Does AWS Cognito remove the need for a 'users' table in my database?

Asked 5 years, 10 months ago Modified 5 years, 10 months ago

Viewed 10k times Part of AWS Collective

52

With AWS Cognito doing its thing with authentication does this mean I no longer need a traditional 'users' table in my database?

Currently the app I have inherited has the traditional 'users' table I reference with sql queries looking for many-to-many and many-to-one joins with organisation, message boards, etc tables.

Do I query the Cognito user groups (that feels wrong) or is there some 'sync' method to have an updated 'users' table in my database.

Reading to docs and watching a few tutorials is great but no one seams to do more than authenticate. Where is the authorisation, and getting user names, email with join queries.

Be gentle, trying to work and study with a new baby in the room

aws amazon-web-services amazon-cognito

Share

Improve this question

Follow

asked Jan 30, 2019 at 11:22

user3067684
1,260 ● 1 ● 12 ● 24

2 Answers

Sorted by: Highest score (default)

57

My experience has been it does not obsolete the need for a users table, for a couple of reasons.

1. I would routinely run into AWS throttling errors when invoking the Cognito getUser/listUser methods while using Cognito as the primary user data store. AWS support would increase the API limits on our account but I was always worried they would reappear.
2. You are essentially limited to querying users by username/email/phone. The listUser method is very limited for searching
3. If you want to store other user data then you have to put them in custom Cognito attributes and managing those got tiresome quickly

The design I ended up on was to set a post-confirmation trigger on the Cognito user pool and have it copy all the user data into a relational database or DynamoDB when a user signed up. The primary key of the users table would be the Cognito username so if necessary I could lookup the user in both the database and Cognito. Basically I just use Cognito for authentication and not as a primary user database

Share Improve this answer Follow

answered Jan 31, 2019 at 11:47

Brian Winant
3,035 ● 16 ● 19

3 Many thanks Brian, thats the path I wanted to take - the best takeaway from your answer is the "post-confirmation trigger". – user3067684 Jan 31, 2019 at 19:32

1 How does this work if a user changes an attribute? I didn't see a lambda trigger for attribute updates. – Chad Greenburg Jun 9, 2019 at 20:55

5 Correct, there is no update user trigger on Cognito and that is annoying. We ended up just using Cognito for basic authentication and authorization and keeping all user profile data in RDS/Dynamo – Brian Winant Jun 14, 2019 at 11:10

If post confirmation trigger failed multiple times cognito will retry up to 3 times but in case maybe the database is down and all retires have failed, how do you keep the database consistent with cognito? – Ahmad Nabil Dec 27, 2021 at 23:48

1 What I could think of right now is to use SQS with a `Post confirmation` trigger together with a `Pre sign-in` trigger which first checks if the user exists in the database, if not then deny the sign in request, that way the sign in request can only succeed when the message in the queue is finally processed. – Ahmad Nabil Dec 28, 2021 at 2:13

9

You can certainly use Cognito User Pools service to manage authentication instead of your 'user' table, but not necessarily. You can also integrate other identity providers and login with Twitter, Amazon, etc, or even your own database. A benefit of using Cognito User Pools is that you don't have to code the authentication flows, as they are implemented by the service correctly and securely.

Best of luck with the code and the baby!

Share Improve this answer Follow

answered Jan 30, 2019 at 13:03

Julio Faerman
13.5k ● 9 ● 61 ● 80