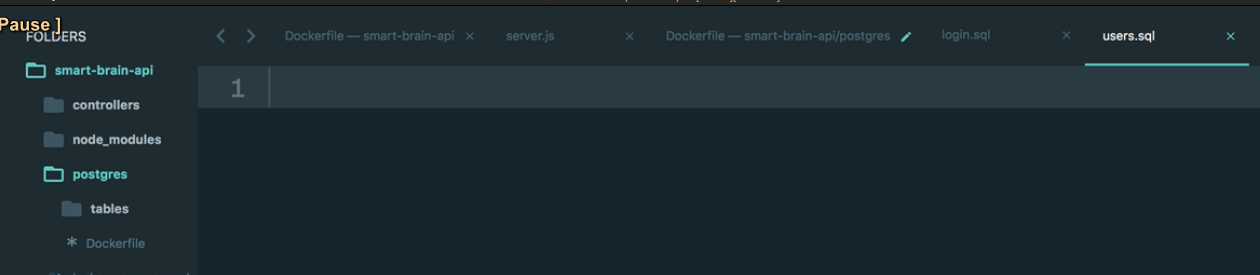




Register thì gặp lỗi



Tại chưa tạo table

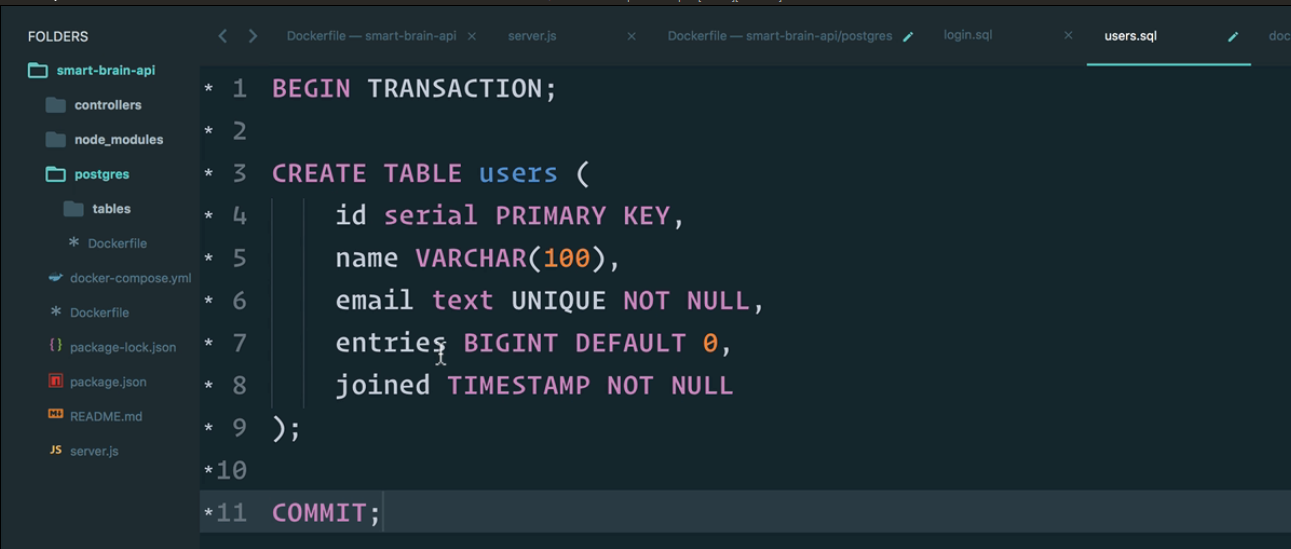


Tạo thư mục postgres, trong đó có thư mục table

Thư mục table chứa các table

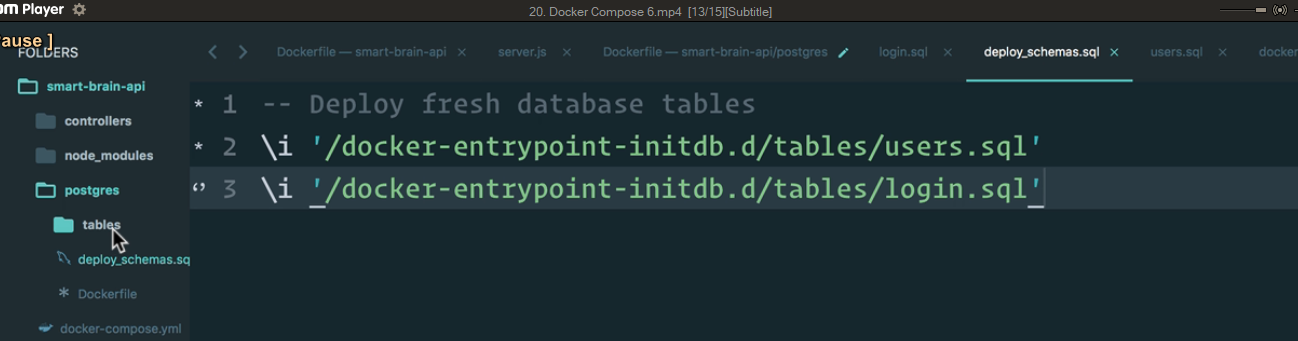
Login.sql

Users.sql





Thư mục deploy schemas cùng cấp với table



File này chạy từng file sql để tạo từng table

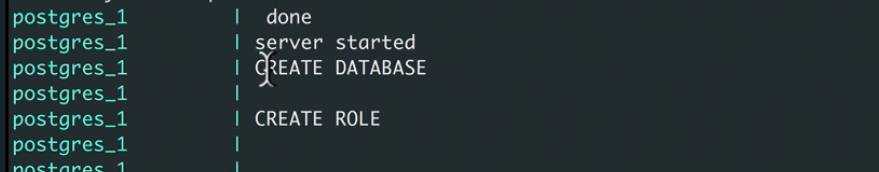


Add table vào

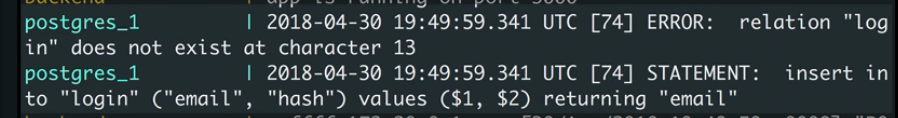
Add file deploy vào

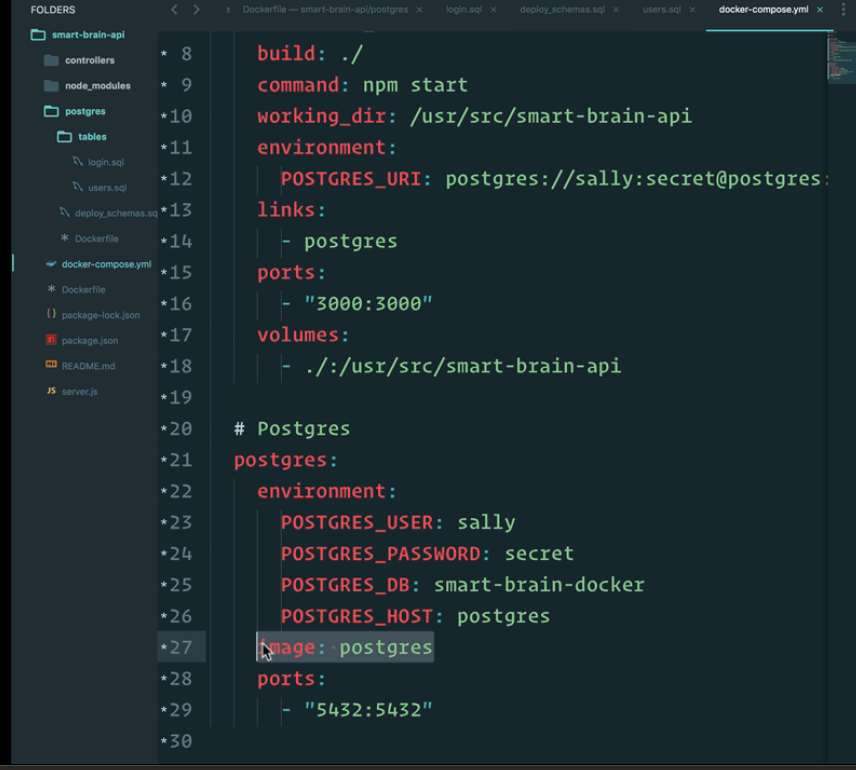


Down xong up lại



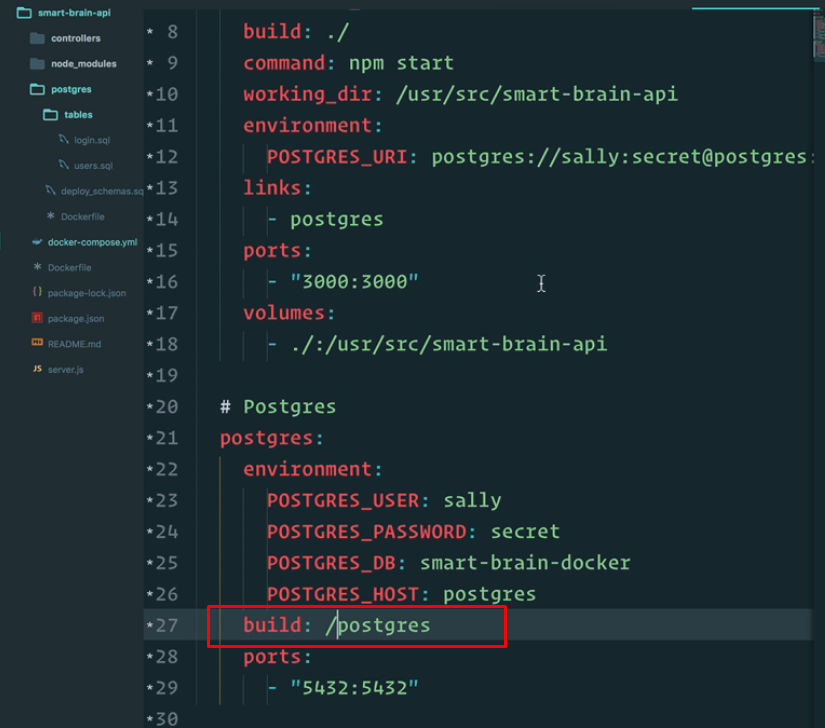
Ahihi





Lúc đâu build từ image, nhưng sau này đã có docker file trong thư mục postgres rồi

* Build



* Boom saca thành công

22