

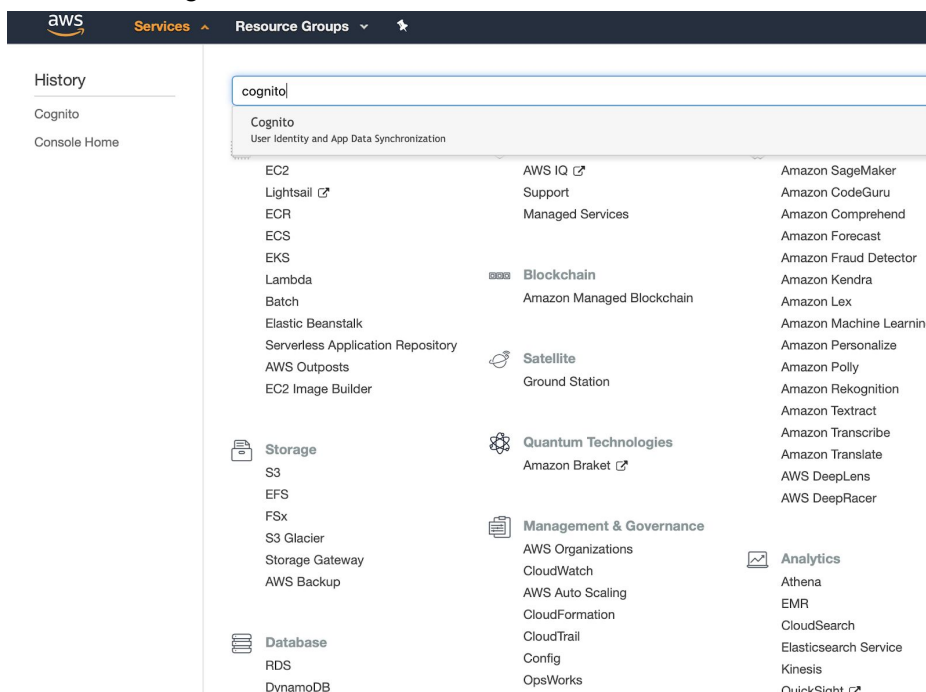
Tutorial/Lab

More AWS: Cognito

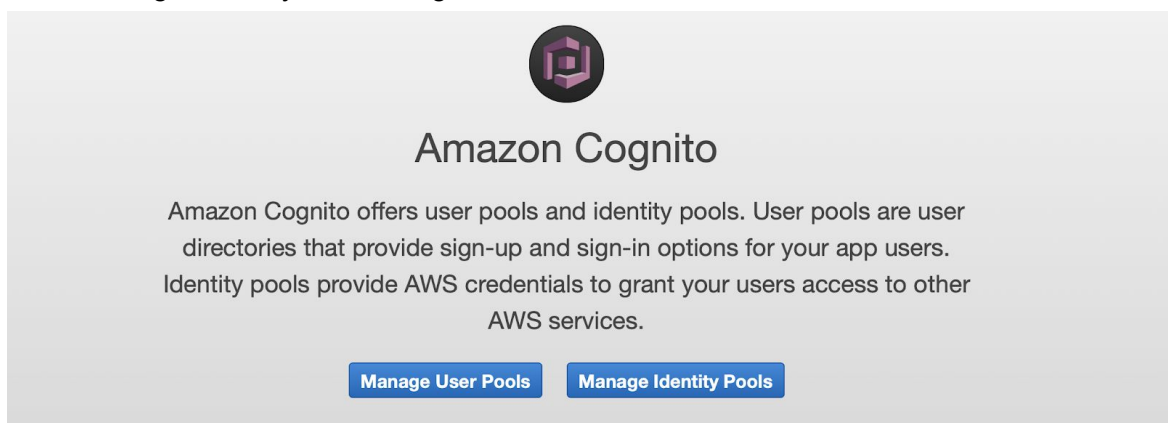
For this tutorial/lab you will first walk through the steps to create an AWS Cognito User Pool. After the tutorial, you will create a simple Android application to use the user pool you have created.

Create an AWS Cognito User Pool

1. Search for Cognito in the AWS Console



2. Click on Cognito and you should get to a screen that looks like this.



3. Click “Manage User Pools” and then “Create a user pool” in the top right corner.
4. The setup screen should look like this. Give your user pool a descriptive name such as Your_App_Name_Users.

5. At this point there are two options: “Review defaults” or “Step through settings”. We are going to choose “Step through settings” to help you better understand what some of the different parts mean.
6. The next step is “Attributes”. For the first part, keep username selected. This means that users will have to sign up and sign in with a username instead of email or phone number. For the required standard attributes we want to select email and given name. This means we are requiring them to enter both of these when they sign up. Password is a mandatory field which is why you don’t see it listed. We won’t add any custom attributes for this user pool.

7. The next step is “Policies”. We won’t change anything on this page, but if you wanted to require your users to have a more or less secure password, you would do that here.

The screenshot shows the 'Create a user pool' page in the AWS IAM console. The left sidebar contains a menu with options: Name, Attributes, Policies (highlighted), MFA and verifications, Message customizations, Tags, Devices, App clients, Triggers, and Review. The main content area is titled 'What password strength do you want to require?'. It includes a 'Minimum length' input field set to 8, and four checked checkboxes: 'Require numbers', 'Require special character', 'Require uppercase letters', and 'Require lowercase letters'. Below this is a section titled 'Do you want to allow users to sign themselves up?' with two radio buttons: 'Only allow administrators to create users' and 'Allow users to sign themselves up' (selected). A note states: 'You can choose to only allow administrators to create users or allow users to sign themselves up. [Learn more.](#)'. The next section is 'How quickly should temporary passwords set by administrators expire if not used?' with a note: 'You can choose for how long until a temporary password set by an administrator expires if the password is not used. This includes accounts created by administrators.' and a 'Days to expire' input field set to 7. A 'Cancel' button is visible in the top right corner.

8. The next step is “MFA and verifications”. Here we will make sure that MFA is off. Under “which attribute do you want to verify?” make sure that **email** is selected. We won’t need to change anything else.

The screenshot shows the 'Create a user pool' page in the AWS IAM console, with the 'MFA and verifications' tab selected in the left sidebar. The main content area is titled 'Do you want to enable Multi-Factor Authentication (MFA)?'. It includes a paragraph explaining MFA and a note: 'Note: separate charges apply for sending text messages.' Below this are three radio buttons: 'Off' (selected), 'Optional', and 'Required'. The next section is 'How will a user be able to recover their account?' with a paragraph explaining recovery options and a note: 'We recommend not allowing phone to be used for both password resets and multi-factor authentication (MFA). [Learn more.](#)'. Below this are six radio buttons: 'Email if available, otherwise phone, but don't allow a user to reset their password via phone if they are also using it for MFA' (selected), 'Phone if available, otherwise email, but don't allow a user to reset their password via phone if they are also using it for MFA', 'Email only', 'Phone only, but don't allow a user to reset their password via phone if they are also using it for MFA', '(Not Recommended) Phone if available, otherwise email, and do allow a user to reset their password via phone if they are also using it for MFA', and 'None - users will have to contact an administrator to reset their passwords'. The next section is 'Which attributes do you want to verify?' with a paragraph explaining verification and a note: 'Verification of a phone or email is necessary to automatically confirm users and enable recovery from forgotten passwords. [Learn more about email and phone verification.](#)'. Below this are four radio buttons: 'Email' (selected), 'Phone number', 'Email or phone number', and 'No verification'. The final section is 'You must provide a role to allow Amazon Cognito to send SMS messages' with a note: 'Amazon Cognito needs your permission to send SMS messages to your users on your behalf. [Learn more about IAM roles.](#)'. A 'Cancel' button is visible in the top right corner.

9. We are now going to skip down to “App clients” (do this by clicking on “App clients” in the left sidebar menu) and click on “Add an app client”. Feel free to read the sections we passed over to see what they do.

Create a user pool Cancel

Name
Attributes
Policies
MFA and verifications
Message customizations
Tags
Devices
App clients
Triggers
Review

Which app clients will have access to this user pool?

The app clients that you add below will be given a unique ID and an optional secret key to access this user pool.

[Add an app client](#) [Return to pool details](#)

10. Your screen should look like this. Now enter an app client name (this will be an Android app that you create shortly). Make sure “Generate client secret” is selected and hit “Create app client”.

Create a user pool Cancel

Name
Attributes
Policies
MFA and verifications
Message customizations
Tags
Devices
App clients
Triggers
Review

Which app clients will have access to this user pool?

The app clients that you add below will be given a unique ID and an optional secret key to access this user pool.

App client name

Refresh token expiration (days)

☒ Generate client secret

Auth Flows Configuration

☐ Enable username password auth for admin APIs for authentication (ALLOW_ADMIN_USER_PASSWORD_AUTH) [Learn more.](#)

☒ Enable lambda trigger based custom authentication (ALLOW_CUSTOM_AUTH) [Learn more.](#)

☐ Enable username password based authentication (ALLOW_USER_PASSWORD_AUTH) [Learn more.](#)

☒ Enable SRP (secure remote password) protocol based authentication (ALLOW_USER_SRP_AUTH) [Learn more.](#)

☒ Enable refresh token based authentication (ALLOW_REFRESH_TOKEN_AUTH) [Learn more.](#)

Prevent User Existence Errors [Learn more.](#)

☐ Legacy

☒ Enabled (Recommended)

[Set attribute read and write permissions](#)

11. After you have created an app client, go to the “Review” step. Scroll to the bottom of the page and click “create pool”.

12. Now your user pool is configured to use in an Android app!

Use AWS Cognito to Authenticate Users in an Android app

Now you are going to create a simple Android application where you will integrate AWS Cognito to authenticate users.

Here is a useful tutorial to help you along the way:

<https://aws.amazon.com/blogs/mobile/using-android-sdk-with-amazon-cognito-your-user-pools/>.

Start at **Using Amazon Cognito User Pools in your Android app** and stop after the code snippet with the AuthenticationHandler. In the tutorial, some of the methods in the Handlers are outdated, so it is easiest to import the Handlers from AWS and then allow Android Studio to implement the methods (explained more under Helpful Hints).

You are given *activity_main.xml*, *MainActivity.java*, *activity_logged_in.xml*, *LoggedInActivity.java* and *AppHelper.java* [here](#) to help get you started. **AppHelper.java is a singleton where you can set various attributes of your user pool and create a CognitoUserPool instance.** It is there mostly to help you separate out this functionality from the MainActivity.

The easiest way to start is to create a new Android studio project with an empty activity. Then copy the MainActivity.java and activity_main.xml over the ones generated for you and update the package statement in your MainActivity.java if necessary. Then create a new empty LoggedInActivity and replace it's .java and .xml files with the provided code as well. Don't forget to copy in the provided AppHelper.java file. As you follow the tutorial mentioned above, look for places in the provided code marked with TODOs where you can write the code described in the tutorial.

Add this dependency to your project:

- <https://mvnrepository.com/artifact/com.amazonaws/aws-android-sdk-cognitoidentityprovider>

Requirements:

- Should be able to sign up and sign in using AWS Cognito
- Should be able to enter a confirmation code into the Android app which should confirm the user, and the screen should switch to the logged in page


Suggestions:

- Implement the AppHelper class first by following the TODO comments listed in the file

Submit a screenshot of the **email with the verification code** and of the **Cognito console with the user verified by email**.

Helpful Hints:

- When you create a new user pool, make sure you pass in the following attributes:
 - new CognitoUserPool(context, userPoolId, clientId, clientSecret, clientConfiguration, cognitoRegion)
 - CognitoRegion must match the region where you created your userpool
- Add internet permissions in your Android Manifest file
 - <uses-permission android:name="android.permission.INTERNET" />
 - <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
- **You will need to use three Handlers for this lab** (SignUpHandler, GenericHandler, and AuthenticationHandler). Stubs for these are located in MainActivity.java. **Import them from AWS** and then let it generate the necessary methods by clicking on the red light bulb and "implement methods".

```
 //TODO: Import these handlers from AWS and then allow it to generate the necessary methods
SignUpHandler signupCallback = new SignUpHandler();
```

- If sign up isn't working, make sure that your password aligns with the password policy that was set when we created the user pool. (Requires: min length of 8, special character, uppercase letters, lowercase letters, numbers).

- To call the method for sign in, and to send back a verification code for sign up, you will first need to create this CognitoUser object.

```
//Use this to get the CognitoUser object  
CognitoUser cognitoUser = AppHelper.getInstance().getUserPool().getUser(usernameText);
```

- If you can't find the call for sign in, look on the last line of the code snippet with the AuthenticationHandler.
- For the CognitoUserAttributes part of the tutorial, you will only need to worry about given_name and email, not phone number.
- You will want to switch activities of your app in the onSuccess of your GenericHandler.
- Verify that you set the correct package name in the AppHelper.java and LoggedInActivity.java files.
- In the AWS tutorial it gives you the following code to signup a user

```
// Sign up this user  
userPool.signUpInBackground(userId, password, userAttributes, null, signupCallback);
```

- The method signature for the onSuccess method of the SignupHandler is outdated in the AWS tutorial. You are not given a 'userConfirmed' parameter. You can get the same information from the SignUpResult you are given instead. Use '!SignUpResult.isUserConfirmed()".