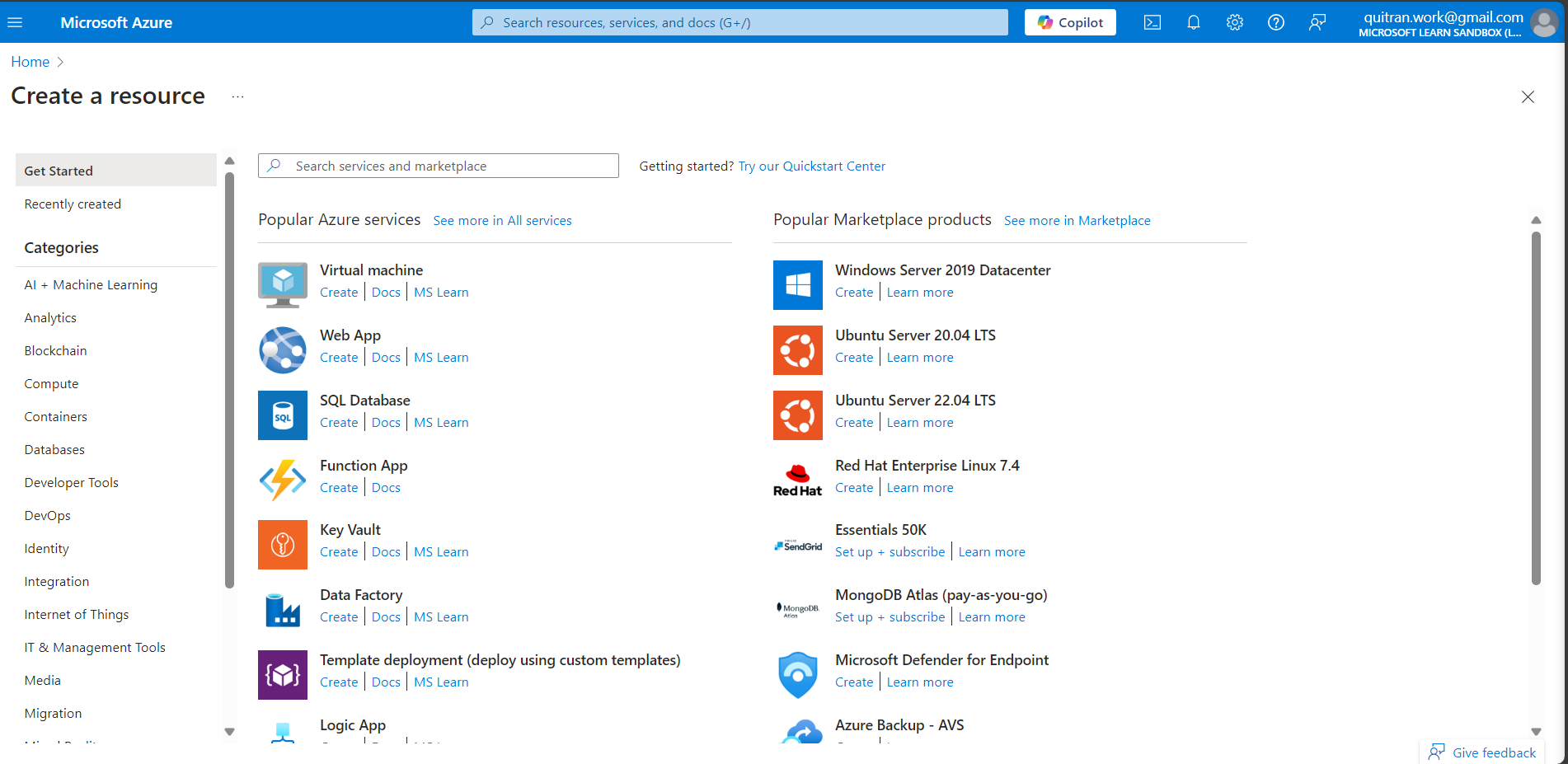
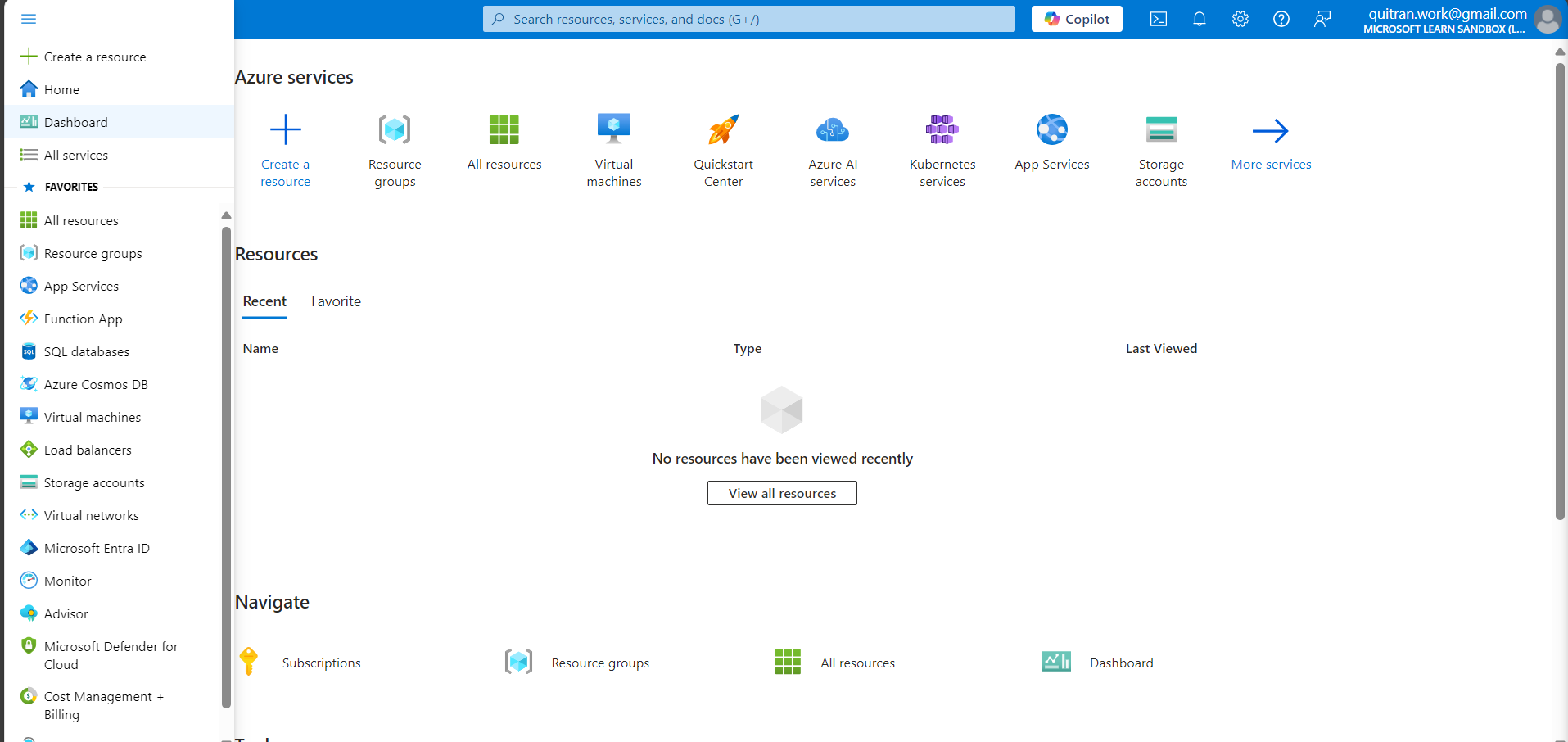
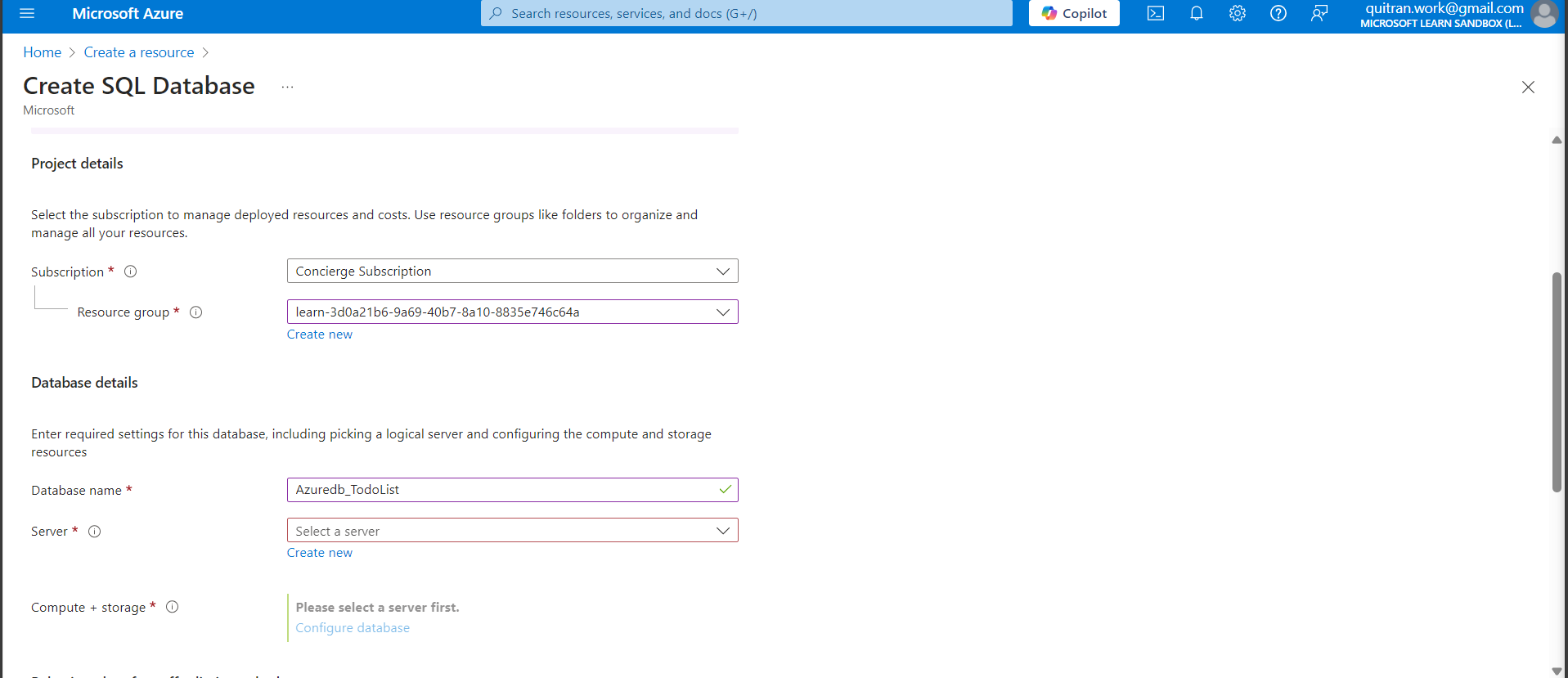
**Hướng dẫn từng bước tạo và kết nối SQL Database trên cloud cho demo.**

- Ta sẽ tạo **Azure SQL Database** bằng tài khoản **Learn Sandbox** của **Azure** để demo do nó miễn phí (lưu ý là tài khoản Sandbox chỉ được kích hoạt 1 tiếng cho mỗi lần dùng và có 10 lần dùng cho mỗi ngày sau khi hết 1 tiếng thì dữ liệu sẽ mất hoàn toàn, sẽ có 1 vài dịch vụ không thể sử dụng do chính sách của bên Azure) và mục đích ở đây cũng chỉ là demo nên ta sẽ sử dụng cho tiện lợi và tối ưu mức chi phí khi tạo database nhất có thể**.**

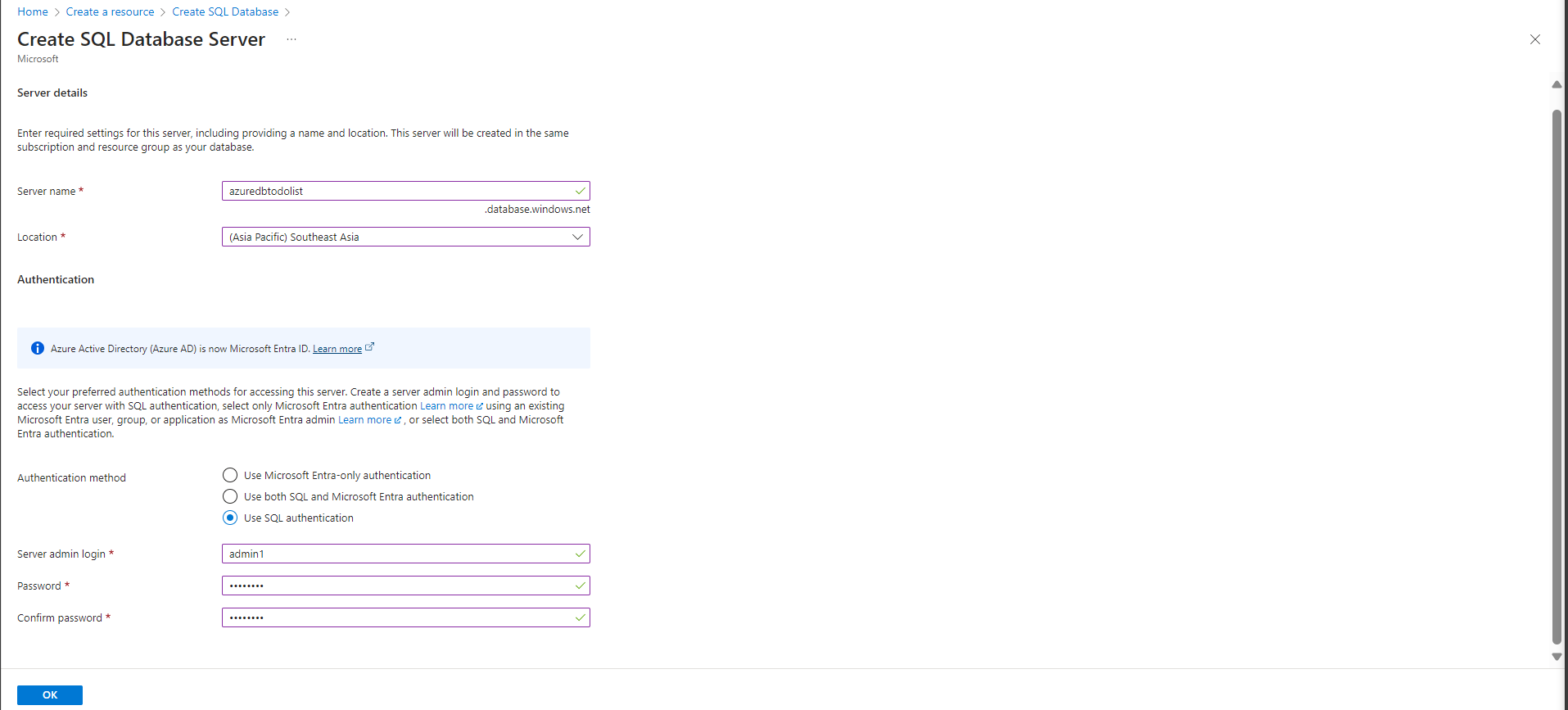
- Sau khi đã đăng nhập vào Azure Portal ở chế độ Sandbox thì ta bấm **Create a resource** hoặc ở menu bên trái và tìm **Create a resource** và chọn **SQL Database** để tạo database cho demo.

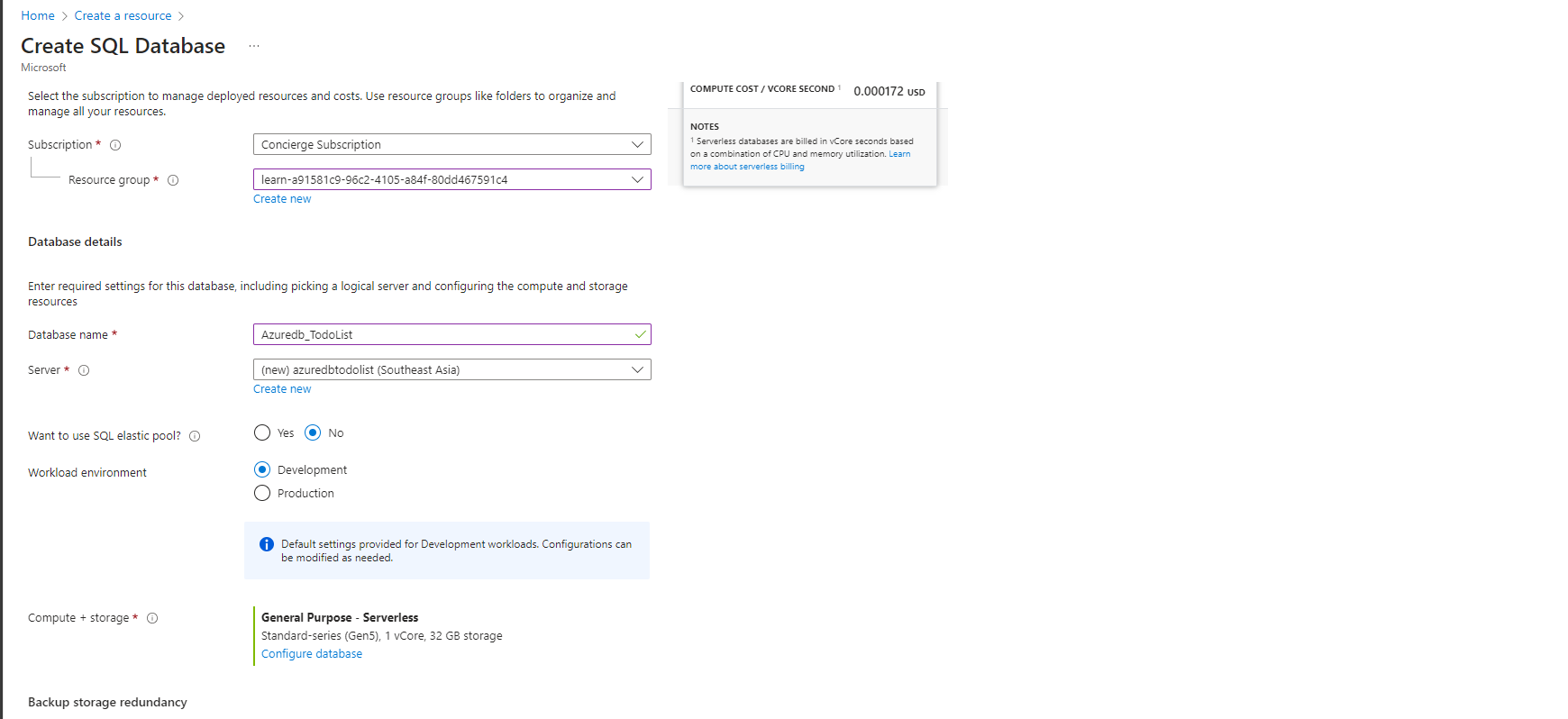


- Điền thông tin đầy đủ và chọn **Resource group**.

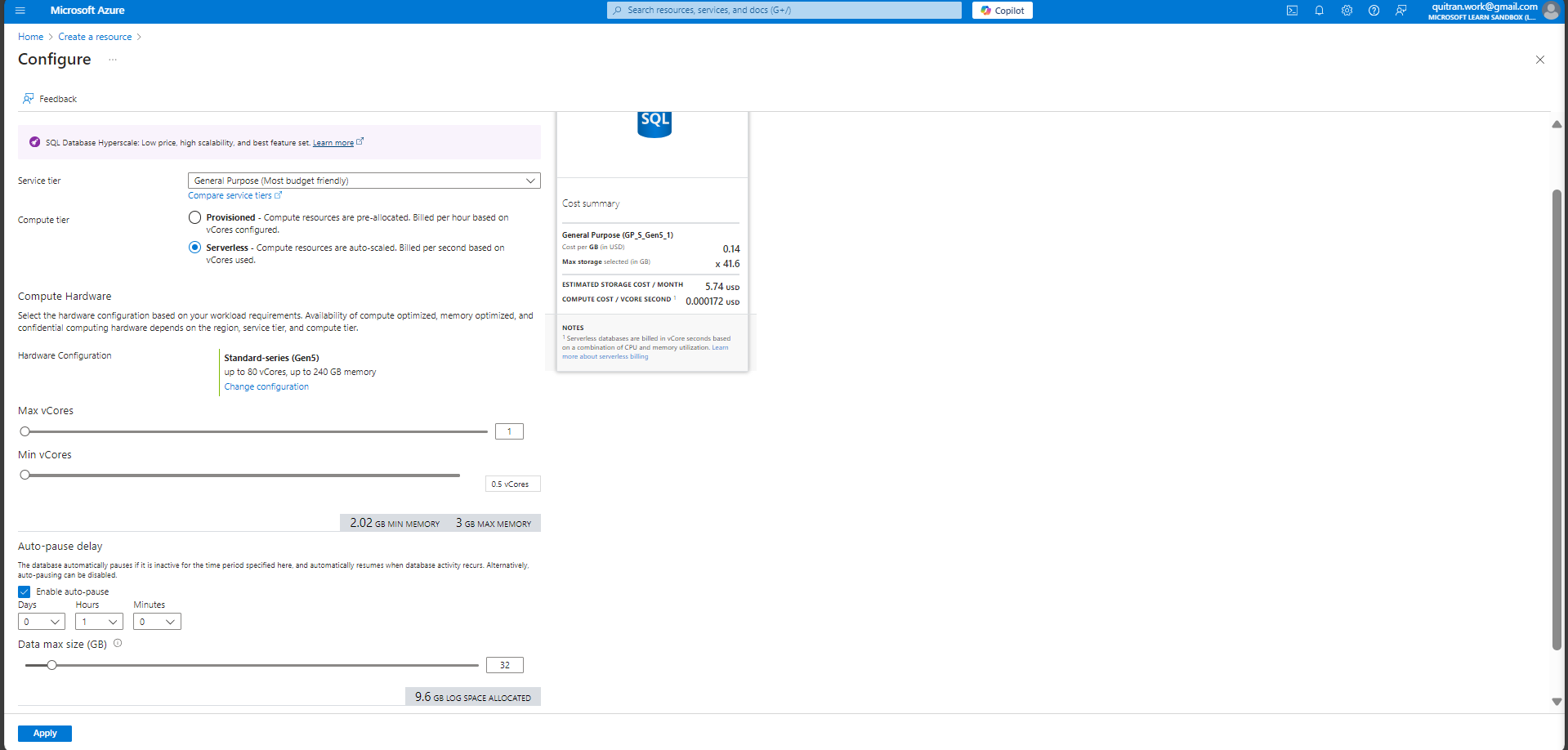


- Ở phần server do ban đầu chưa có server nào nên ta sẽ chọn **Create new** để tạo mới.Mục **Authentication method** ta chọn SQL Authentication và điền thông tin server login và password mục đích của server này là 1 **url** để liên kết đến database server, sau khi đã cấu hình xong thì nhớ lưu lại login và mật khẩu để tiện sử dụng sau này.Chọn **Ok** để hoàn tất.



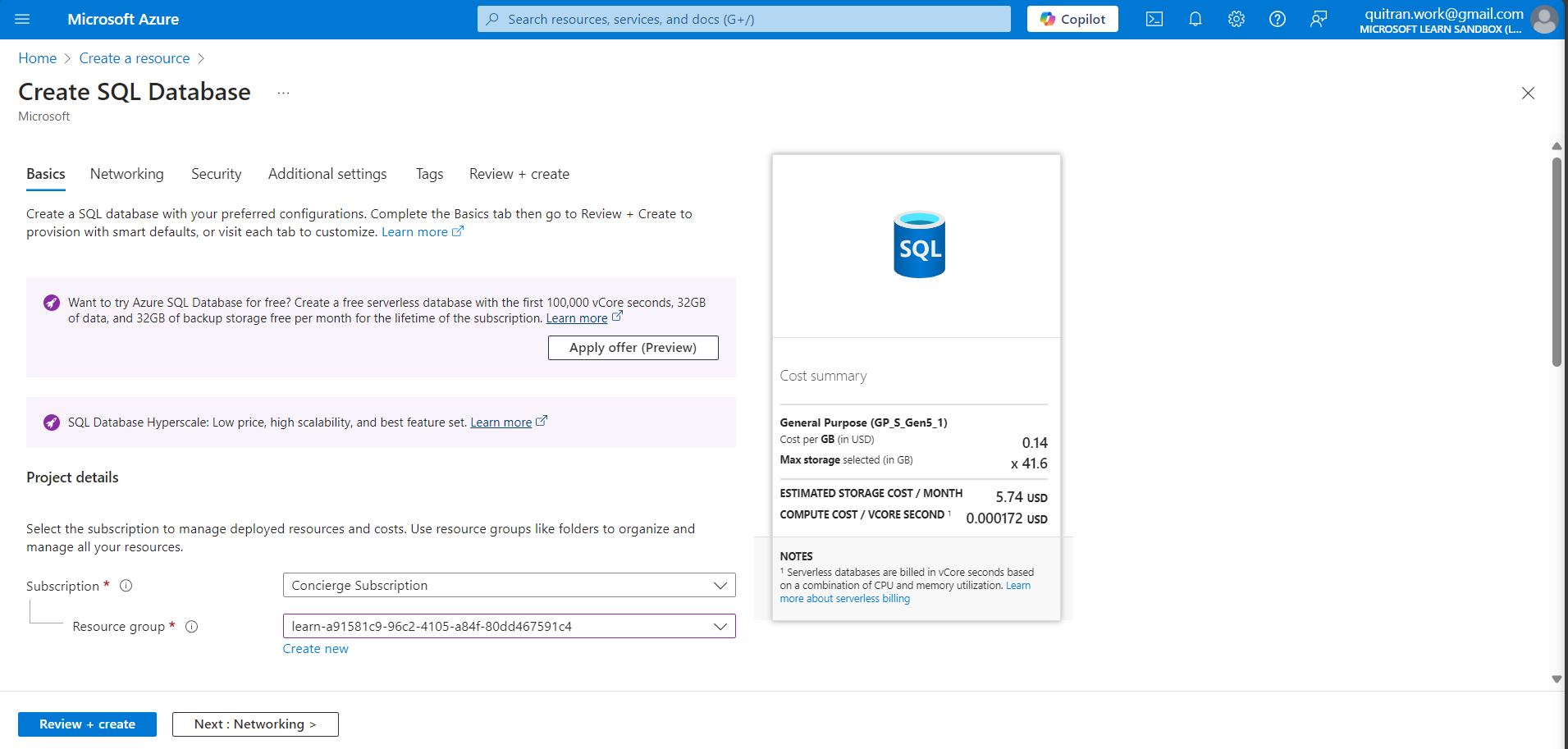


- Chọn Configure database để cấu hình cho database.

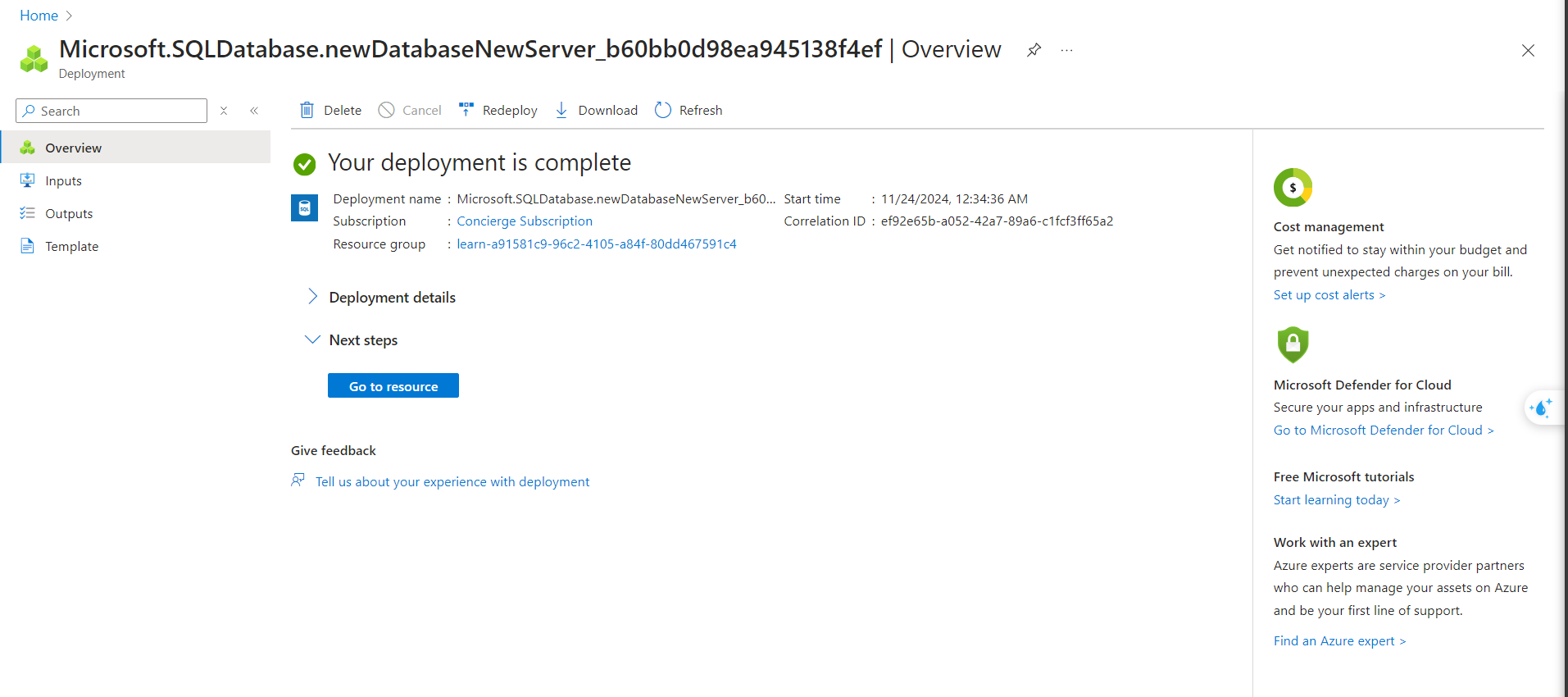


- Sau khi đã cấu hình theo ý muốn thì nhấn **Apply** để lưu.

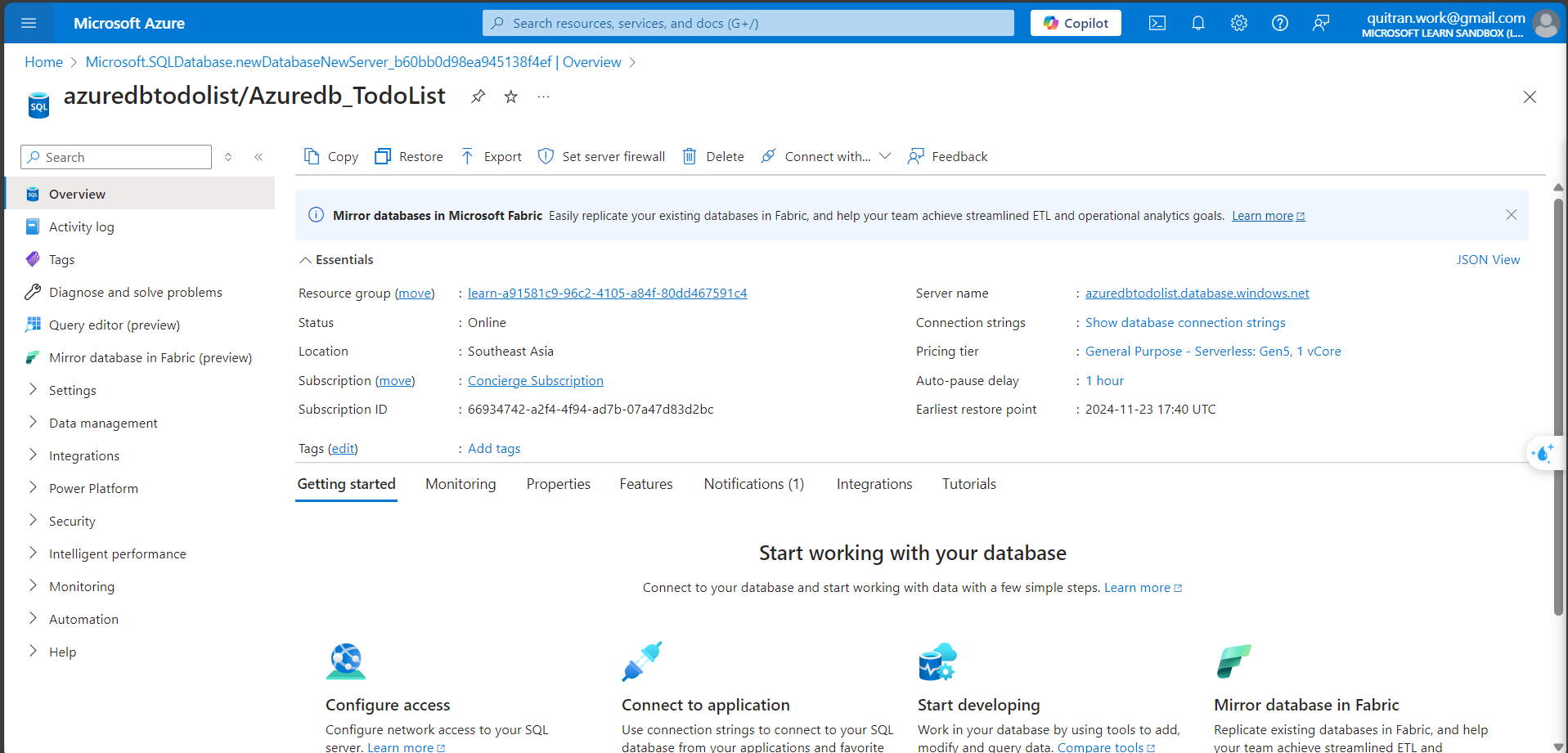
- Do hiện tại khi đang tạo bên Azure đang có 1 offer tạo 1 máy chủ miễn phí với 100.000 giây vCore đầu tiên, 32GB dữ liệu và 32GB dung lượng lưu trữ sao lưu miễn phí mỗi tháng trong suốt thời gian đăng ký(lưu ý khi hết 100.00 giây vCore thì database sẽ bị dừng và phải trả phí để tiếp tục bạn có thể cân nhắc điều này). Có thể chọn apply để sử dụng miễn phí cấu hình trên.



- Nhấn tiếp **Review + create** và nhấn **Create** để hoàn tất tạo database sau đó chỉ cần chờ database tạo thành công là xong. Khi hoàn tất nhấn **Go to resource** để xem database vừa tạo ra.

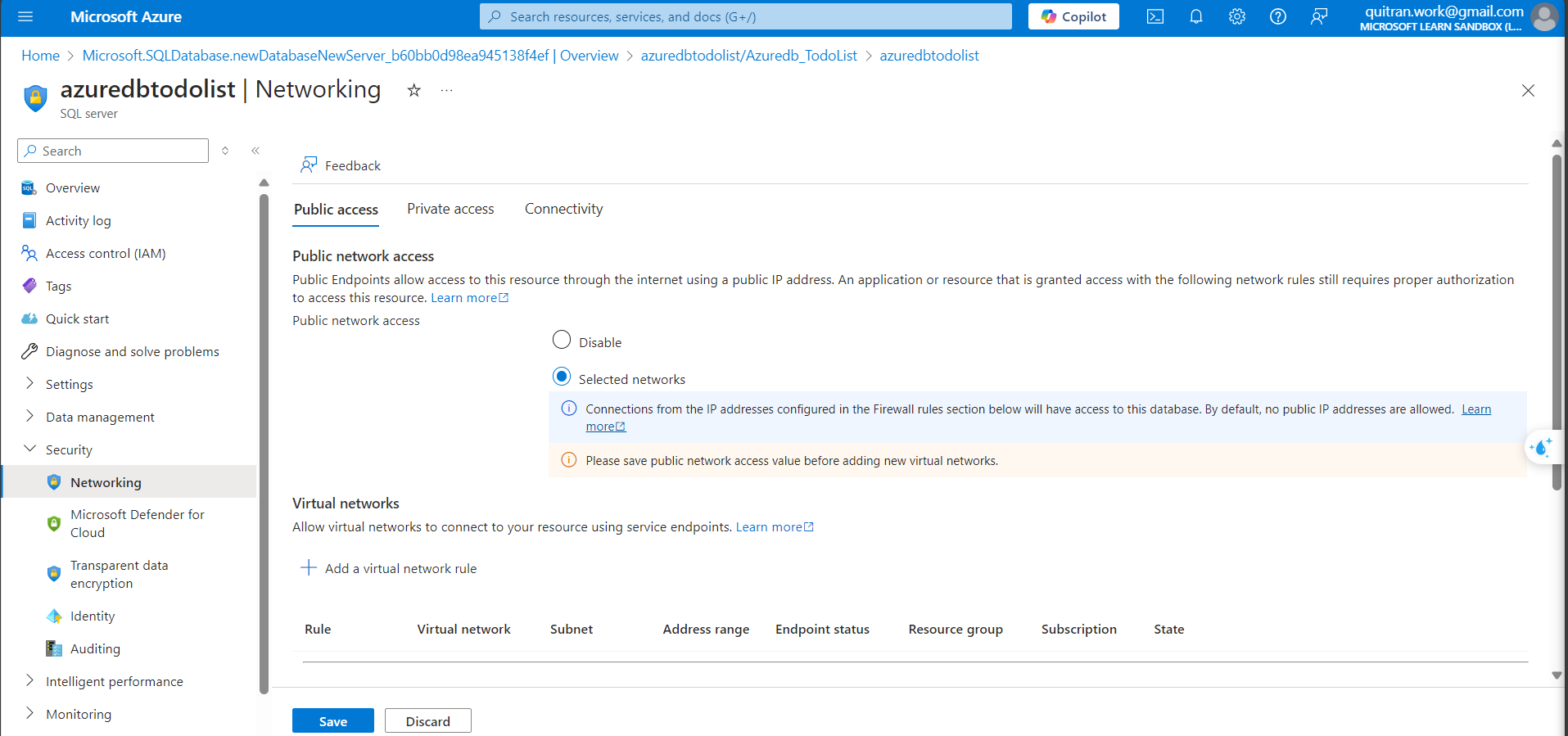


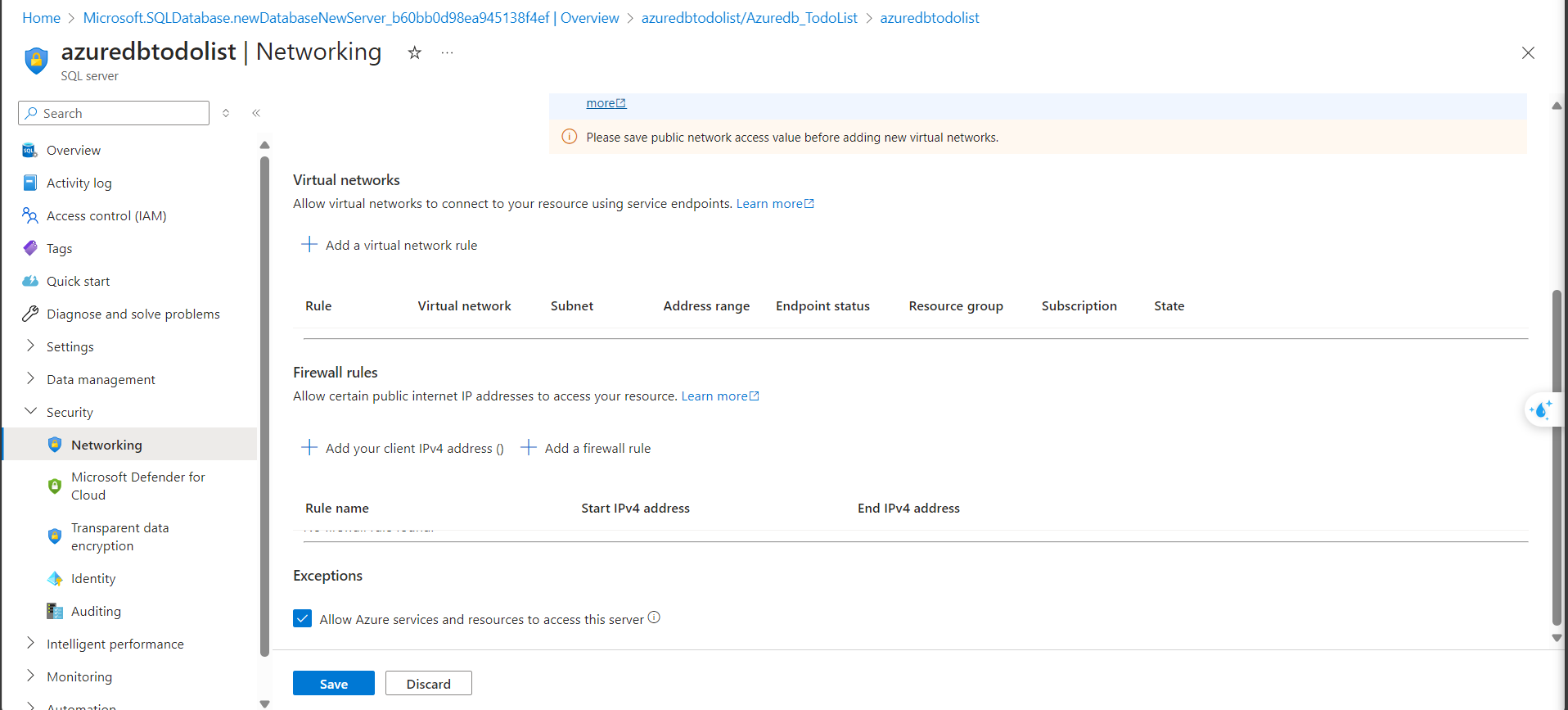
- Để có thể cho phép kết nối với database trên cloud vừa tạo thì chúng ta phải cấu hình Firewall. Ngay tại giao diện **Overview** ta có thể thấy mục **Set server firewall** ta nhấp vào đó.

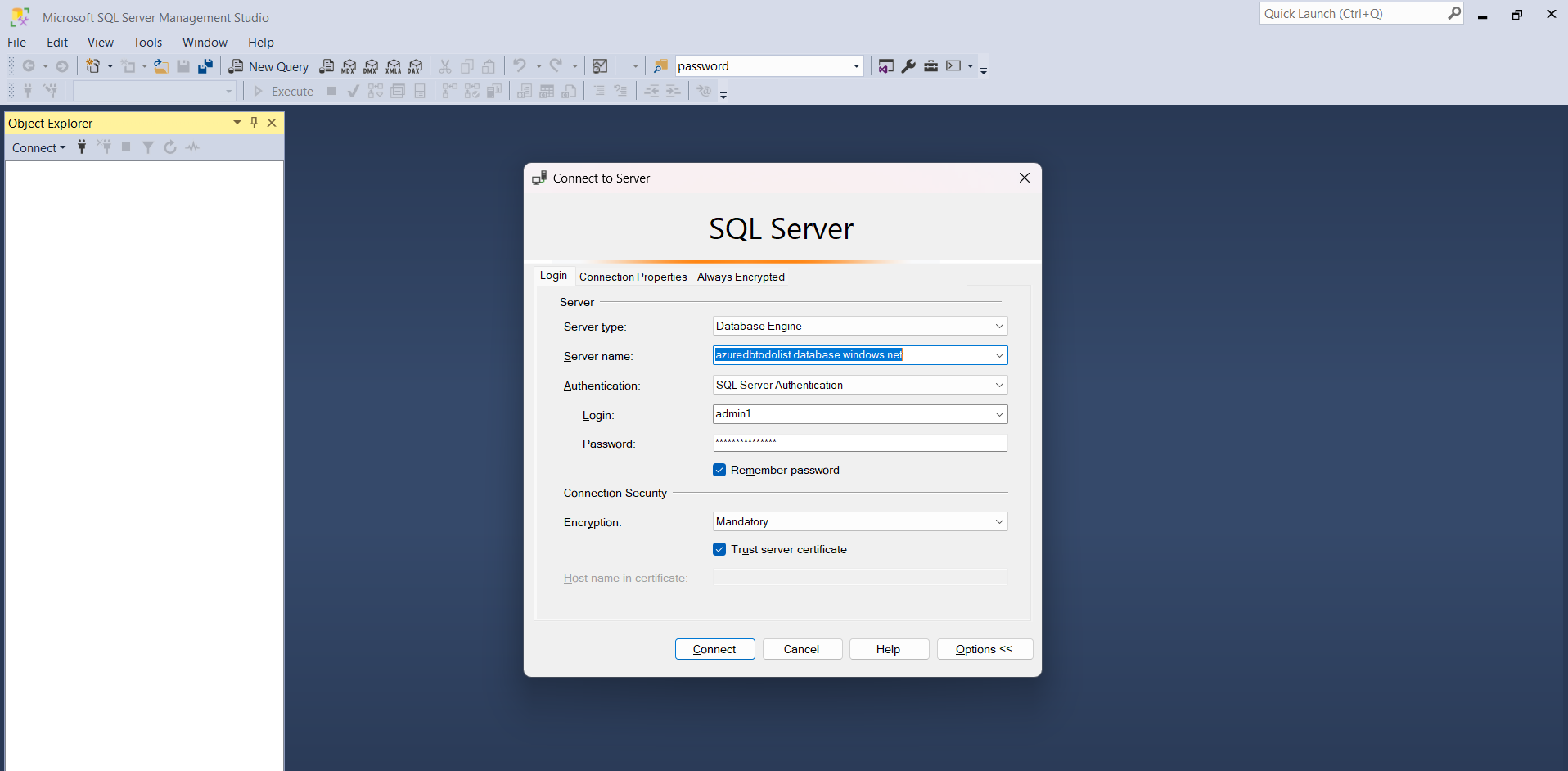


- Vì lý do bảo mật, chỉ các máy ảo/app service nằm chung resource group/virtual network với database server mới có thể kết nối tới server đó. Do vậy, phải thêm IP của mình vào whitelist để không bị chặn.

- Ở mục **Public access** ta chọn **Selected networks** lăn chuột xuống ta có thể thấy mục cho phép dịch vụ Azure và resource có thể truy cập vào server này ta **Allow** sau đó bấm **Save**.

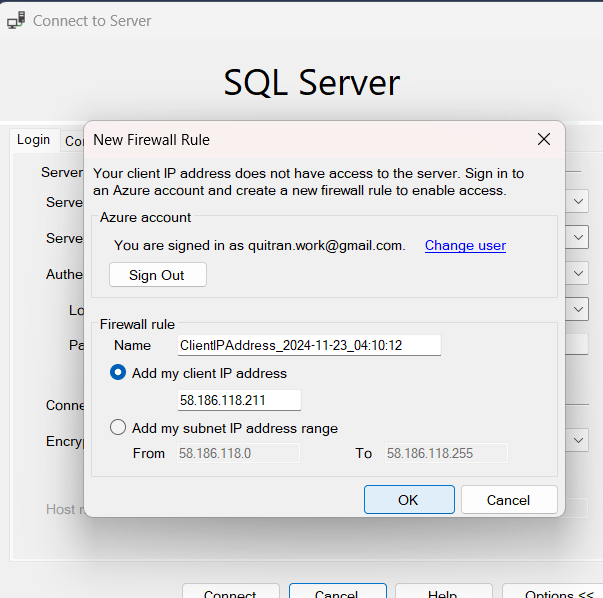
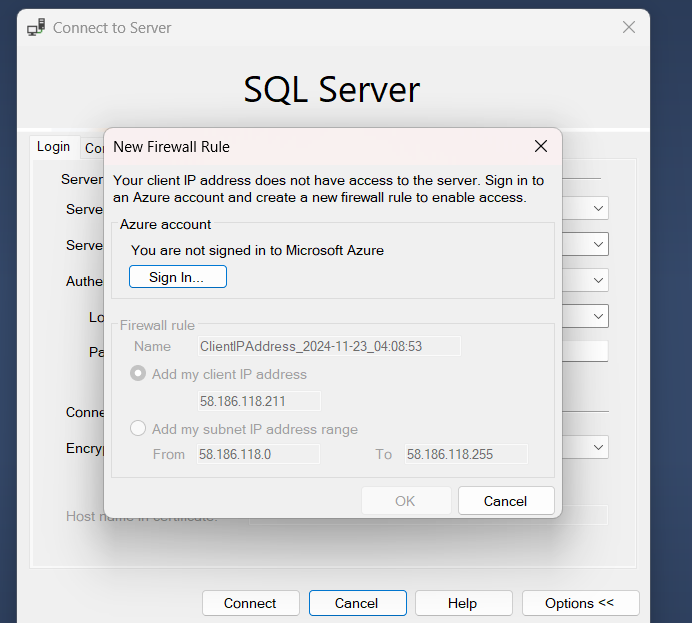


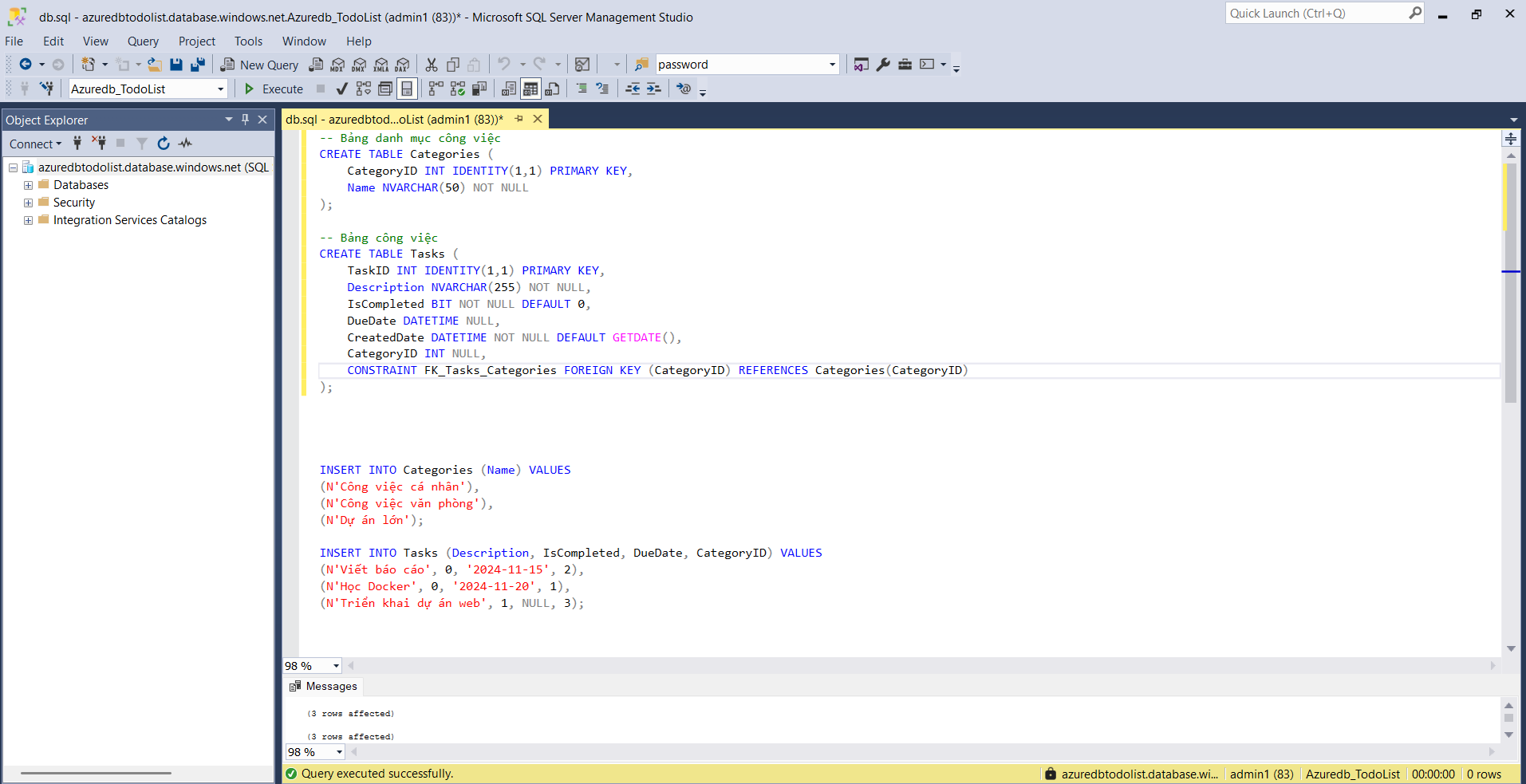


- Bước tiếp theo mở SQL Server và đăng nhập bằng tài khoản đã cấu hình bên trên.

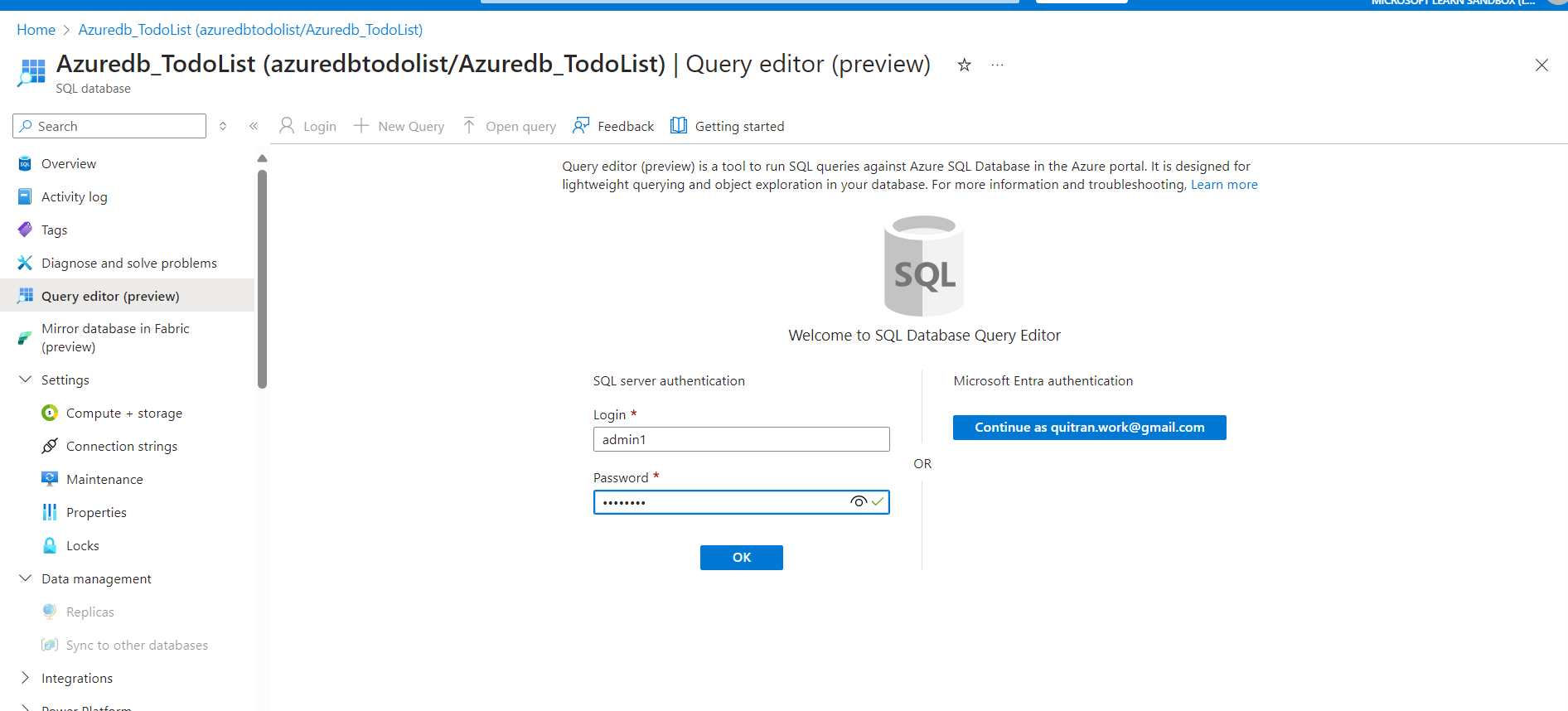
- Nhấn Connect và Sign in để chuyển hướng đăng nhập vào Azure Portal.

- Sau đó chuyển từ master sang database đã tạo khi cấu hình. Thực hiện thêm dữ liệu vào database.

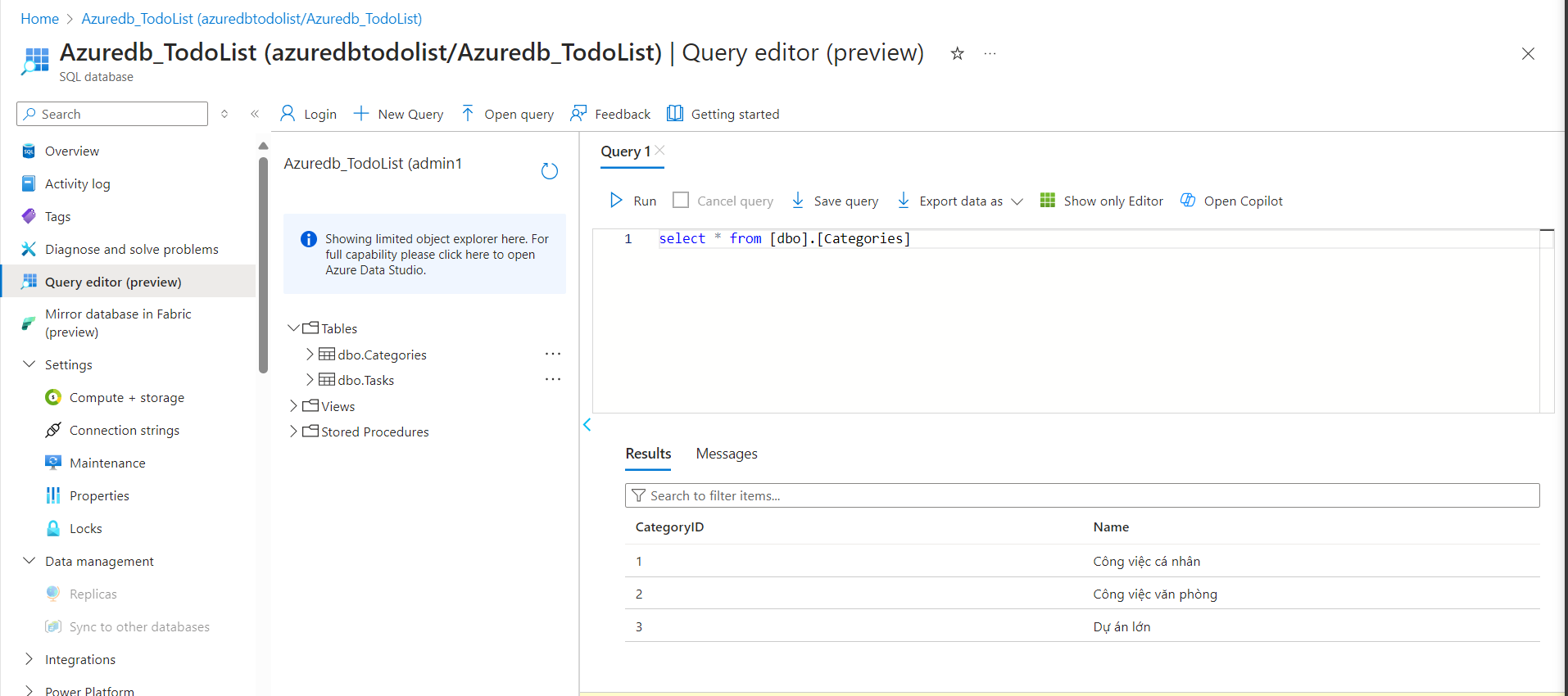




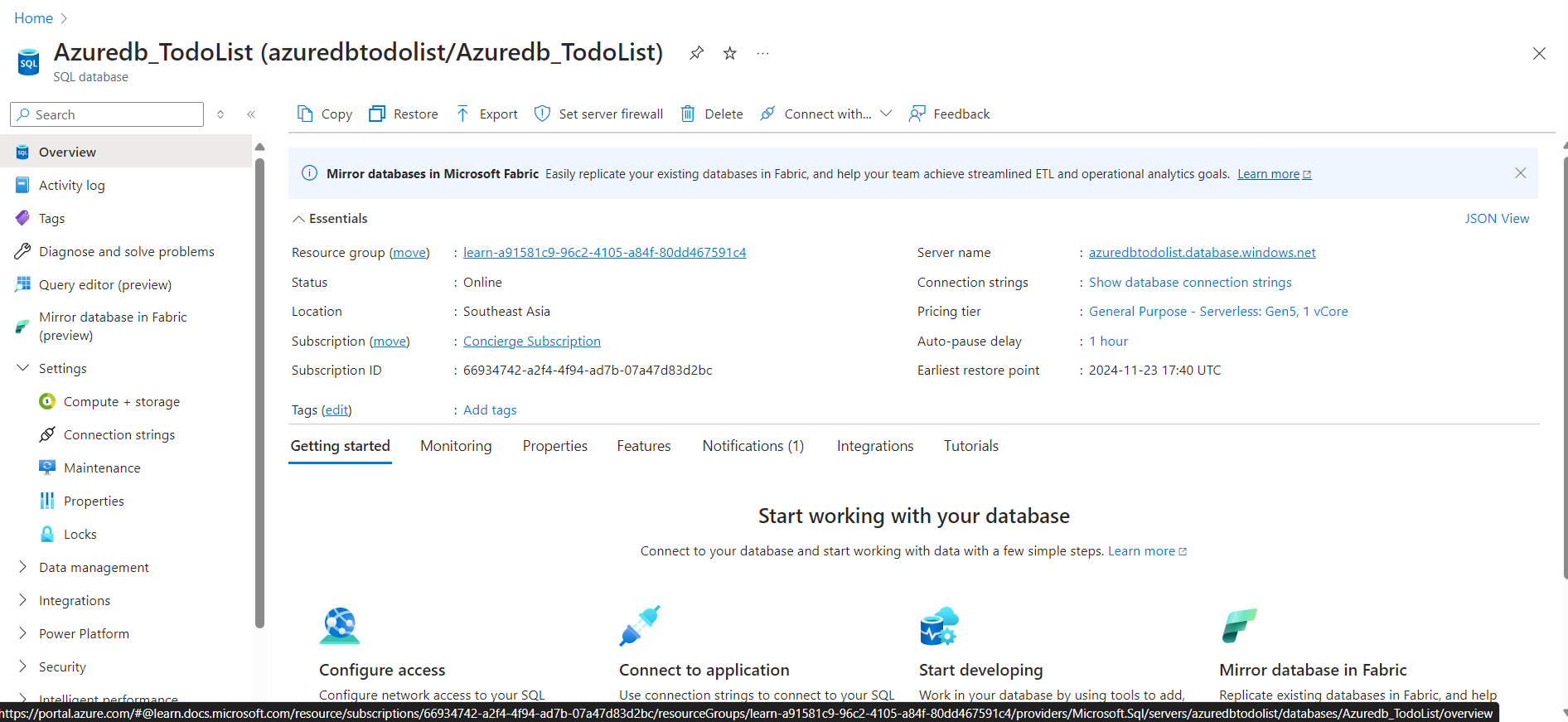
- Để kiểm tra trên cloud các dữ liệu có được thêm vào đồng nhất hay không. Ta quay lại SQL Database đã tạo trên Azure Portal chọn **Query Editor** sau đó đăng nhập bằng tài khoản ta đã cấu hình ở trên.

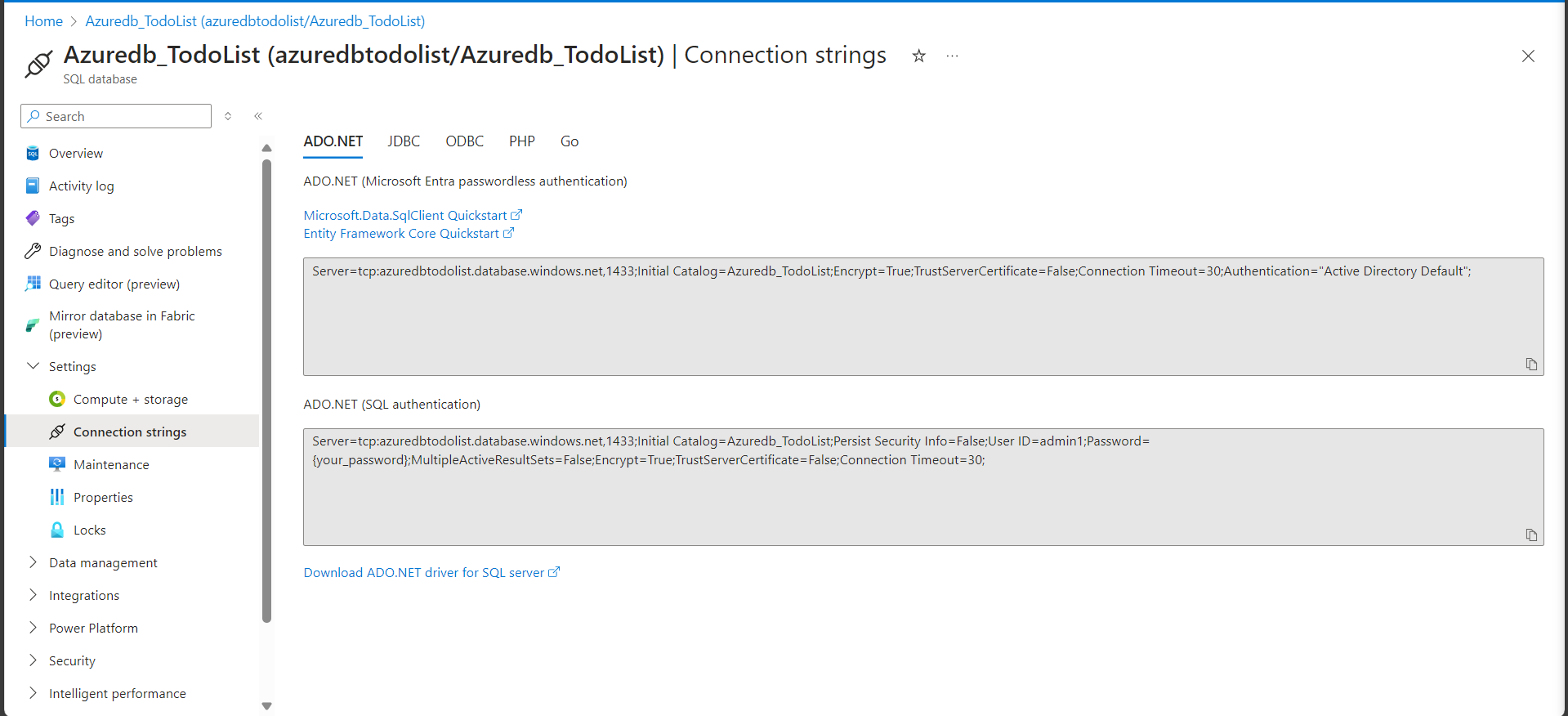


- Gõ lệnh truy vấn để kiểm thử.



- Như ta đã thấy ở trên dữ liệu đã được đồng nhất hoàn toàn giữa SQL Server và Azure SQL Database. Tất nhiên, database server thì phải có ứng dụng để đọc/ghi dữ liệu. Azure còn có một chức năng khá tiện lợi để lấy **Connection String** cho ứng dụng của bạn. Chỉ cần bấm vào Connection String bên menu trái, các bạn sẽ lấy được connection string cho app C#, Java, PHP,…Với database này, có thể dễ dàng chia sẻ connection để bạn bè, đồng nghiệp trong team cùng kết nối tới để dev app. Không còn lo cảnh mỗi đứa một database, không có data để test hoặc mỗi lần sửa database phải chạy script lại nữa.





Tương tự với lại tài khoản **Default Directory** các bước thực hiện trên sẽ tương tự và ta sẽ cấu hình theo nhu cầu của bản thân.

**Thực hiện build Image và push lên docker hub**

- Trước khi thực hiện các bưới dưới để push lên docker hub cần gõ docker login để đăng nhập vào **Docker Hub** hoặc đăng nhập vào Docker Desktop**.** Mở CMD trong thư mục của demo hoặc sử dụng CMD để điều hướng đến thư mục demo.

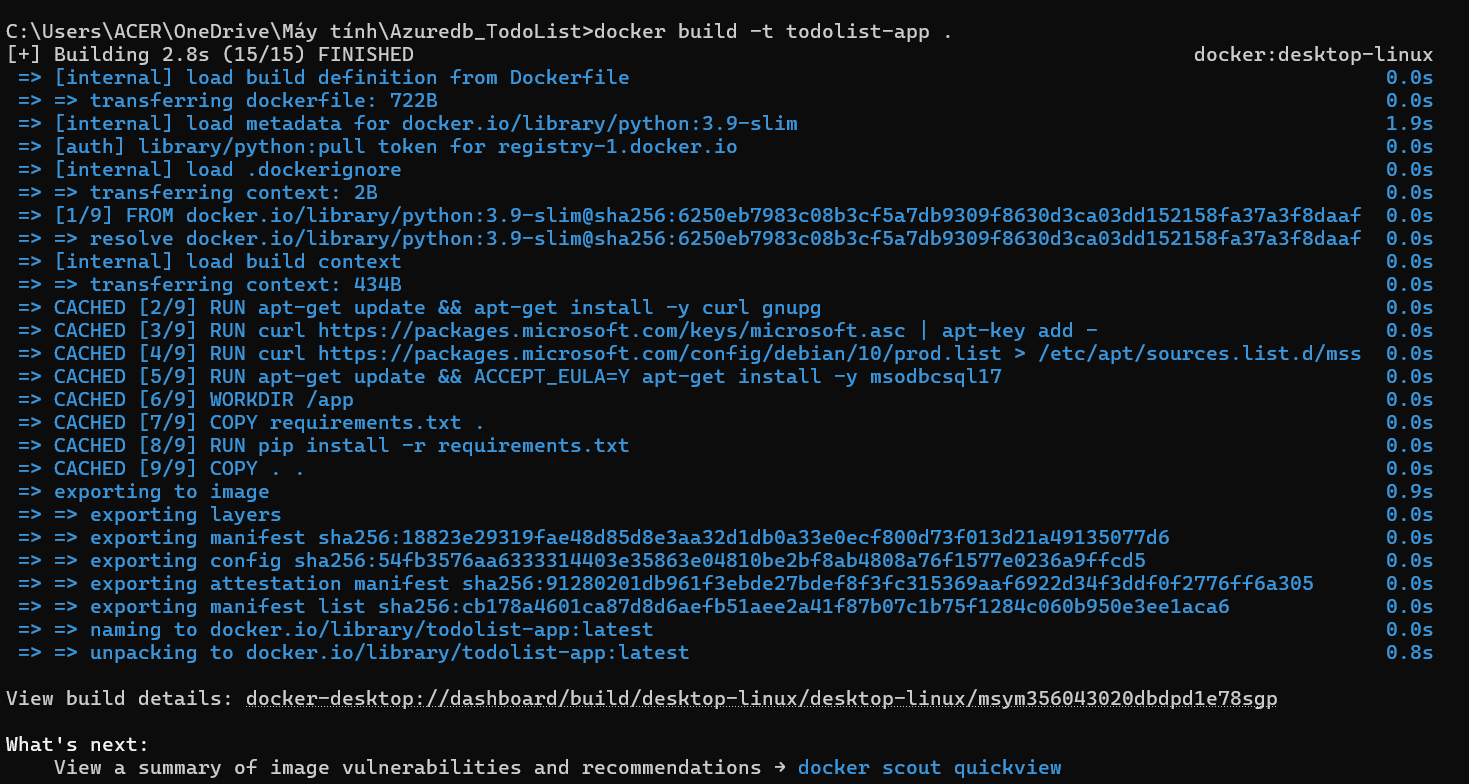
**Gõ lệnh(dấu chấm sau cuối câu lệnh biểu thị là dockerfile đang nằm trong thư mục hiện tại:**

-tcho phép ta đặt tag cho image nếu không đặt thì mặc định là latest

docker build -t <tên-image>:tag

docker build -t todolist-app .

docker images (lệnh kiểm tra các images đang có)

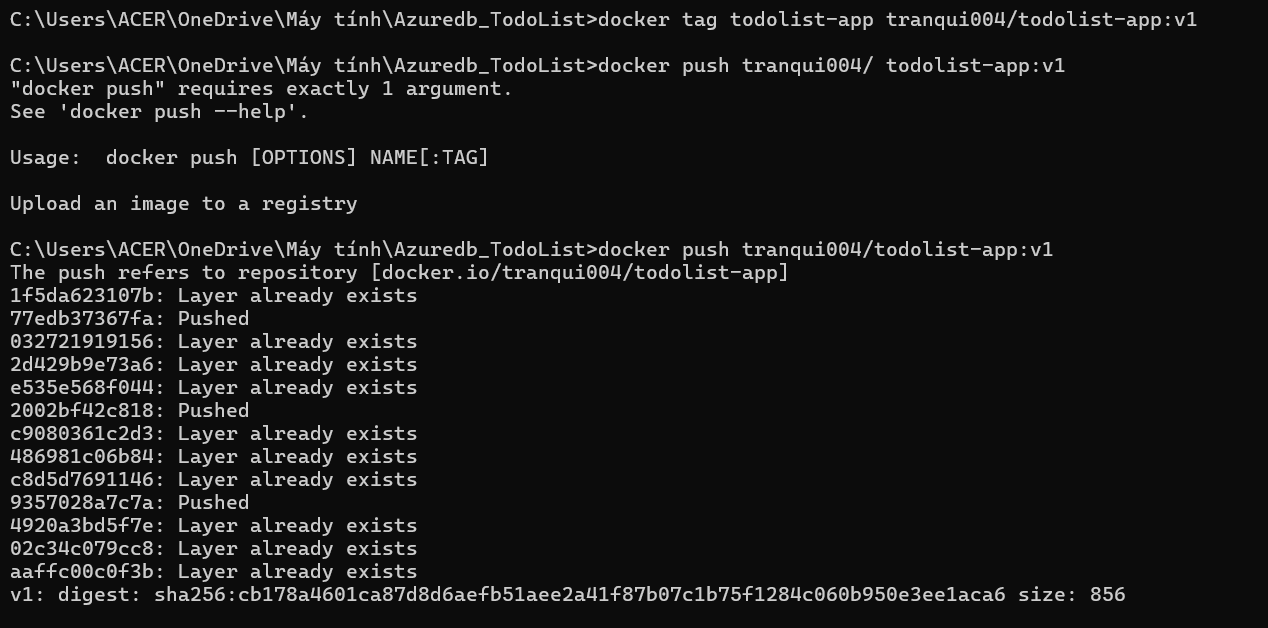
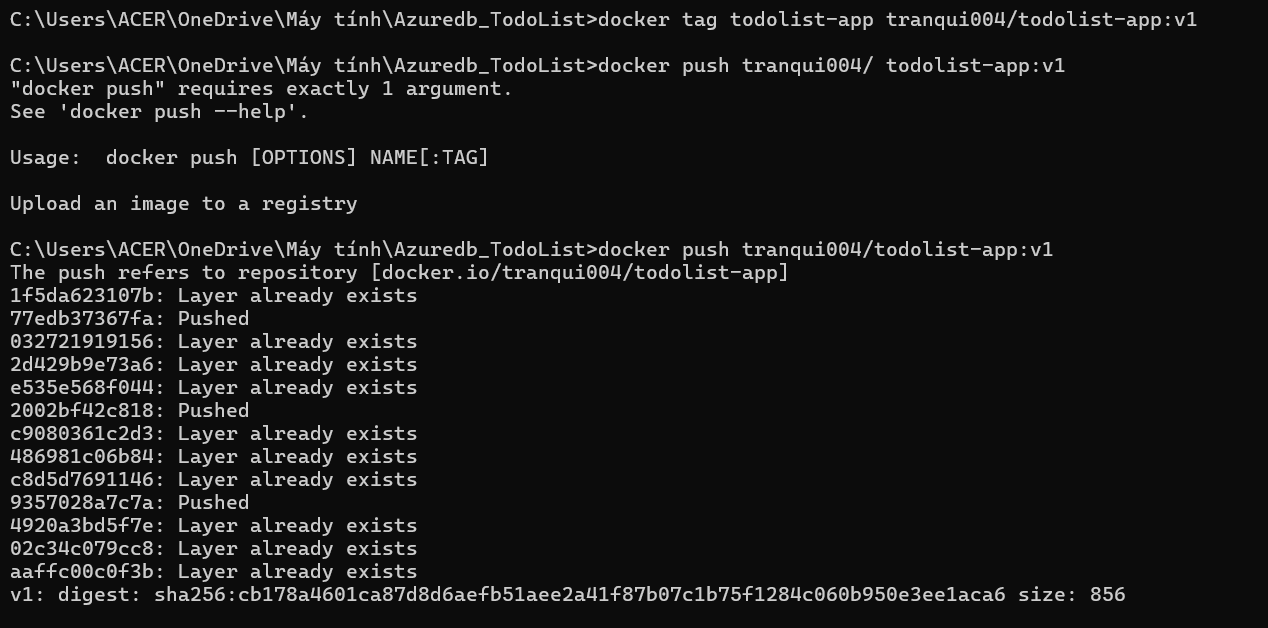
****

**Thực hiện đổi tag và push lên docker hub chú ý kĩ xem có dư dấu cách hoặc không đúng cú pháp không:**

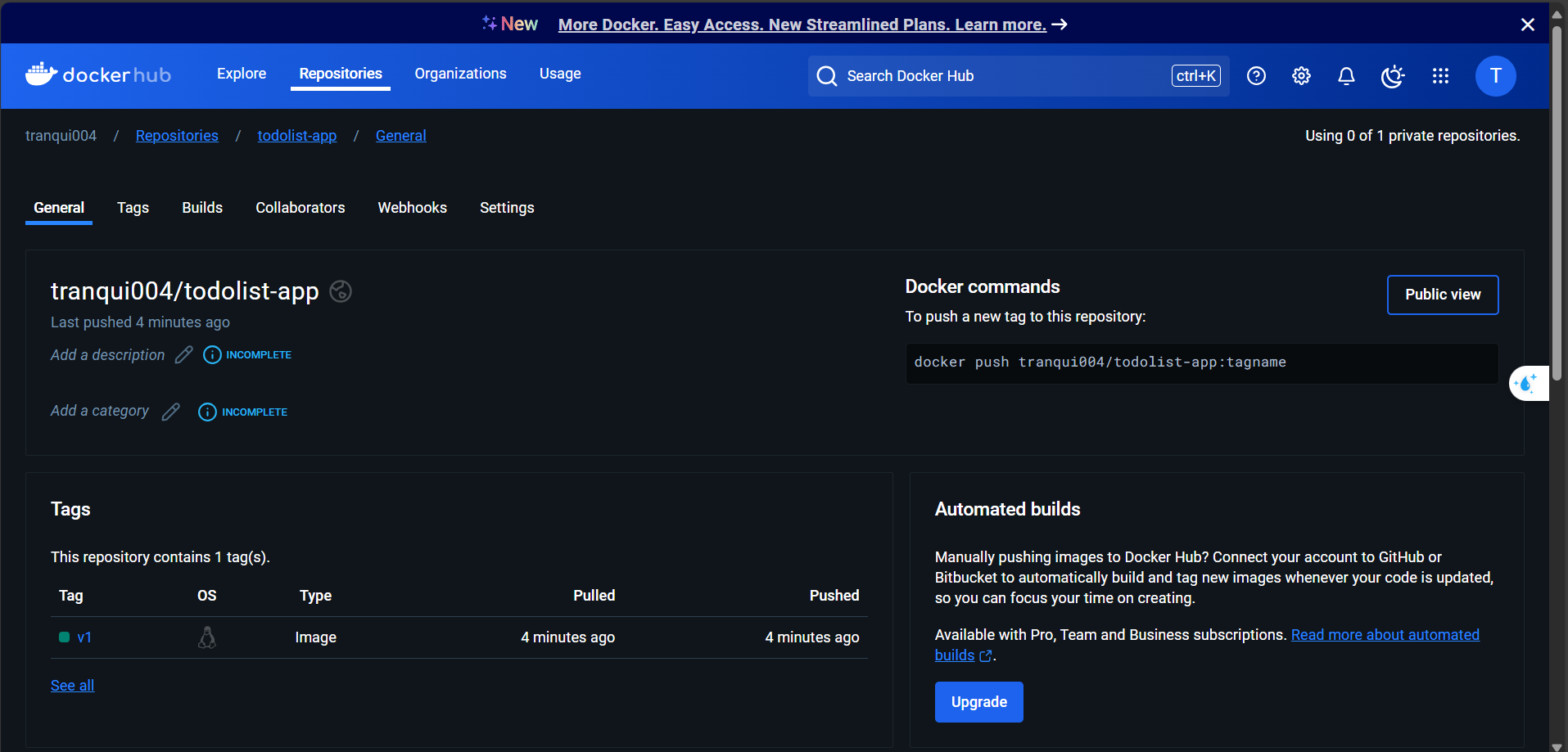
docker tag <image\_id> <your\_hub\_username>/<your\_repository>:<tag>

docker tag todolist-app tranqui004/todolist-app:v1

docker push tranqui004/todolist-app:v1



Truy cập vào trang Docker Hub <https://hub.docker.com/> để kiểm tra xem image có được push lên chưa. Như ta đã thấy sau khi thực hiện các bước trên thì images đã được push ở Repository tranqui004 thành công.



**Thực hiện pull image từ Docker Hub và triển khai container.**

Để kiểm tra sự tiện lợi của docker là chỉ cần build 1 lần là có thể xây dựng và chạy bất kì đâu thì sau đây chúng ta sẽ thử pull image từ trên docker hub ở bước trên vừa triển khai về và tiến hành tạo container và kiểm tra xem demo có được triển khai nhờ image được pull về hay không.

**Triển khai demo thành 1 web app server on-cloud**

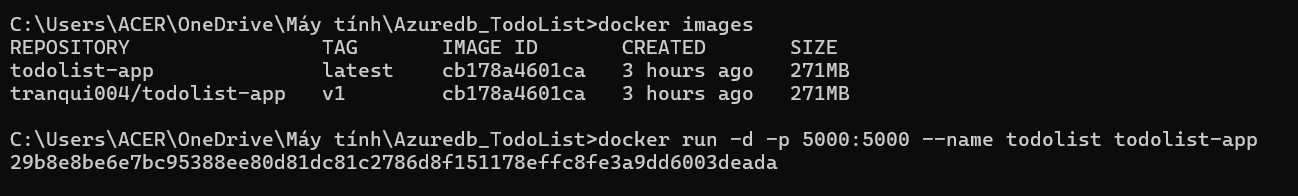
Trước khi triển khai demo thành 1 web app on-cloud thì chúng ta sẽ kiểm tra lại 1 lần nữa là demo có hoạt động tốt ở local hay không, ta sẽ xây dựng container để kiểm tra.

docker run --name <container\_name> -d -p <host\_port>:<container\_port> <image\_name>:<tag>

docker run -d -p 5000:5000 --name todolist todolist-app:v1

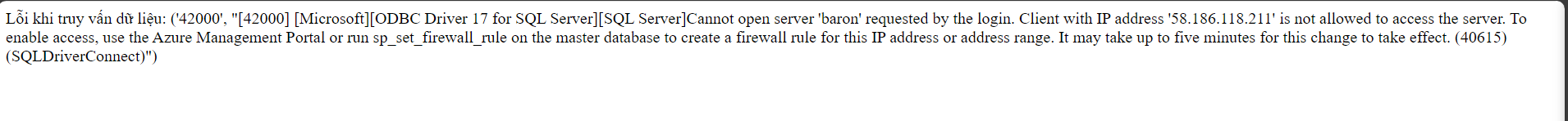
-d: tuỳ chọn Detached mode cho phép chạy nền trong terminal

-p <host\_port>:<container\_port>: ánh xạ (map) một cổng trên máy chủ (host) tới một cổng trong container.

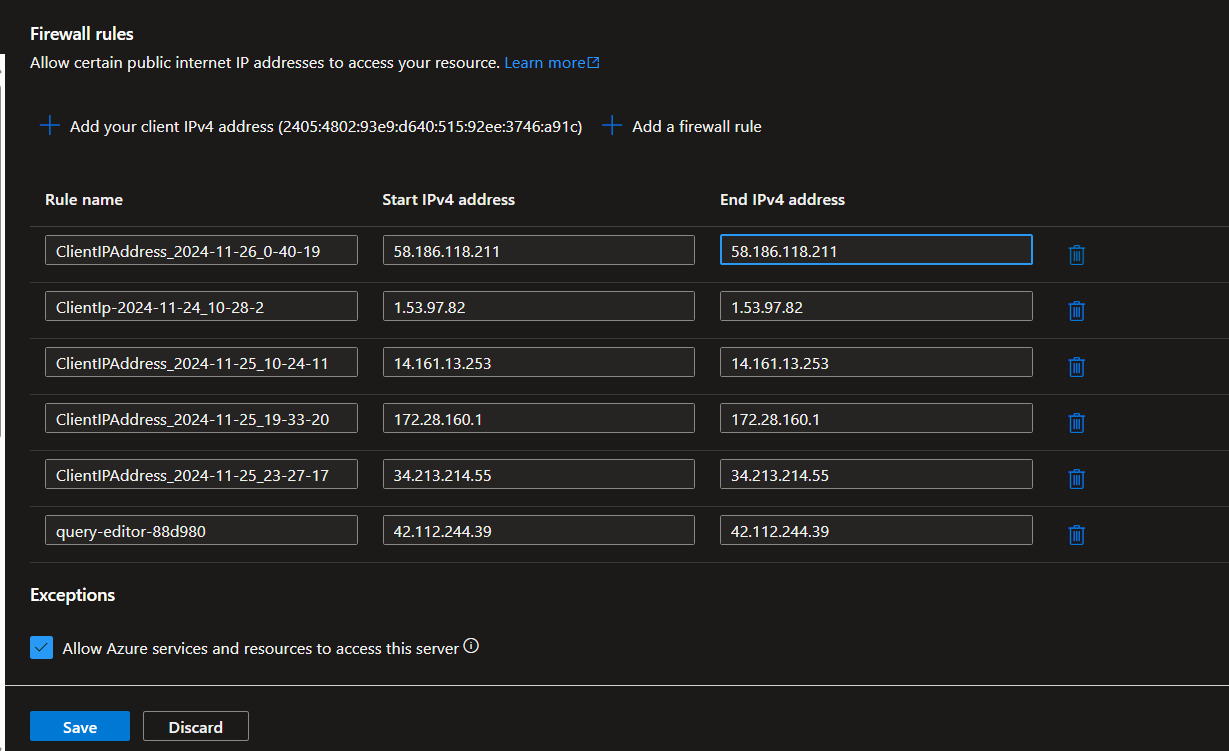


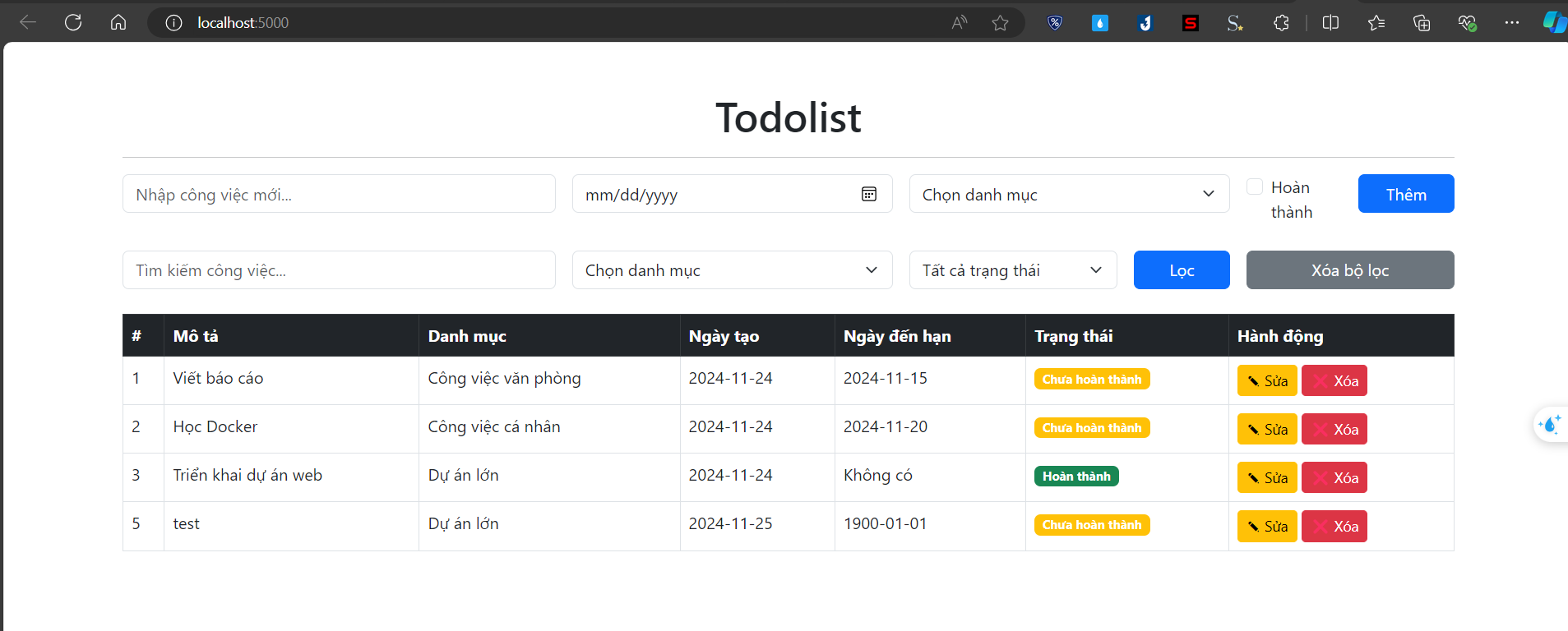
Ở đây demo sẽ chạy ở cổng 5000 ta truy cập <http://localhost:5000/> để kiểm tra

Nếu thông báo như sau ta sẽ quay lại Azure SQL Database đã tạo trước đó và allow ip bên dưới



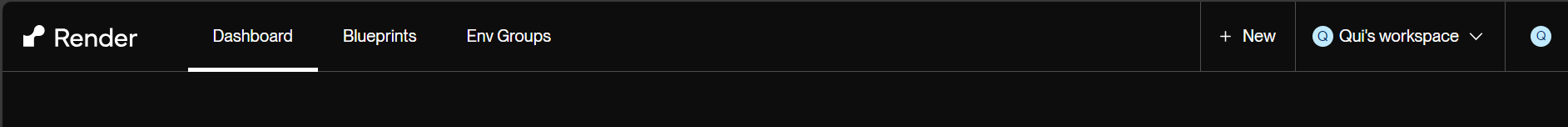
Bấm **Set server firewall** lăn chuột xuống và bấm **Add your client IPv4** sau đó add ip được khoanh như ảnh trên vào 2 ô **Start IPv4 address** và **End IPv4 address** sau đó bấm Save và quay trở lại kiểm tra web đã hoạt động được chưa.





Nếu đã hoạt động ổn thì ta tiến hành bước triển khai thành web app on cloud cho demo

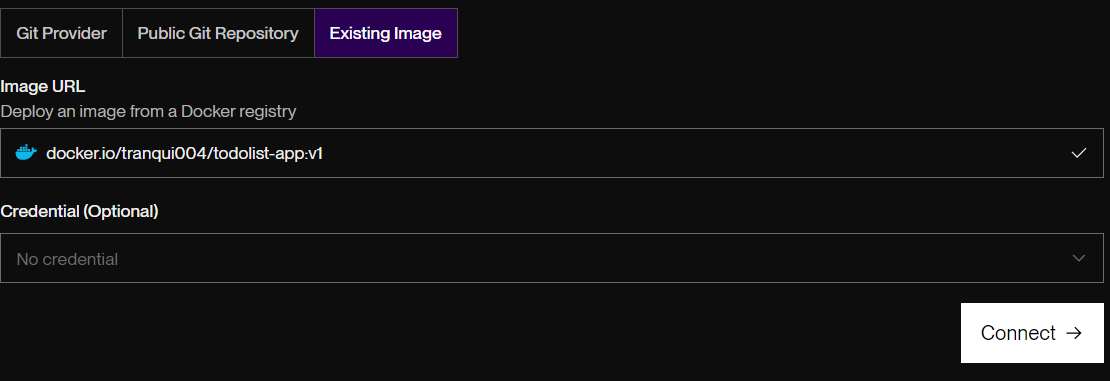
Ta truy cập <https://render.com/> sau đó tạo tài khoản hoặc đăng nhập. Sau khi đăng nhập xong ở thanh menu bên trên ta thấy được mục **New** nhấp vào đó chọn mục **Web Service** để tạo cloud application platform.



Sau khi nhấp **Web Service** ta có 3 mục như ảnh ta chọn qua mục **Existing Image**.

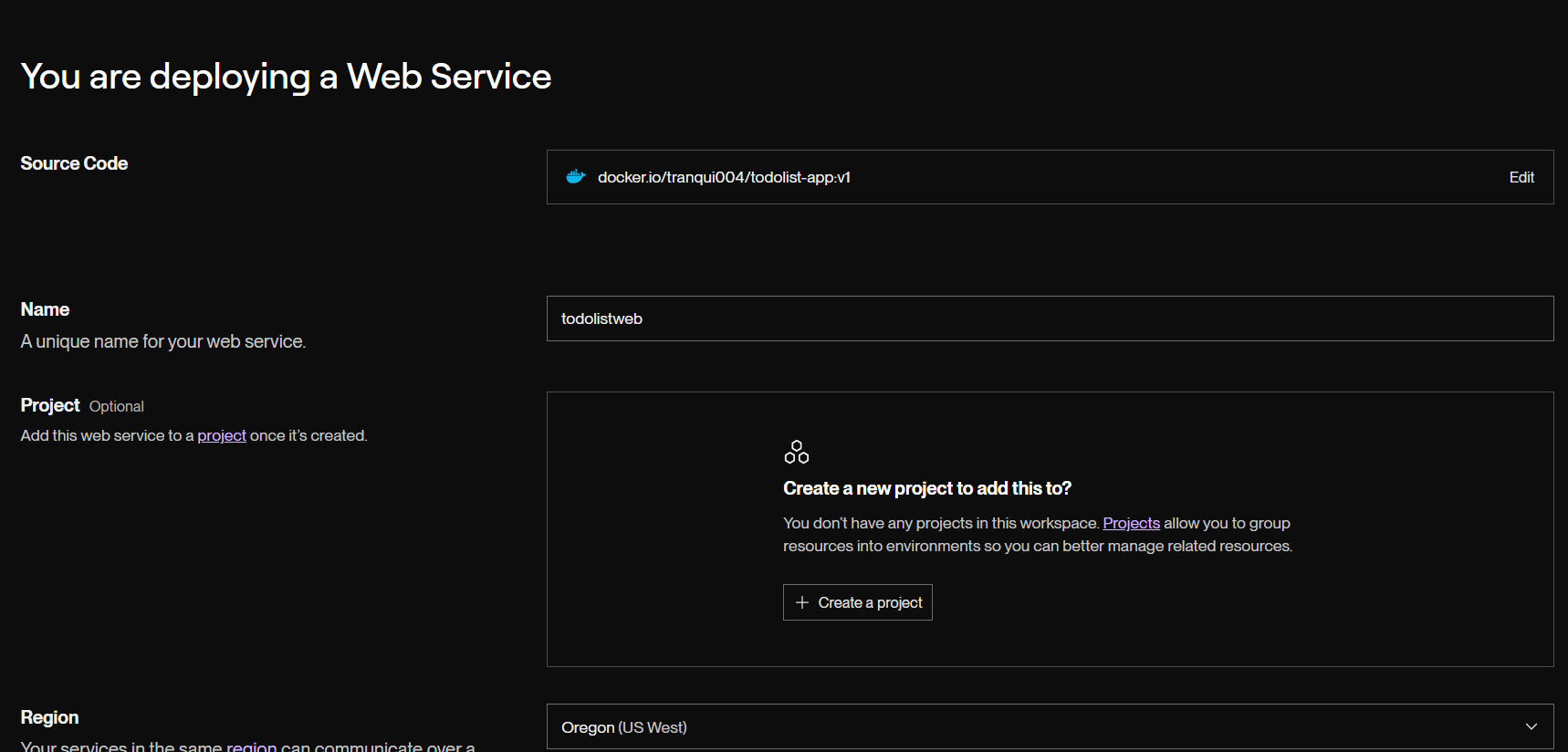
**Trong đó:**

* Git Provider: liên kết với tài khoản git.
* Public Git Repository: sử dụng link repository của git được public.
* Existing Image: Lấy image trên Docker Hub để deploy

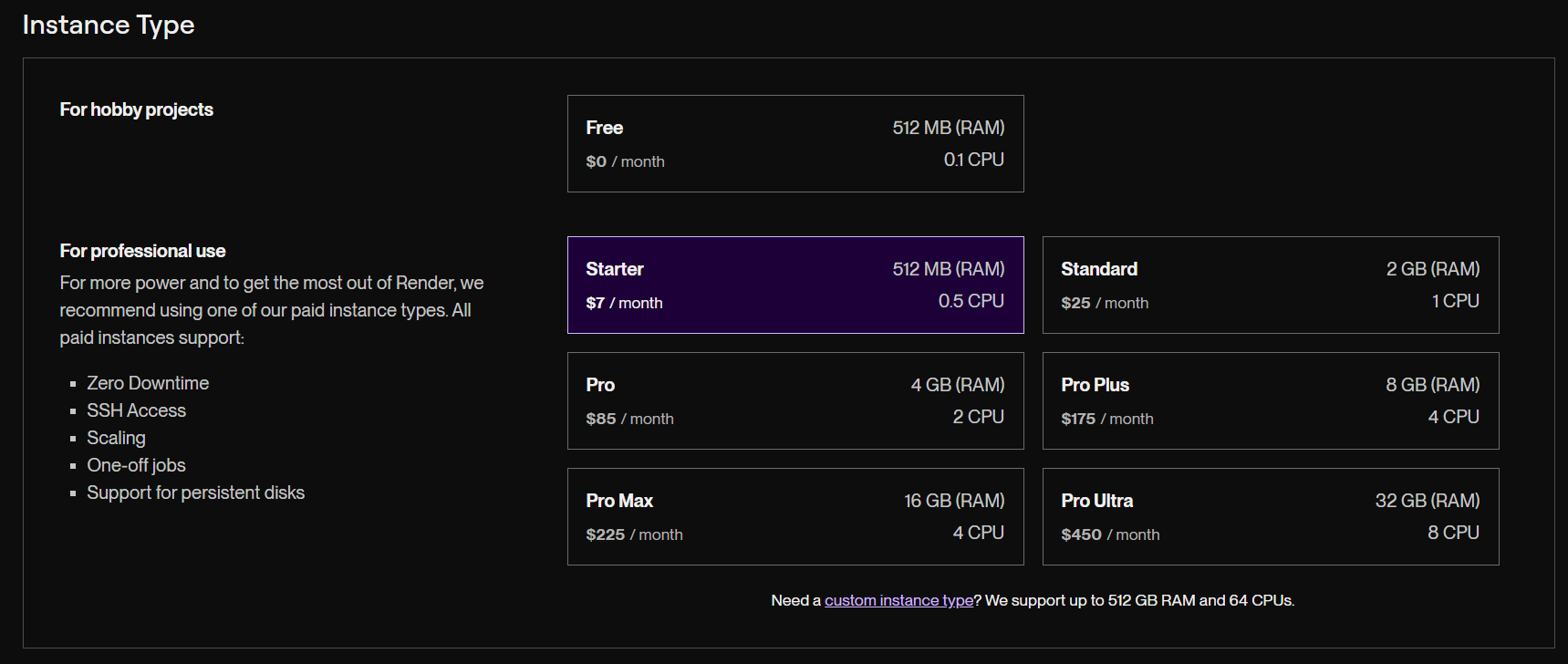
Khi chọn **Existing Image** sẽ hiện ra giao diện như ảnh bên trên, ô đầu tiên ta sẽ điền đường link đến image cần deploy, ô thứ 2 để nhập chứng chỉ nếu không có ta bỏ qua. Nhấn **Connect**.

Sau khi nhấn **Connect** thì ta sẽ thiết lập cấu hình cho dịch vụ web:

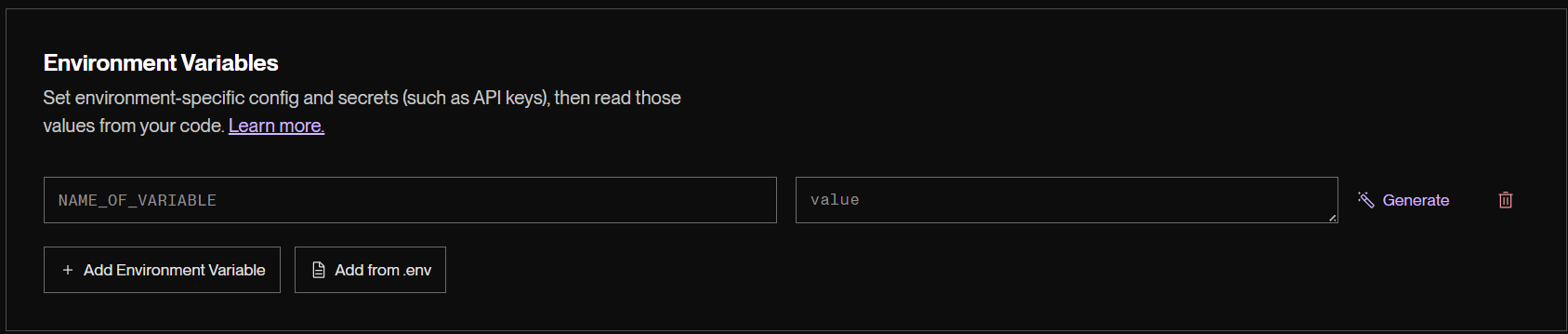
* Source code: là đường dẫn đến image mà chúng ta đã push lên docker hub.
* Name: Đơn giản chỉ là đặt tên cho trang web.
* Region: Chọn vùng

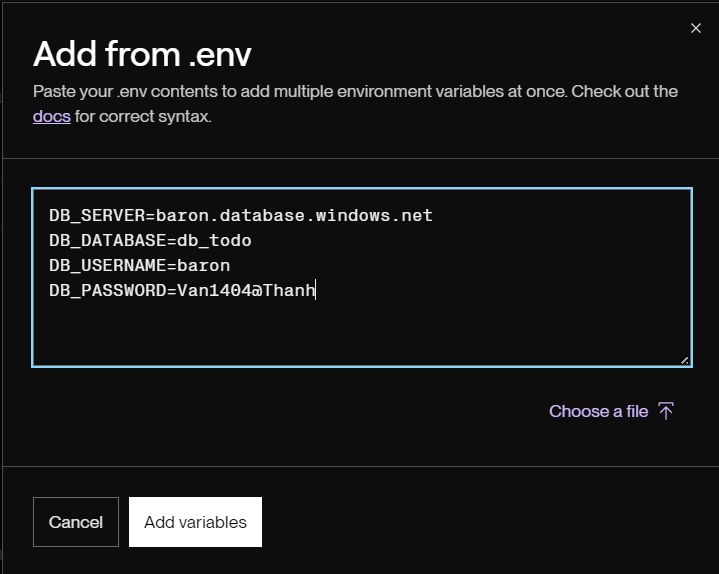


Lựa chọn gói chi phí để triển khai web ở đây ví dụ ta chọn Free. Tuy nhiên là web ở gói dịch vụ này sẽ tạm ngừng hoạt động nếu như không có lưu lượng truy cập dẫn đến độ trễ cao



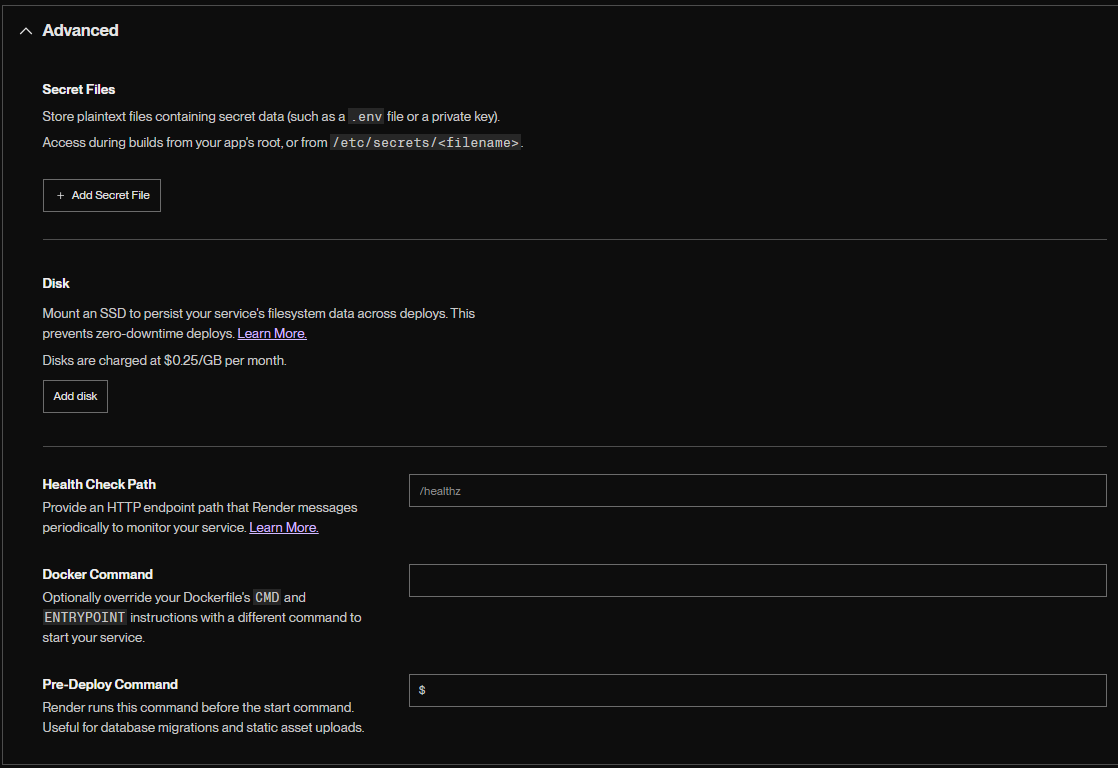
Lướt xuống dưới ta sẽ thấy mục **Environment Variables**(biến môi trường) bởi vì demo của chúng ta ở đây có kết nối với **Azure SQL Database** nên chúng ta phải cấu hình ở phần này như sau nếu không có kết nối với database thì ta có thể bỏ qua. Nhấp Add Environment Variable để thêm thủ công từng ô hoặc Add from env để thêm cấu hình kết nối từ file .env(nếu có) hay gõ cấu hình kết nối ở đây ta dùng **Add from env** cho nhanh. Nhớ rằng biến môi trường trong demo chúng ta như thế nào thì ở đây cũng phải triển khai y như thế đó.





Sau đó bấm **Add variables** là hoàn tất.

Xuống 1 chút nữa là phần **Advanced**



**Secret Files (Tệp bí mật)**

* **Mục đích**: Dùng để lưu trữ các dữ liệu nhạy cảm (ví dụ: khóa API, thông tin đăng nhập cơ sở dữ liệu, hoặc tệp .env) một cách an toàn.
* **Truy cập**: Các tệp này sẽ được ứng dụng truy cập trong quá trình xây dựng hoặc khi chạy tại đường dẫn /etc/secrets/<tên-tệp>.

**Disk (Ổ đĩa)**

* **Mục đích**: Cho phép gắn thêm bộ nhớ SSD để lưu trữ dữ liệu của ứng dụng, giúp duy trì dữ liệu qua các lần triển khai (deployment). Điều này rất hữu ích cho các ứng dụng như cơ sở dữ liệu hoặc ứng dụng có chức năng tải lên tệp.
* **Chi tiết**:
  + Ổ đĩa đảm bảo triển khai không bị gián đoạn (**zero-downtime**).
  + Chi phí là **$0.25/GB mỗi tháng**.

**Health Check Path (Đường dẫn kiểm tra trạng thái)**

* **Mục đích**: Xác định một endpoint HTTP (ví dụ: /healthz) mà nền tảng sẽ kiểm tra định kỳ để đảm bảo dịch vụ đang hoạt động bình thường.
* **Cách hoạt động**: Nếu endpoint không phản hồi, nền tảng có thể khởi động lại dịch vụ hoặc gửi thông báo lỗi.
* **Mặc định**: /healthz thường được sử dụng, nhưng bạn có thể đặt thành bất kỳ endpoint nào mà ứng dụng của bạn hỗ trợ.

**Docker Command (Lệnh Docker)**

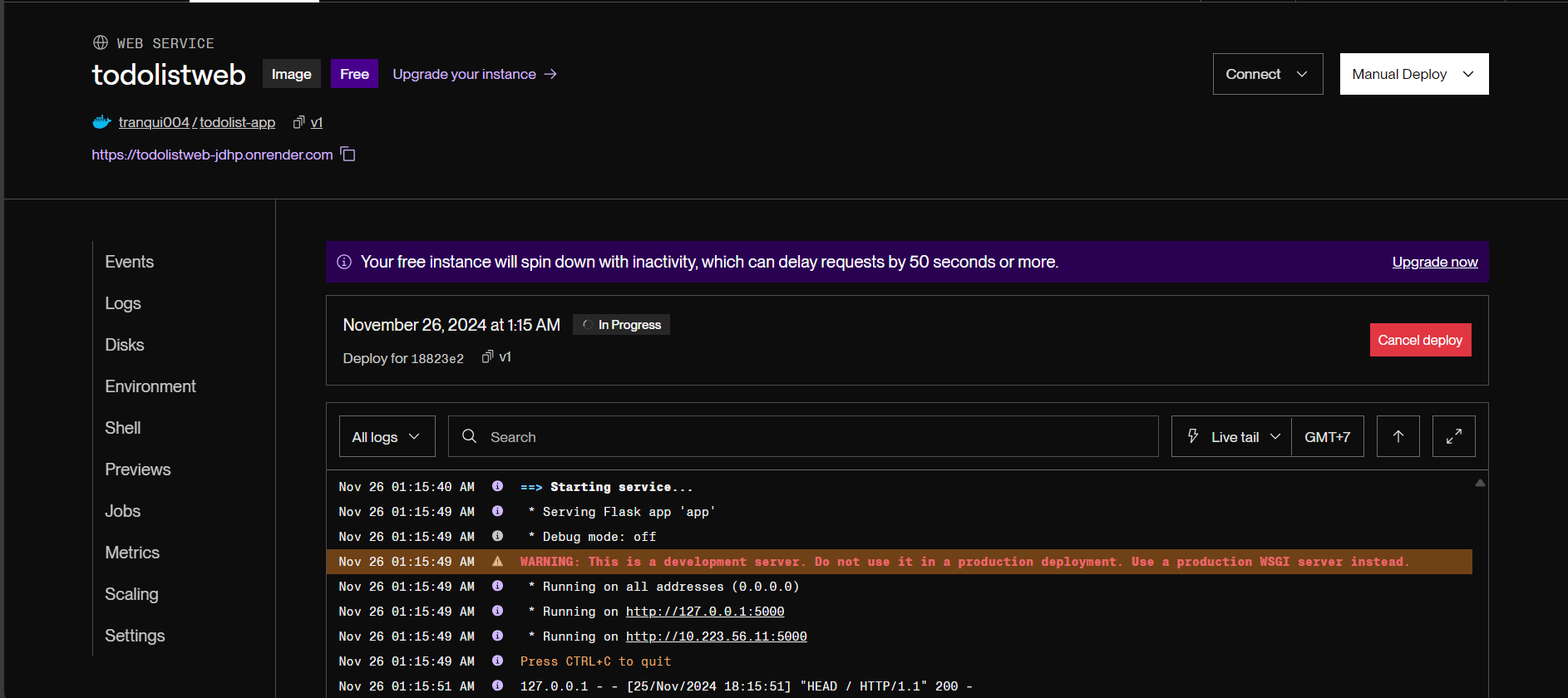
* **Mục đích**: Ghi đè lệnh CMD hoặc ENTRYPOINT mặc định được định nghĩa trong tệp Dockerfile.
* **Trường hợp sử dụng**: Nếu bạn muốn tùy chỉnh cách container khởi chạy (ví dụ: để gỡ lỗi hoặc dùng một script khởi động khác), bạn có thể điền lệnh vào đây

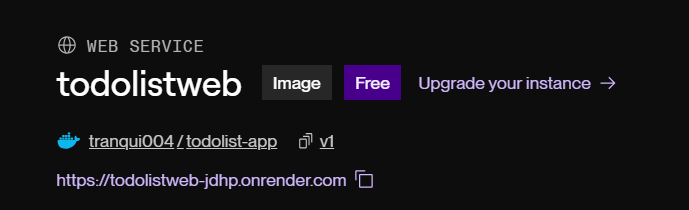
**Pre-Deploy Command (Lệnh trước khi triển khai)**

* **Mục đích**: Chạy một lệnh **trước khi** ứng dụng chính được khởi động trong quá trình triển khai.
* **Trường hợp sử dụng**: Thường được sử dụng cho các công việc như:
  + Chạy lệnh cập nhật cơ sở dữ liệu (database migrations).
  + Tải lên các tệp tĩnh (static assets).
* **Ví dụ**: Đối với một ứng dụng Django, bạn có thể chạy lệnh:

python manage.py migrate

Nhưng ở đây ta sẽ không cần sử dụng tới bất kì phần nào trong Advanced nên ta sẽ bỏ qua và nhấn Deloy Web Service để tạo. Sau đó ta sẽ thấy được giao diện như sau



Ở phần bảng Log ngay bên phải ta chờ cho web được xây dựng và khi thấy dòng Your service is live thì web đã hoạt động ta chỉ cần truy cập đường link như trên ảnh để kiểm tra web có hoạt động hay không.

**Kết quả**

