Learning AWS for Developers – Bear Cahill LinkedIn Learning

Table of Contents

[Learning AWS for Developers – Bear Cahill LinkedIn Learning 1](#_Toc125617314)

[Introduction 2](#_Toc125617315)

[Chapter: 1 AWS Console 2](#_Toc125617316)

[AWS account and pricing 2](#_Toc125617317)

[AWS console and services 2](#_Toc125617318)

[Certified Developer-Associate 3](#_Toc125617319)

[Chapter: 2 Security 3](#_Toc125617320)

[Identity and Access Management (IAM) 4](#_Toc125617321)

[Users and groups 4](#_Toc125617322)

[Roles and policies 5](#_Toc125617323)

[Cognito 6](#_Toc125617324)

[Identity pools 6](#_Toc125617325)

[Identity provider (IdP) 7](#_Toc125617326)

[User pools 7](#_Toc125617327)

[Chapter: 3 Development 8](#_Toc125617328)

[SDK and tools 8](#_Toc125617329)

Learning AWS for Developers – Bear Cahill LinkedIn Learning

Introduction

Amazon Web Services (AWS) is the leading cloud service provider today, ahead of competitors such as Microsoft Azure and Google Cloud Platform. It is seen by many as the standard for functional, reliable, interoperable, and affordable cloud services. Developers need to understand what it means to develop on and migrate to the cloud—and comprehend the overall landscape of AWS before diving into the platform. This course provides a high-level overview just for developers, focusing on the AWS features and services of the most interest to them. Instructor Bear Cahill introduces AWS Lambda, DynamoDB, ElastiCache, Elastic Beanstalk, S3, SQS, SNS, and many more key features and services.

Chapter: 1 AWS Console

# AWS account and pricing

Of course, one of the first things we should talk about when developing on AWS is getting an AWS developer account. So if you go to aws.amazon.com, you see the typical start page with a lot of information and panes and tiles and links, and it's a lot of really great information. So what I like to do is show people the aws.amazon.com/free site. Free is good. And here it encourage you to create a free account. Now, what it means is that a lot of the services are free. It also means the account's free. When you sign up for account, you do have to give them a credit card number in case you go over any of the limits or use services that cost. But, we can filter the different free services by featured or 12 months or always free. And you get a really good sense of what services are free and what kind of limitations are on them. So if we click the always free over here, we see that you can always use 25 gig in DynamoDB and it'll be free. You can always have a million requests to your Lambdas in a month and it'll be free. So the limits for the cost are very high. It's very generous, in my opinion. And so you can get a real good sense of what costs here. Everything we do in this course should be free. If anything does cost, you can always go to the billing feature, which we'll look at later and see what the cost is going to be estimated at and then turn it off. So you can click this create an AWS account up in the top right, and it'll walk you through the typical process for signing up for an account. Again, you have to give them the credit card number in case you do go over any of the limitations for charges. In my experience, even if you do, it's going to be very little, so not too much concern there, but always try to be aware of what does cost. So, sign up for the account and then we'll get started.

# AWS console and services

So once you have your AWS developer account, you're going to want to start being an AWS developer, right? Of course. So a great place to start is the console. I go to console.aws.amazon.com and it loads up the Console Home. So this is a great place to see a lot of the services that are available. Your current costs, different tutorials for solutions and so forth. And depending on your configuration it might show more or less based on font size. You can move these widgets around. You can completely remove these widgets if you want and so forth. You can see the different services you've visited recently, if you've visited any. And you can see your favorites up here on the top left. If you click on the Services button right above there you can see different groupings or organization of the services. You're recently visited, your favorites. All the services alphabetically, and then the kind of conceptual grouping of them. So we could look at something like Analytics or Blockchain or Compute. Now I use Lambda a lot. That's under Compute. It's a great way to be able to build serverless functionality and languages like Python and Node js. You also have things like databases where you have DynamoDB, which is a NoSQL Database. You also have RDS, which is a relational database service. So these are grouped under database. Same with developer tools, for tools for committing, deploying, building code and the whole pipeline. Things for front-end, web and mobile. So things like Amplify that ties a lot of services together for development using web or mobile platforms. And AppSync for creating a GraphQL API. I love using that service. It has a great offline feature for that. And then of course, things like security where you have to use authentication and authorization for allowing services to call each other. IAM is kind of the whole clearing house for all the different authorization and policies. And Cognito is a great service for allowing users to log in either with existing accounts or creating new accounts just for your app. And then there's storage. You may have heard of S3 buckets, great place for allowing users to store files or just the app storing files that need to serve files to users or something during processing, whatever it is. It's a great place for file storage. So that's a bit about the organization. When you come to the console you can kind of get a sense of this is your home field. Over time, you may use this more or less, depending on how you get familiar with AWS, but it's a great starting place. One other thing I want to point out is depending on your account, you should be assigned kind of a region, a default region where you're going to do your development. And a lot of AWS services work in different regions. Some services only work in one region and some services work across regions. So sometimes you need to be a little bit careful about where you define like a Lambda versus the DynamoDB it's going to be hitting. You want those probably to be in the same region. In other cases, it doesn't matter. Now, when you come to the console, it should use your default region, right there in the URL it's specified. And also from this dropdown on the top right by your account name. And all the links that it should lead you to from the console should also use that same region. That's another good reason to try to use the console as kind of your home base for AWS development. So anyway, now that we're kind of getting familiar with this, let's take one more step in seeing what the future could be for your AWS experience.

# Certified Developer-Associate

Once you become pretty skilled at being an AWS developer, you may want to be able to have some quantifiable proof of what you know. And one way to do that is with AWS certification. You can go to aws.amazon.com/certification and get a lot of information on what's required to become certified and how to do it. There's a variety of different certifications, and if you look down here under the associate certifications there's one for development. Now we're going to be talking about development in this course, so I'll click on that one and we can see some details. So it's saying who should take this exam. So this can give you an idea of what's required. So, some high-level programming language experience, application life cycle management, the developing, deploying, and debugging of cloud-based applications, uses of tools like APIs, the command line interface, SDKs, and so on. So you can get a sense of what kind of developer they're talking about here. You can also see some details on the exam. What does it look like? And you can download things like the exam guide and sample questions and a lot of material on preparing for the certification. So hopefully this course and other courses we provide will allow you to understand what's required and build a strong foundation in case you decide to go the direction of getting certified as an AWS developer. We'll be looking at a lot of the services that are required for this test, so hopefully this will be a great start for you if you decide to go that route.

Chapter: 2 Security

# Identity and Access Management (IAM)

As with any platform where you have an account, you should always consider security. We're going to talk about security first using IAM. You can get there from the recently visited if it's listed, your favorites or from the services drop down. I'm going to scroll down to security, identity and compliance and then find IAM and click on that. IAM stands for Identity and Access Management or IIAM or IAM. Now, there's two primary concepts I want to talk about here. One is users. So as you might expect, you can create various user accounts, give them different user names and so forth and then you want to give them permissions. To do that, you attach policies to the specific account. Notice that this root account has one policy which is administrator access, which of course gives them permission for everything. If we look at the policy summary here, we'll see that for every service they have full access on all resources. We can look at the JSON. We see it uses wildcards to allow everything on everything. Now, if you were creating another user account, you probably wouldn't want to give them all those permissions. So instead, when you clicked add permissions or during the account creation, you might attach existing policies directly and do a search for something like DynamoDB and give them access to that. You can also do it through user groups which we'll talk about in the next video or you can copy permissions from an existing user. So users get permissions through policies and there are a lot of existing policies. You can even create your own policies. There's over 900 policies here that are defined in AWS and you can create your own. User groups allows you to create a group and attach policies to it. For example, if you have a tester group and then whenever you create a tester account, you can just assign them to the group and if necessary assign them other policies, but they would have a collection of these policies that you could change just in one place. Roles is the other big concept. This allows you to assign policies to a role that a service can adopt when it's running. So if you have a Lambda that needs to access DynamoDB and SNS, you could create a role with those two and assign it to the Lambda. Then the Lambda would have permission for that. All of your Lambdas may have the same role or they may use different roles. You can also use identity providers in AWS to allow users to log in with that and you can specify various account settings for your accounts that get created for things like changing the password. We're going to go into some more details of users and user groups in the next few videos. So hopefully it'll become more and more clear.

# Users and groups

So now we're going to look at a couple of things that I mentioned in the previous video, users and user groups. So all of these items in here relate to each other directly and indirectly. So you can kind of start at any point and get to the other points. So for users, you should have a root user, that might be who you're logged in as. It's good practice to go ahead and create another user that has the same privileges, but they still won't be a root user from the standpoint of being able to actually delete the entire account. So let's look at creating a user. I'm going to click on Add users. I'm going to give it the username adminuser. And you can get an access key, which will allow you to programmatically log in for things like the API, command line interface, SDKs, and so forth. Or you can do a password login. This will allow the user to log in through the console like we are here. You can customize the password or autogenerate it. You can force the user to create a new password when they log in as well. So fairly basic stuff. And then you can add the user to a group, and that way they get their permissions as part of the group definition. You can copy permissions from an existing user, like the root user has administrator access. Or you can attach existing policies directly. When you do this, you can select all the policies you want. You could say I want to give them all the DynamoDB access or Lambda access or whatever it is. In this case, I'm actually going to use the administrator access. So even though this is not a root user from the standpoint of being able to delete the account, they will have all the permissions. I'm not going to add any tags. We can review this, and I can create the user. At this point, we get the access key that we can copy. We can see the secret access key and the password. No, I'm not going to show those to you. And you can email this information to a user. Now this is the only time you'll be able to access this key and this password, so you should copy it here. Otherwise it's gone. So I'm going to just close this for now. We've created this user, and we can go look at this user, and we can see the permissions, and we could add additional permissions to this. We could also add them to a group. I don't have any groups defined, so let's do that. So I'm going to click on User groups over on the left and create a new group, and it's going to be fairly similar, and I'm going to call it my TestGroup. And I could add different users to that. Maybe I'll add just the adminuser. And then I could add additional policies here if I want to. And then create the group. And now I've got this group. I could click on it and add additional users, or I can go to the users themselves. And I could click on the Groups tab under the users and add them to a group. This is just a handy way to be able to manage policies as a group. And so if I wanted to change all of the permissions for everyone in the TestGroup, I can just do it in that one place, very handy. So that's just a quick look at users and user groups, how they relate to each other. Users can be in the groups. Users have policies, groups have policies. Users in groups, they can have policies that way as well. So just different ways to be able to manage users and their policies, which is all the permissions for when they log in through the console, or the command line interface, API, SDK, that sort of thing.

# Roles and policies

In the previous video, we talked about users and user groups, and how they use policies to give them permissions into other services. Now we're going to talk about roles. Roles are adopted by services that don't necessarily have a user associated with them. They're just going to be run either periodically or when they're hit from an API, whatever it is, this service is going to run and it needs some permissions. A real good example is a Lambda. Lambdas often need functionality into other services, like an S3 bucket or a DynamoDB table. So I'm going to create a role to kind of look at how that works. So the first thing I want to do is I'm going to use AWS S3 service. And I'm going to pick Lambda right here. So my Lambda that's going to use this role needs some permissions. So I'm going to click Next, and give it those permissions. Let's say it needs to access a DynamoDB table. So I'm going to use a filter of dyna. And I can give it, for example, the LambdaDynamoDBExecutionRole. Depending on what you're trying to do, you may want to look down into the details and see everything that's provided by these policies. But as an example, I'm just going to use the LambdaDynamoDBExecutionRole. I'll click Next here. I'm going to give my role a name of MyLambdaRole. It already has a description. I can see the JSON. I see the permissions. I'm going to create the role. Now I have a role that any Lambda I write that needs those permissions can adopt. So when you create a Lambda, it's going to say what role do you want this to run as? And I've made that collection of policies with this Lambda role. You can also create specific policies if for some reason there isn't one that quite satisfies what you want. There's hundreds of them, so there probably is but maybe you want to limit to just a certain DynamoDB table or a certain S3 bucket. In that case, you can create a policy. So you pick the service that you're going to limit for, for example, DynamoDB. And then you select what actions you want to allow. Maybe we want to allow list and read but will not allow the user of this policy to write. Then you can specify specific resources, specific DynamoDB tables to limit each of these actions. To do that, you would add an ARN, an Amazon Resource Name. You would get this by the table itself. So you'd go to the DynamoDB tables console or if it's an S3 bucket, you'd go to S3, and you would get the ARN for that specific table or you can allow it on all resources. So we're basically creating a kind of list/read-only policy here. We're going to click Next for tags. Then we're going to review it. And here we give it the name. So we'll do it like a ReadOnlyDynamoDB policy. List and read commands. And we could look at the details here and then create the policy. And now if we ever want to create a role or a user that only has these permissions, we can assign this policy to that user or that role or again, as the user group. So you do have a lot of power and I am for creating users, user groups, roles and policies. In many cases, you can use the provided policies but occasionally, you might want to create your own.

# Cognito

Continuing our discussion on how users access services and features through your app or website, we're going to talk about Cognito. So Cognito offers two types of pools, user pools and identity pools. User pools are user directories that provide sign-up and sign-in options for your app users. So basically users that create an account through your app or your website using AWS. Identity pools provide AWS credentials to grant your users access to others AWS services without possibly a login directly through your app. So maybe through a third party or maybe through a custom login that you already have on the server. So user pools are created using the settings and options that allow users to log in. So you can specify if they have to have a username and email address, the different criteria for their password, and also MFA and anything else that you would require during login. Identity pools also allow users to log in, or in some cases not, to have access to AWS services. Again, you might allow users to access files in AWS S3 buckets without actually even having an account. For example, let's say you store your background image of your app in an AWS S3 bucket. Well, you want the user to be able to access that, so you give them credentials through an identity pool. This can also facilitate users with accounts on other systems, so for example, if you already have a login system, but you want to add AWS Services to your app or website. And we'll look at those in a little bit more detail next.

# Identity pools

Now that we've talked about roles and their policies, we want to talk about the very common scenario of having a case where users are logging into your app or website, and then they're going to need permissions. Once they're authenticated, they need authorization to do something, typically to hit an API or to hit a Lambda, something like that. And for that, we're going to use Cognito. So I'm going to go back down to the services of Security, Identity, & Compliance and click on Cognito. Now I mentioned the two types of pools, user pools and identity pools. User pools helps the system authenticate a user. It's the identity pool that gives them authorization. So I'm going to click Identity Pools, and we're going to create a new one. So the identity pool, we'll just call it MyIDPool. And then you have to decide if you want to allow unauthenticated entities. So this is kind of interesting. This is the case where maybe you have some API endpoints that you want anybody to be able to use. Maybe if your app fetches images from an S3 bucket, you do want to allow the user to be able to get some images. They just can't write any images, something like that. Or maybe they can call certain Lambda endpoints but not other ones. So you might want to allow some unauthenticated entities. Otherwise you pretty much just want the classic flow, but we're not going to use the multi-step. And for authentication providers, you have a variety of kind of common options here. One of them being Cognito itself. You can use Cognito to use a user pool, and that'll do the authentication. Or maybe you want to allow users to log in with an Amazon account or Apple or something, maybe multiple ones. You can add additional providers there. And then you create the pool. Now we need to allow some aspect of what role these users are going to have when they're authenticated. So this is going to create new IAM roles for us. We can look at the authenticated role and what do they have permission to do. So analytics and some Cognito stuff. Really, almost nothing. We can change those later, or if we'd already defined roles, we could use our defined roles. The unauthenticated role only allows for puts on the analytics and a little bit of Cognito sync. So less, again, we could edit these and add other permissions that we need, for example, calling an API endpoint or a Lambda or whatever we need. But once we allow these, this should create those permissions. So I'm going to go back to IAM, and I'm going to look for those roles, and there they are. And as necessary, I could edit these and add additional services to them to allow logged in users to either call an API endpoint or whatever permissions they need. And unauthenticated users to be able to call just certain Lambdas or certain API endpoints. And that would be how I control it. Now how does a user get into the identity pool? How do they get that role? Well, that is the user pool or some other identifying authorization provider. And we're going to look at that in the next video.

# Identity provider (IdP)

So now we have an identity pool. That means for any authorized or unauthorized user, it assigns them that role that we specified depending on their authorization. And then they have the permissions related to that role. So how do they get into the identity pool? That's done by the authentication providers. So we specify how they get authenticated into this identity pool. Now, a very common one is to use Cognito itself. That's where the user pool comes in. But you can use some other services like Amazon, where you put in the app ID or Apple, Facebook. All of these services have different routes, but typically you would go there, create the Facebook app or the Google app, and you would get the client ID and follow the steps for creating that. Open ID and SAML typically are for existing server logins and you would have to facilitate that authentication through the steps in IAM. We're going to use Cognito. So we have to create a user pool to allow users to create their account and log in and then assume the authenticated role. So let's look next at how to create a user pool.

# User pools

Since we're going to use Cognito as our authentication provider, we can stay within Cognito. There's a couple of ways we can get to the user pool section. We can either go back to Cognito or we can click on this federated identities and then select user pools. We don't have a user pool yet. We want to create one and then use it in our identity pool as the authorization provider. So I'm going to click create a user pool, and I'm going to give it a name, MyPool. And then we can either step through all the settings or just review the defaults. We're just going to review the defaults. We'll see that they log in with their email, there can be some other attributes we set up, the requirements for their password, the from email address and how to deliver the verification email, a multifactor authentication, so forth. And then we're going to create the pool. This triggers is actually pretty handy if you wanted to run a Lambda at any point during the process, creating or authenticating a new user. So I'll click create pool. And here I have a new pool, and here's the ARN and the pool ID. I probably want to copy those to put into the identity pool. But for now, I want to show you that once you have your pool created, you can look at all the users that are created and see if they verified their account, you can look at the attributes for how the user's account is stored, if they have the username, email address, and any other attributes that you want to collect, the various policies for the password, MFA, security, you can remember the user's devices when they log in so you know if they're using the same device as a previous login. If you're going to use this in an app or a website, you can add an app client and that'll give you some details to put into your app or website for accessing this pool through the web. Triggers, as I mentioned, you can run Lambdas at different portions of the process, analytics, and again, your app client settings in here to be able to specify how to integrate with various apps and websites, and so forth. So let's go back to the general settings and I'm going to select the pool ID and copy that. And then I'm going to open Cognito in another tab. And now, in that tab, I want to go back to my identity pool and set as the authentication provider our new user pool. So I'm going to select on my identity pool, and then I'm going to click on edit identity pool. And down under the authentication providers, the pool ID, I'll paste what I copied. And then for the app ARN, I want to, I want to create an app client. Here, I can name the client whatever I want, like MyMobileApp. And I can specify the refresh token, access token, and other basic settings. I'm going to use the default. Create this app client. And I'm looking for this app client ID that was just created, and I can select that and put that into my client ID. And once I save these settings, I have specified that my user pool can be used as an authentication provider into my identity pool. So now, when I use these details in my mobile app, in this case, it will be used as an authentication provider into my identity pool, and then the users logged in will adopt the role assigned with the authorized user. And I could create another app client for my website or a different mobile app. It depends on how you want to organize it. But I would recommend having something different in each case, so that if you need to make changes to one, you don't change them all. But now, I have a user pool as my authentication provider, and anyone creating an account and logging in using that account in the user pool will go to the identity pool and be assigned the role that can then hit the API endpoints or whatever else I have as functionality in my app.

Chapter: 3 Development

# SDK and tools

When you start developing on AWS, you're not out on your own in the wild west. They give you a lot of tools and help online to give you a leg up, get you a head start. And these are the kinds of sites I love to see when I'm going to develop on a platform. So here I am at aws.amazon.com/tools. And again, this is the kind of website that's like just a toy chest to me. I love seeing websites like this. So for all these different languages, depending on what you're going to develop in, if it's something on the web or even local, you can get help for building applications, working with your various IDEs, some tutorials, community, a lot of resources out there to help you really get started and keep going strong. They also provide some really cool tools. Now, when I see command line tools, I'll be honest, I get a little intimidated. I like having an interface, even though I write code and I don't design interfaces. I like using them. But these command line interfaces are really helpful and pretty easy to use. The AWS CLI, you'll see a lot of commands when you're working and going through tutorials of how to do things with AWS through the command line and the Amplify CLI's really helpful when you're developing locally for a web app or a mobile app. It can add features and services to your app locally and sync it automatically with AWS and your project online. You have a lot of other sections for tools and SDKs. Some of the ones that I really like are the mobile SDKs. So I use this iOS one a lot. You also have the front end web and mobile tools. Here's Amplify again. This is not the CLI. You can also do Amplify through the console or the CLI and services like CloudWatch where you can access this from a lot of places, so this isn't necessarily anything unique. It's just grouping these things for you. And of course the various SDKs for your languages. So as we've seen with the listing of services and all these other menus, AWS loves listing and grouping and providing this type of information in a very helpful way. Here's another site that is very similar. So this is AWS.amazon.com/getting-started/tools-sdk. So you'll notice it's very similar. So you have the SDKs up here. You have the web and mobile SDKs right here, internet of things, the IDE. So you can code in your own IDE, even the online IDE. The command line tools again, the developer tools. So this can really help your entire pipeline. So when you commit code, it can build and deploy for you. You can even use the online IDE and have your entire environment in AWS and things like the forums and the developer center. So again, resources for you. So there's a lot of things out there for you, and sometimes you'll go to one website and you'll think, "Well, is this different from that one?" Sometimes it is, sometimes it isn't, but I highly encourage you to explore all of these different things, at least to get familiar with what's out there, especially if you're going to be going for a certification for AWS.