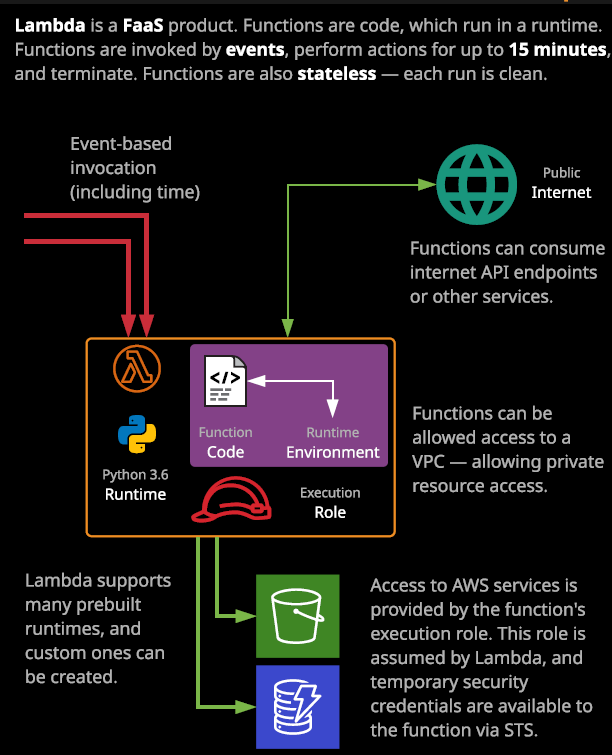
**Lambda**



Lambda is a key part of any event driven or serverless architecture inside AWS. **It's a FaaS or function as a service product.** With Lambda, you would define a function, and you define a function using some code and that some code that performs a certain task. **Once you've got the Lambda function defined with the code, the Lambda function can be invoked when it's invoked, it runs for a certain amount of time, and then it terminates and you're only billed as a user of Lambda for the time that the function is executing and that's a key benefit of FaaS in general and Lambda specifically.** You don't need to be running the compute service constantly like EC2. With EC2 you're going to pay that per second rate, regardless of whether you're using any compute on that instance. **With Lambda you're truly only being billed whenever your function is running.** Now, a Lambda function can run for up to 15 minutes, and that's a key value to remember for the exam because it influences if you should or should not use Lambda. For anything that's running for less than 15 minutes, and where you can easily access the code, it performs one or a small number of tasks, and you can define a function around that code then Lambda is ideal. For anything that runs for longer than 15 minutes, unless you can use step functions, then Lambda probably isn't suitable.

**If whatever your running needs a fully fledged operating system such as Microsoft Windows, then Lambda is not the right tool. Lambda is a function as a service product.** You give it code, you make a function, and that function could be executed and interact with other AWS and external services. Now Lambda functions are invoked either **manually or using event based invocation.** That can include invocation based on date and time. Events though are generally something which is generated by another service, and these could be used to invoke a Lambda function. Now, examples of these event based indications might be an object being uploaded to an S3 bucket or an EC2 instance being terminated, stopped, or started. **In all of these cases, whatever service generates the event it's going to pass some data to Lambda containing the details of that event and that's critical to understand many services inside AWS can invoke a Lambda function directly.**

In this lesson, we're going to demonstrate this by creating a simple serverless workflow. The scenario we're going to use in this lesson is that when you upload images to an S3 bucket, you want to generate smaller thumbnails of those images and put them in a second S3 bucket to use for a web application. So I'm going to be stepping through all of the functionality that Lambda provides as well as its key components but I'm going to do so by using a demonstration. I'm going to need two S3 buckets for this demonstration. So before I get started with Lambda, I'm going to create these S3 buckets, so I'll open the S3 console, go to create bucket, and I'm going to call my bucket AC, which are my initials and then animal pics, because it's going to be an animal picture themed lesson and the first bucket is going to be the upload bucket so I'm going to call it ac-animalpics-upload. Once I've used that I'll click on next and accept all of the rest of the defaults. Once the first one is created I'll be creating an almost identical bucket, but instead of it being the upload bucket, I'm going to call it thumbs for thumbnails. So ac-animalpics-thumbs and just as with the last bucket, I'll accept the rest of the defaults move to the end and create bucket. We've got a second bucket created. That's two S3 buckets, one where I'll upload the images to and the second and one where the thumbnails will be generated. So now that we've got the supporting infrastructure in place, I'm going to go to services and open the Lambda console which is where I'll create a Lambda function. I'll hit, Create Function and you've got a few basic options when creating a Lambda function. I can author the function from scratch, which is what I'll be using and there are also options to create one from a blueprint which I would encourage you to have an experiment with. There are some really good examples AWS have created which demonstrate the functionality of Lambda, and you can use these blueprints to give you a good starting point for creating your own Lambda functions as you get more confident. There's also the service app repository which I'll be covering a lot later in the course but for this demonstration, I've already got the code prepared, so I'll author it from scratch.

**Every Lambda function needs a unique name in the region in your account, those things are used to create the ARN or Amazon resource name, so it needs to be a unique name in the region in your account**. So for this demonstration I'll call mine image thumbnails. **Every Lambda function has what are known as runtimes.** So if you click on this diagram of illustrated, this one is the Python 3.6 runtime. **Your code, which our Lambda function uses to operate, needs a runtime. It's what actually performs the instructions in your code. So if you run a Python 3.6 or 3.7 script on your local machine, you're actually using a Python 3.6 or 3.7 runtime.** It's the thing that takes the instructions in your code and actually performs the tasks. Now, the demonstration code I'm going to be using it's a Python script. So I'll need to pick a Python runtime. Now, with Python, the first number is always the major version. So the current major version is three and the second number is the minor version. So in this particular case, I know that my code works fine with Python 3.7. So I'm going to select that.

**Now a Lambda function when it executes, it needs permissions to interact with other AWS resources**. **At a base level, it will need permissions to be able to log in to CloudWatch logs.** So that's the base level permissions that every Lambda function should also have. If your Lambda function interacts with any other AWS resources you also need to make sure that you give it the permissions to do so.

Remember early on the course when I talked about EC2 instance roles these are IAM roles which could be assumed by EC2 to give your applications on the instance AWS permissions. **Well Lambda uses execution roles. When Lambda function is invoked and executed, it has access to permissions based on that role.** So for now, I'm going to expand this permissions box and I'm going to create a new role with basic Lambda permissions. This gives it the bare minimum permissions required to operate as a function. So I'm going to select that once I have that selected that I'm going to go ahead and click on Create Function and this will create a basic Lambda function. Now, because I am offering it from scratch, it will contain some really basic example code. So I'll wait for this to complete. So this is about the most basic Lambda function that you can create. It's a Lambda equivalent of Hello World. In this case, the output is a simple message Hello from Lambda but we are able to test it. If you click on test we'll be asked to create a test event. So remember that **a Lambda function is generally invoked based on an event being generated by another AWS service.** **What a test event allows you to do is define some skeleton data or test data that you can deliver to a Lambda function while you're developing that function rather than having it actually being invoked by real events from real AWS services, you can utilize these test events to make development quicker.** So I'm going to go ahead and just call mine test event. I'll create this test event and once it's created I'll test my Lambda function, and we'll see that the execution result was succeeded. If expand the details, we can see the output. So it's essentially just run Hello from Lambda. That's the most simple Lambda function that you can actually create. Now, the reason why I wanted to test this Lambda function was to highlight a number of important things.

**The first is the duration**. So this Lambda function was executing for a total duration of 17.91 milliseconds. Now, for Lambda functions, the minimum billed duration is always going to be **100 milliseconds.** So if you consume less than that as we did, you going to be billed for 100 milliseconds. Now, with Lambda, **you actually get 3.2 million seconds of execution included free as part of the free tier so you could run a single Lambda function constantly for a month and still be under that limit.** So it's very hard to actually get any bills for a Lambda function unless you're running large scale enterprise deployments.

Now, the second thing that I wanted to cover is this **resources configured**. It shows 128 MG. Now, not all Lambda functions have to run with same amount of resources. If I scroll down, you're able to give your Lambda function anywhere from 128 MG of memory all the way up to 3008 megabytes of memory and the more memory you give it, obviously the better performance it's going to be.

I also want to highlight this **timeout value**, and this is the maximum duration that your Lambda function will execute before terminating. There is a **hard limit of 15 minutes** and no Lambda function can currently go above this 15 minutes, but you are able to set a lower value than that and I'm going to do that. I'm going to set mine just for this demo to five minutes zero seconds. Just make it a nice round five minutes and then we're good to go already briefly mentioned test events when I tested this function initially, I passed in some event data.

Well **event data is generated by any other AWS services that invoke Lambda functions** and so, with this example, where we're using S3, it's going to be S3 that generates the event, and it will pass data into the Lambda function that contains details of exactly what's happening. So for this demonstration, it will be an S3 object being uploaded to a bucket and so the event data will contain the object details. So which buckets it's been uploaded to, what is the name of the object and various other bits and pieces of information about the exact event and, of course, that differs depending on what event it is, whether it's an EC2 instance starting or stopping or whether it's an S3 bucket where an object is being uploaded when you're developing a Lambda function, you need to keep that in mind because every event will be different and it will contain different data.

Okay, so now that I've got this Lambda function created I have selected a **runtime**, it's got an **execution** **role** and it's got this **sample** **code** and the next step is to upload the **real code** that will perform the thumbnails on these images. Now they're a couple of ways that you can upload code into a Lambda function. The simplest is to **edit the code in line**, and this is suitable for any simple functions which don't use any additional libraries. This is the important thing to understand, so there's a concept known as a **runtime environment**. So when you pick a specific runtime, so Python 3.6, when you add some code, that code is running inside a temporary environment. **That temporary environment has got some space where the Lambda function can store some files, you can actually use it to install your own library**. So if you've got any specific functionality that you need inside your function, you can add bespoke libraries. Now you can't do that from this inline code editor perspective. So if you use any libraries that are outside of the standard set included in the standard runtimes then the only option you've got is to **package them up into a zip file and deliver those to Lambda** and you could do that in two ways. You can upload a zip file or you can upload a file from Amazon S3. For this demonstration, I'm going to show you how we can create a zip file containing our Python code on the libraries that we'll need to use and then I'll upload it to Lambda we'll use that as the Lambda function. Now the files that you'll need for this lesson are in the GitHub repository for this lesson. Specifically, we've got two files Lambda\_handler py which is the code that will be generating the thumbnails and I'll talk through this in a second and an execution role, which is the JSON that we'll be using to define the permissions that the Lambda function has and I'll show you how to use both of these as we go through this lesson but this is the actual code that we'll be using throughout this lesson and essentially what it does is **it's provided with a destination bucket so the S3 bucket that will be used to store the thumbnails. It receives the event from S3. So remember when you upload an object to S3 an event is generated and then it gets this event and extracts the bucket where the image has been uploaded to, the key which is the name of the object that's been uploaded, and then it generates an object name for the resultant thumbnail and it makes that by prepending thumb onto the beginning of the object name that was uploaded. Once it's got the object name to use the thumbnail then it uses a Python library to generate the thumbnail and uploads that to the thumbnail S3 bucket.**