AMAZON COGNITO: AUTHENTICATION MANAGED BY MEANS OF SINGLE SIGN-ON

Amazon API Gateway | Multi-Factor Authentication | Single-Sign-On (SSO)

beSharp | 23 August 2019

One of the most recurring features in web and mobile applications is certainly user authentication; being able to off-load the responsibilities related to user authentication management greatly increases the resilience of the solutions implemented and the speed of development.

The use of managed services allows the development team to automate and make more resilient critical functionalities such as user authentication. Especially when creating web applications. The managed services dedicated to solving this aspect is **Amazon Cognito**.

Amazon Cognito provides a building-block that can accelerate and secure the process of authenticating and authorizing users to mobile or web applications.

Before getting into the heart of the matter it is useful to define the specific terms of the service and the preparatory concepts to understanding the topics covered.

What is Amazon Cognito?

Amazon Cognito allows you to add registration and authentication to Web and mobile applications. It allows users to authenticate either through a fully managed user pool or through an external identity provider (IdP). It can also provide temporary security credentials to access AWS resources.

Amazon Cognito is compatible with external identity providers that support the SAML or OpenID Connect standards like social identity providers such as Facebook, Twitter and Amazon. And also allows you to integrate your own identity provider.

Amazon Cognito is a distributed and public access service; this means that anyone can call the Cognito API to authenticate to a service. This is a standard and well-established manner; it is normal for an authentication service to be public and accessible in order to allow clients to carry out login operations.

The security of the solution is guaranteed by Amazon, which completely manages the service and the APIs accessible by the public.

Clients can then log in using one of the supported IdPs, or send Cognito a username and password to obtain an **identity token** and, optionally, a set of IAM credentials whose policy is controlled by Cognito through specific configurations.

Clients with an identity token can authenticate with services that use Amazon Cognito as an IdP and can use IAM keys (if provided) to call the application backend that supports IAM authentication or to directly access resources on the AWS accounts allowed by specific policies.

Amazon Cognito **integrates with API Gateway,** thus protecting the back-ends in a completely managed and automatic way.

The API Gateway back-ends protected by Cognito will not receive requests that do not pass the authentication checks, ensuring a more resilient and cost-efficient solution.

The integration between Amazon Cognito and API Gateway allows you to implement only authentication or authentication and authorization.

If you intend to develop authentication only, it will be possible to take advantage of **the integration through Cognito User Pool**, which will guarantee access to the API to any user who is correctly authenticated by credentials or through an external IdP.

If **IAM integration** is selected, authentication and authorization features can be obtained. The identity token of a User Pool will be used to obtain a pair of IAM keys to be used to make back-end calls.

It becomes possible to limit the sets of APIs accessible to different groups of users. Furthermore, it is possible to give controlled and secure access to AWS resource sub-sets through the temporary keys and the specific managed policies; for example, uploading a file to S3 or putting a message in a queue without having to interact with the back-end of the application. In fact, off-loading few critical operations with deep integration with our Service Provider is very safe and effective.

Users are managed through **two types of pools,** which are at the heart of Amazon Cognito's operation: **User Pools and Identity Pools.**

Let's move on to describe the main concepts of Cognito.

User Pools

A user pool is a user directory configurable for use with a Web and/or mobile application. A user pool allows you to securely store your users' profile attributes.

This is a convenient way to completely off-load users' management who decide to register using a username and password. Among the operations which can be off-loaded there are certainly the secure storage of user data, the verification of possible telephone numbers or e-mail addresses, the management of the API and the flow of registration, login, logout, and password reset.

User pools are a fundamental component of any authentication system based on Amazon Cognito. It is also possible to connect a user pool with an external IdP to allow service users to register and log in via Facebook, Google, Amazon or any public IdP that supports OpenID.

Identity Pools

Identity pools are containers used by Cognito Identity to keep the application's federated identities organized. An identity pool associates federated identities from external identity providers with a unique specific user identifier. Identity pools do not store user profiles, but only their unique ids generated and managed by Cognito. An identity pool can be associated with one or more applications.

Cognito Identity assigns users a set of temporary credentials with limited privileges to access AWS resources, so it's not necessary to use AWS account credentials to allow users to interact with cloud resources. Authorizations for each user are controlled through customizable AWS IAM roles. It is possible to define rules for choosing the IAM role of each user; if you use groups in a Cognito user pool, you can assign IAM roles based on those groups. Cognito Identity also allows you to define a separate IAM role with limited permissions for non-authenticated guest users. Finally, you can use unique identifiers generated by Cognito to define user access to specific resources. For example, you can create a policy for an S3 bucket that allows users to access only their own folder.

Now that we've defined all the fundamental concepts we can move on to the central part of our article, or the **tutorial to configure Amazon Cognito in order to allow users to be authenticated through their Google identity.**

Tutorial:

The first step to complete Cognito configuration is to create a User Pool that we will subsequently federate with Google and with an Identity Pool in order to allow users to identify themselves via Google and obtain an identification token and a pair of keys to access cloud resources.

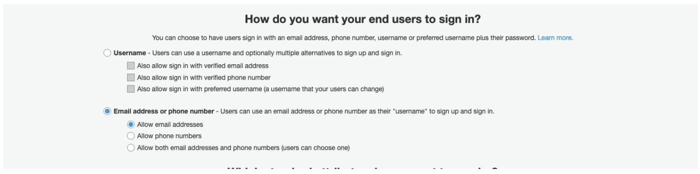
Creating a User Pool

From the Amazon Console, navigate to the Cognito dashboard and click on the "Create a User Pool" button to start the creation wizard.

First, a name for our User Pool will be requested. At this point, we can decide whether to proceed with a step by step wizard or to create a User Pool with the default options and to revise it before creation. We choose the first option by clicking the "Review Settings" button.

The first step of the wizard allows us to choose with which information users can log in on Cognito. The options are many: we can choose to authenticate the user through the use of a username, an e-mail, a phone number, etc… It is important to note that these possibilities are valid only for those users who will log in directly on Cognito (after registering) and for those who will use any external IdPs.

For this tutorial, we are going to select “E-mail address or phone number” and then “Allow e-mail addresses” (As per image). **In this way, the only possibility for users to login will be by using a valid e-mail.**



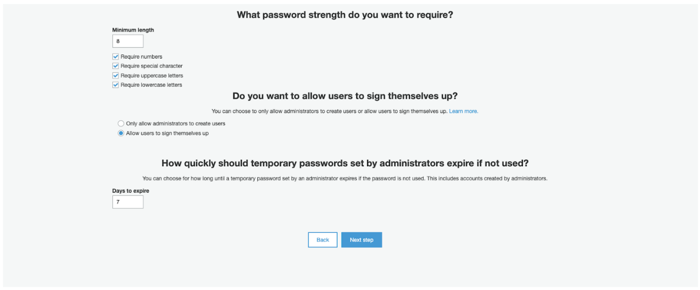
How do you want your end users to sign in?

You can choose to have users sign in with an email address, phone number, username or preferred username plus their password. Learn more.

- ○ **Username** - Users can use a username and optionally multiple alternatives to sign up and sign in.
  - ☐ Also allow sign in with verified email address
  - ☐ Also allow sign in with verified phone number
  - ☐ Also allow sign in with preferred username (a username that your users can change)
- ● **Email address or phone number** - Users can use an email address or phone number as their "username" to sign up and sign in.
  - ● Allow email addresses
  - ○ Allow phone numbers
  - ○ Allow both email addresses and phone numbers (users can choose one)

In this step, it is possible to configure the level of security to which every user will have to follow in order to create a secure password. We can insert the minimum limit of password characters and various constraints on the mandatory use of special characters, numbers, capital letters, and lowercase letters. We leave for the rest the default values and continue with the other configurations.

The second and third points of this step allow us to configure the way users are created and the maximum duration of temporary passwords generated by administrators for new users. Basically, we have two ways to create users. The first allows users themselves to be able to register through a registration page, the second instead provides an invitation by the system administrator. This means that it will be the user who owns the application, through the Amazon Console or through the CLI, to create the user with a temporary password that the user will change at first login. For our tutorial, we select the possibility of being able to register users independently (as shown in the image) and continue by clicking the “Next step” button.



What password strength do you want to require?

Minimum length: 8

- ✓ Require numbers
- ✓ Require special character
- ✓ Require uppercase letters
- ✓ Require lowercase letters

Do you want to allow users to sign themselves up?

You can choose to only allow administrators to create users or allow users to sign themselves up. Learn more.

- ○ Only allow administrators to create users
- ● Allow users to sign themselves up

How quickly should temporary passwords set by administrators expire if not used?

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.

Days to expire: 7

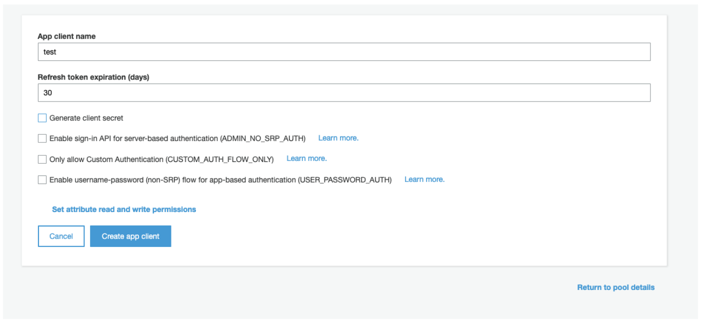
In this part of the wizard, we can **enable the Multi-Factor Authentication** and optionally the verification of the e-mail/phone during registration. For our tutorial, we can leave the MFA disabled and select “e-mail” as an attribute to verify in case of independent registration.

In the next step, instead, it is possible to configure all the automatic messages that will be sent by Cognito to the end-user. These messages include temporary passwords and verification codes that can be sent via e-mail or via SMS by configuring Amazon SNS. For our tutorial, we can leave everything at default and move onto the next step.

At this point in the wizard, we can add tags to the resource we are going to create. We add the appropriate tags or even none if it’s just an experiment and move onto the next step.

Now we are asked if we want it to automatically save the devices so as not to require the use of the MFA at every access. Since in the previous step we left the MFA disabled, we select “No” and proceed to the next step.

We are now asked to create an App Client which will then be used by our application to make the necessary registration and login calls. To do this, simply press the “add an app client” button, enter a name, remove the checkmark from“Generate client secret” and press the “Create app client” button.


App client name
test
Refresh token expiration (days)
30
☐ Generate client secret
☐ Enable sign-in API for server-based authentication (ADMIN_NO_SRP_AUTH) Learn more.
☐ Only allow Custom Authentication (CUSTOM_AUTH_FLOW_ONLY) Learn more.
☐ Enable username-password (non-SRP) flow for app-based authentication (USER_PASSWORD_AUTH) Learn more.

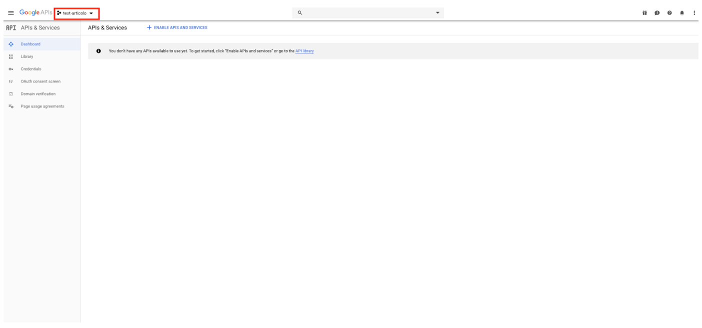
We can then finish the wizard by **creating the user pool by pressing the button on the last screen.**

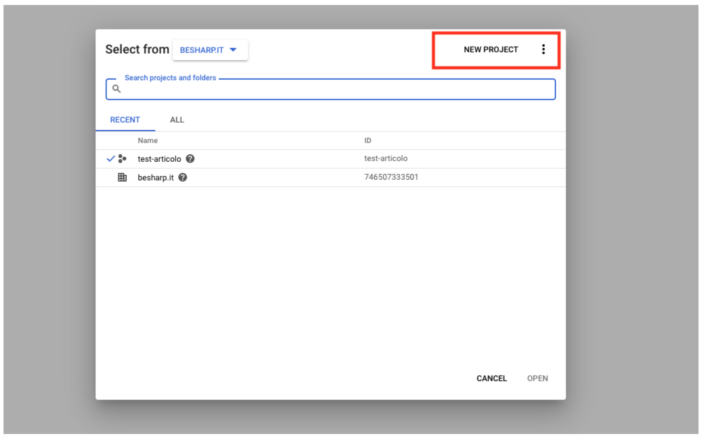
Before moving onto the integration configurations and choose a domain name for our User Pool. Once chosen, click on the “Check availability” button and then on “Save changes”.

External Identity Pool Integration Configuration

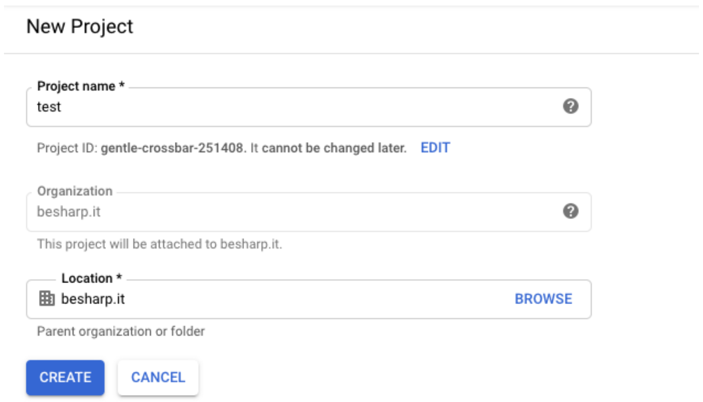
Once the user pool has been created, we can move onto integration with our SSO. In order to do so, we need to create a **new project from the Google Developer Console** and get login credentials. Once the credentials are obtained we will pass to the Cognito configuration.

Let’s go to the Google Developer Console and create a new project (as in the image).


You don't have any APIs available to use yet. To get started, click "Enable APIs and services" or go to the API library


Select from BESHARP.IT

Name	ID
✓ test-articolo	test-articolo
besharp.it	746507333501


New Project

Project name *
test

Project ID: gentle-crossbar-251408. It cannot be changed later. EDIT

Organization
besharp.it

This project will be attached to besharp.it.

Location *
besharp.it BROWSE

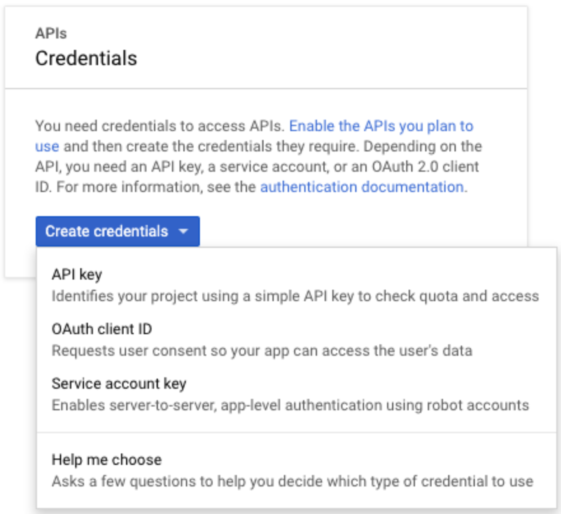
Parent organization or folder

Once we have created our project, we go to the "OAuth Consent Screen" entry.

Here we will configure the name of the application and we will enable the domain

"amazoncognito.com". To do this, just insert it in the appropriate field called "Authorized Domains" and click on the "Save" button.

Let's now go under the "Credentials" field and click on the "Create credentials" button selecting "OAuth Client ID" from the proposed drop-down menu.


APIs
Credentials

You need credentials to access APIs. Enable the APIs you plan to use and then create the credentials they require. Depending on the API, you need an API key, a service account, or an OAuth 2.0 client ID. For more information, see the authentication documentation.

API key
Identifies your project using a simple API key to check quota and access

OAuth client ID
Requests user consent so your app can access the user's data

Service account key
Enables server-to-server, app-level authentication using robot accounts

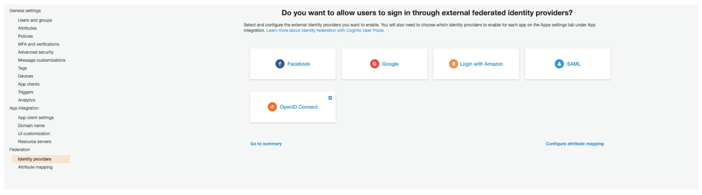
Help me choose
Asks a few questions to help you decide which type of credential to use

We then select "Web application", insert a name for our client and **as authorized javascript sources, we enter the domain name of our Cognito User Pool** previously created (eg: https://test-articolo.auth.eu-west- 1.amazoncognito.com). We then go on to enter the same domain name followed by **/oauth2/**idpresponse within the authorized redirection URI field (eg: https://test-articolo.auth.eu-west-1.amazoncognito.com/oauth2/idpresponse) and press the "Create" button.

At this point, a popup will appear containing our "Client ID" and our "Client Secret". **We copy the secrets into a text file to have them quickly available later on.**

Let's now move on to the integration of Cognito and Google starting with Amazon Cognito.

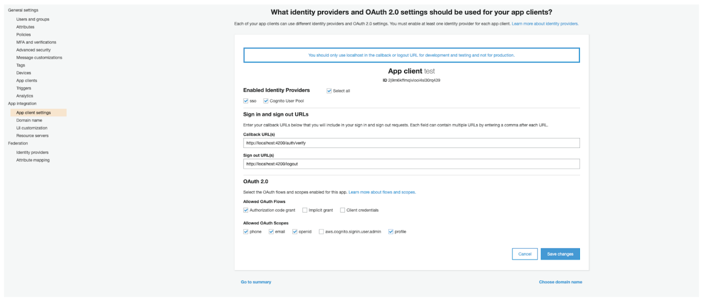
Navigate through the “Identity Provider” menu and select the type of integration we want to carry out.


Do you want to allow users to sign in through external federated identity providers?

Select and configure the external identity providers you want to enable. You will also need to choose which identity providers to enable for each app on the Apps settings tab under App integration. Learn more about identity federation with Cognito User Pools.

From this point of view, Amazon offers integration with major identity providers including Facebook, Google, Amazon and, more generally, all providers that support OpenId and SAML protocols. For this tutorial, we’re going to integrate with Google. To do this, we click on the “Google” button and we will enter the information of the identity provider with those returned to us during the creation of the Google project.

The last remaining step concerns the linking of the two previously created resources. To do this, just navigate through the “App client settings” tab. In the proposed screen, we will have to select our identity provider previously created as Identity Provider and supply the callback endpoints as in the image. The callback endpoints will be the endpoints called at the end of authentication and will contain our token.


What identity providers and OAuth 2.0 settings should be used for your app clients?

Each of your app clients can use different identity providers and OAuth 2.0 settings. You must enable at least one identity provider for each app client. Learn more about identity providers.

You should only use localhost in the callback or logout URL for development and testing and not for production.

App client test

ID 2j9m6kffmqivlooi4sl30rq439

Enabled Identity Providers ✓ Select all

✓ sso ✓ Cognito User Pool

Sign in and sign out URLs

Enter your callback URLs below that you will include in your sign in and sign out requests. Each field can contain multiple URLs by entering a comma after each URL.

Callback URL(s): http://localhost:4200/auth/verify

Sign out URL(s): http://localhost:4200/logout

OAuth 2.0

Select the OAuth flows and scopes enabled for this app. Learn more about flows and scopes.

Allowed OAuth Flows: ✓ Authorization code grant ☐ Implicit grant ☐ Client credentials

Allowed OAuth Scopes: ✓ phone ✓ email ✓ openid ☐ aws.cognito.signin.user.admin ✓ profile

To do a test we can use the Hosted UI provided by Cognito browsing to the domain we previously chose and passing our client id, the redirect uri and some additional information as an example:

https://test-articolo.auth.eu-west-1.amazoncognito.com/authorize?client_id=**{client-id}**&redirect_uri=http://localhost:4200/auth/verify&response_type=token

Once the authentication process is complete, we will automatically be redirected to our callback address http://localhost: 4200/auth/verify with the difference that we will find our access tokens in the query string parameter.

At this point, it is up to the application to retrieve the tokens from the callback call, which for the purpose of this tutorial we set to localhost, and use them for subsequent back-end calls.

With this tutorial, we have seen how to configure the integration between Amazon Cognito and Google from start to finish. We have illustrated and tested how to get the identification token and how to use it to access back-end services.

Stay tuned for more interesting tutorials!

beSharp

Dal 2011 beSharp guida le aziende italiane sul Cloud. Dalla piccola impresa alla grande multinazionale, dal manifatturiero al terziario avanzato, aiutiamo le realtà più all'avanguardia a realizzare progetti innovativi in campo IT.

Get in touch

beSharp.it
proud2becloud@besharp.it