mkdir /tmp/lambdafunction

cp lambda\_handler.py /tmp/lambdafunction

cd /tmp/lambdafunction

wget <https://files.pythonhosted.org/packages/ae/2a/0a0ab2833e5270664fb5fae590717f867ac6319b124160c09f1d3291de28/Pillow-5.4.1-cp37-cp37m-manylinux1_x86_64.whl> //downloading the python library file

unzip Pillow-5.4.1-cp37-cp37m-manylinux1\_x86\_64.whl //unzipping the downloaded file

rm -rf Pillow-5.4.1.dist-info //just to clear screen

zip -r9 lambda.zip PIL lambda\_handler.py //zipping both the files code and the library

So now that I've stepped through this code at a high level, the next step is to make the zip file that we'll be uploading to Lambda is it file will contain this code, and the associated Python libraries that are required to generate the thumbnail. Don't worry if you don't understand this code. This is a solutions architect associate course you don't need any developments experience. I'm just using this to illustrate how to use Lambda. So I'm going to move across to my terminal and I've already gone ahead and made a folder inside my temporary folder called Lambda Function and I have copied in the Lambda\_function.py file from the lesson files for this lesson. Now this lambda function is going to use a Python library called PIL the Python imaging library and step one be to download that library into this local folder because we'll be including it in the zip file. I'll make sure that all of these commands are available within the lesson description. So if you do want to follow along with this, you'll be able to but step one is to download the PIL library. So I'll do that first and then what's have downloaded it, I'll need to unzip it. So use this command to unzip the downloaded file also into this folder. I'll clear the screen just to make it easier to track what I'm doing. So I've got the file that I downloaded and the extracted version of that file in this folder. So now we need to create the zip file to upload to Lambda. So I'm going to do that by running a zip and then -r9, I'm going to specify the name of the ship that I'm going to create and then inside that zip, I want to include this PIL folder together with lambda\_function.py. So I create the zip file, and then I'm going to move the zip file to my downloads folder that'll just make it easier to upload to Lambda. So I've moved that. I'm going to switch back over to the console. I'm going to select upload a zip file. Click on "Upload." Locate this file in my downloads folder and then click "Save." So now this is refreshed. You'll see that not only do I have the Lambda\_function file, but I also have this third party library that'll need and whenever a Lambda function executes it'll extract this into the runtime environment, and that's what Lambda will use to execute this function. So not only the Python function code but any libraries that are included in this zip file. So remember for the exam, if you ever need to use a Lambda function with additional libraries, then you should use the upload zip file method either zip file or a file from S3.

Now, if you recall, I talked about how we need to specify the **destination bucket so where the thumbnails are going to be stored.** Now this uses os.environs. So this is an environment variable and a Lambda function allows us to specify environment variables are injected into this runtime environment, and that's what we need to do. So we need to define an environment variable. Now the environment variable the code is expecting is one that's called **dest\_bucket**. So the key is going to be a dest\_bucket and the value needs to be the name of the S3 bucket that I've created where my thumbnails will be stored. If you remember i called this ac-animalpics-thumbs. So I'm going to enter that, and that will mean that that gets passed in and added as an environment variable. So the code is able to refer to that. So that's the update that we needed to do so I'm going to click on "Save," and that now means whenever this function executes and it gets given an event from the source bucket here, it will be able to take that event, resize, generate some thumbnails, and store those in the thumbnail S3 bucket but it won't be able to do that quite yet.

If you remember, when I created this Lambda function I created it with an **execution role** that contained the bare minimum permissions required just to operate. What I need to do now is to update this with a new JSON document that gives it the extra permissions that I'll need. Now if I scroll to my code editor and then open up the execution role.json, which is also available inside this lesson's files. This is the JSON document that I'll be using to give the Lambda function the permissions it needs essentially it **provides access to write logs into CloudWatch logs so it can log any diagnostic messages but also allowing it to get objects from the upload bucket and to put objects into the thumbnail bucket**. So I'll copy that into a clipboard. I'll move back across the AWS console, and I'll select this link below existing role because this will open up the IAM console with this execution role pre opened just a quick way of getting where I need to go.

Now this role, **so the execution role for this Lambda function uses a managed policy**. Remember early in the course, when I talked about managed policies being their own entities? Well, I'm going to need to edit that managed policy, so I'll click on it select "Edit Policy." Go to JSON. I'm going to replace all of this auto generated permissions that AWS added with the file I was just stepping you through. The only thing I'll need to do, though, is replace these placeholders with the correct names of the S3 buckets. So this first one is get object. So I need to give it permissions to get any objects from the upload bucket. So when any objects are uploaded to the upload bucket the Lambda function gets invoked and because we're generating thumbnails it will need to read those from that buckets only to give it that bucket name. So that's ac-animalpics-upload and it will also need the ability to put objects into the thumbnail bucket. So I'm going to highlight this place holder and again change it to ac-animalpics-thumbs and that's the thumbnail bucket. Once I've done that, I'll review the policy. Assuming it checks out, I'll save the changes. Okay, so that looks good the next step now that we've got this Lambda function configured and we've got the execution role so that it's given the ability to interact with the AWS resources that it needs to we need to configure how this function is actually executed, and that's known as a **trigger** we need to create a trigger to invoke this function. Now to do that, we could configure it from the Lambda side but in this case, because it's S3 it's simpler to move across to the S3 console or open that in a new tab. I'm going to go to the upload bucket. So this is the bucket will upload the animal pictures to. I'm going to go to properties, scroll down select "Events." I'm going to add a notification. I'm going to name it "Object Upload," because that's what it does and I want this event to trigger whenever any put operations occur. So put is an upload of an object to S3. So whenever a put event happens, I want it to send a notification to a Lambda function and the Lambda function I want to be the image thumbnails Lambda function. So I select that and click on "Save." If I go back to Lambda console and hit "Refresh." So it's now got a trigger configured that's an S3 generated event. **So whenever any objects get uploaded to S3 bucket it will generate an event. It will deliver that event to Lambda. It'll trigger this function. The function will execute, run the image resize code, and then output that to the thumbnail bucket together with any logging information to CloudWatch logs**. Okay, so at this point, we're ready to test. We've done everything we need to.

**We defined the function, added the code, selected the right runtime, configured the execution role that it needs, and set up the triggering event.** So now I'm going to move to the S3 console, and I'm going to select the upload bucket. So this is the bucket that I'm going to upload all the images too. I'm going to click on "Upload" add files and then inside this lessons files is a folder called Test Images and I've got some example animal photos. So I've got kitty.jpg, which is my colleague Mark's cat. We've got Sparky.jpg which is my colleague Mike's dog and then Winky Truffles, Ruffles, and Penny that are my four cats. So I'm going to select these six objects and click "Open." I'm going to upload all of these objects to the upload bucket with the default option set, so I'll just accept every default and click on "Upload." It'll take a couple of seconds, but it's going to upload the full size version of all of these objects to this bucket and then, based on that, some events are going to be generated. Our Lambda function is going to execute. It's going to load in these original full side images and generate some thumbnails, and we should see those thumbnails in the thumbnails bucket. So I'm going to open that. We can already see they're already prepopulated. So, for example, if I look at Mark's cat I'll open thumb kitty, click on "Open" and there we go a small thumbnail version of Mark's cat. Now this has been a really simple example. What we've done is responded to some events that are generated by S3 ran a Lambda function in response and by giving it permissions needed, we've generated thumbnails based on the original objects that have been uploaded. It's a simple example of Lambda, but should illustrate the power. It's a really powerful product that can be used within event driven or serverless architectures.

There are a couple of things that I do want to draw your attention to though before we finish, because these will be really useful for the exam. So if I scroll down I want to highlight the networking section so long the function that I configured within this lesson, it's a public Lambda function. **It has access to the public internet, public endpoints, as well as any public AWS resources, but it will not have access to any private or corporate networking**.

You're able to specify a VPC for a Lambda function to occupy, and by doing that, it gets an IP address within the VPC, has access to VPC networking, as well as any private networking that's connected to that VPC. There are some downsides. It does take slightly longer to start up when it's inside the VPC but if you do face any exam questions that talk about **private networking, then that's how you do it. You set up a VPC for the Lambda function. Now when it does run, it will inherit any of the networking configuration inside that VPC. So if there's any custom DNS or custom routing, you do need to keep that in mind. Now, the other thing I want to draw your attention to about Lambda is its lack of persistence.** You need to pull any data into the runtime environment that you will use and then any output degenerate you need to store somewhere. One of the most important things to understand about Lambda is that you cannot rely on anything but an entirely clean, runtime environment. It is possible that two of the same functions in your account can operate in the same runtime environment, if the wrong one after the other but you have to assume that's not the case. Whenever a Lambda function is invoked, it gets executed, it gets downloaded to a fresh runtime environment, it runs, and then it's terminated. You've got to assume that this is completely empty and when you finish up with your Lambda function, it needs to store its output data somewhere**. Lambda functions by design are stateless. I**t's one of the reasons they're so powerful you don't need to worry about any management or any cleanup and it's the reason that they can scale almost infinite levels because they don't have to worry about state. Scaling is an important concept to understand with Lambda. Here we've only run one Lambda function, and we've run one execution of it concurrently, a specific Lambda function can run many concurrent versions of itself, and you can have lots of Lambda functions in an account, so keep that in mind whilst we're only using it for low usage it can scale to near infinite levels.

Now the output of Lambda function executions as well as being stored in this console. You can also use CloudWatch to access monitoring and CloudWatch logs to access any logs. So by clicking on the monitoring tab for a specific function, you're able to get detailed metrics on any invocations, durations, errors, success, throttles, that kind of thing. If you click on "View Logs" in CloudWatch you'll be taken to the CloudWatch log for this function and you'll be able to see details on any execution. So this is one such execution, and it details the thumbnail generation for kitty.jpg, which is Mark's cat. So it generates the thumbnail, calls it thumb.kitty and uploads it to the thumb's bucket.

Now, a key thing for the exam is to understand when you would use Lambda and when you wouldn't. So Lambda essentially runs inside this runtime environment. You can think about it as a mini Linux machine. You need to think about whether whatever you're trying to run suits having an isolated piece of code. Is it a piece of code that does one specific function? Can it run inside Lambda with the resource limitations that you've got? So if you recall, you can change the memory settings for a Lambda function anywhere between 128 MG on 3008 MG a Lambda function has a maximum possible runtime of 50 minutes. So if your usage fits inside those constraints then Lambda is a perfect product to run your short term compute, you don't need to worry about EC2, you don't to manage EC2, and you don't need to worry about the billing of EC2 where you wouldn't use Lambda is if it needs a fully fledged OS, if it needs an extensive Linux deployment, if it does need a consistent CPU. So if it's an application that requires a consistently running compute resource Lambda that's not the right product. If it requires an operating system such as Windows, it's not the right product. There are other limitations that you need to be aware of and I'll make sure I put a link in the lesson description to those limits but in general, Lambda functions are used as part of a serverless workflow as part of an event driven workflow. You can use them for automation. There's lots of situations where Lambda functions are ideal but for the exam, keep in mind anything longer than 15 minutes does not to Lambda.