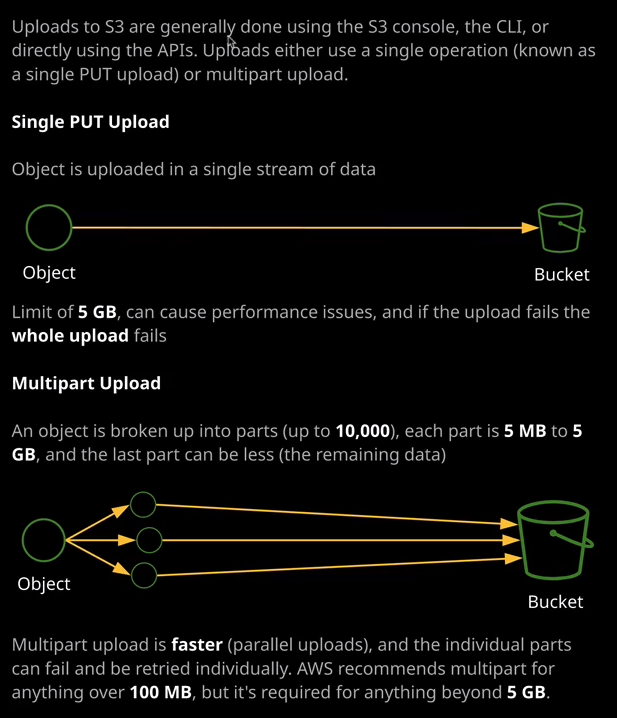
Welcome back and in this lesson, I want to briefly talk about the different ways you can upload objects into S3. Now uploads S3 are generally done using the **S3 console, the command line interface, or directly using the APIs** and the architecture of S3 uploads is to use either a single operation which is known as a **single put upload** or you can use a process known as **multipart upload**. Now, both of these come with strengths and weaknesses and in this brief lesson, I just want to illustrate exactly when and where you'd use these two different methods.



So let's get started. Now the default method of uploading to S3 using the console UI is to use a single put operation. So if I go into ac-catpics1337 and I attempt to upload an object, then by default, it will use this single put upload architecture. So let's do that. Now to do it, I'm going to move across to the terminal and generate a file that I'm going to upload. So I want to generate a file that's around 100 mg in size so I'm going to use mkfile, specify - n 100 meg for the size and I'm going to call the file 100MB.data. So there we go. That's generated the file. I'll move back to my console, I'll click on upload, add files, and then select that 100 meg file. I'll accept the rest of the defaults and then allow the process to complete. Now this is using a single operation, and it's uploading that single 100 meg file. Now single file uploads have a number of critical limitations. The first is there is a limit of five gigabytes of data that you can upload using a single put upload. So if you're using the console or using the single put upload architecture, the object that you're uploading can only ever be 5 gb in size as a maximum and if you're uploading anywhere near that size using single put upload than you are going to run into performance issues, a single put upload is using a single stream of data so you can only ever get the maximum throughput that you can do with this single stream and most the time you're uploading over the internet to remote endpoints. You often have limitations with single streams of data, so you're not going to get the maximum speed of your internet connection but perhaps more importantly, is that if this single upload fails this single operation then the entire process fails. I'm currently uploading this 100 mg file and it looks to be successful. It's almost complete but if it did fail than the entire operation itself would fail, and I'd need to rerun it. Now the risk of failure becomes even more important for larger objects. Obviously, we have this limit of five gig for the single port upload but if we're uploading an object that maybe is 500 meg or 800 meg or 1 gig or maybe even four gig in size, then if we have a failure, say, 25 or 50% of the way through then we've wasted all of the data up to that point and we have to rerun the process over again. If there wasn't this limit of five gig, we potentially be wasting or risking wasting even more data upload if we're using the single put upload for larger objects and that's where multipart upload comes in handy.

**The process of multipart upload allows individual objects to be broken up into smaller parts up to 10,000 individual parts. Each of those parts is between five meg and five gig in size, and the last one can be smaller just to cope with any leftover data.** So the architecture of this is pretty simple. You begin a multipart upload. You indicate to S3 that you want to upload an object using multiple parts. That object is essentially broken up into smaller segments. Each of those segments or parts is uploaded in parallel to S3 then once all of the parts have been uploaded. You inform S3 to complete the multipart upload and all of those parts recombined into a single object that then exists on S3.   
**Now the benefits of multipart upload is because it's multiple streams of data at the same time then you benefit from the enhanced transfer rate by combining all of those. So if you've got a per connection limitation, via your ISP or just network conditions in general are impacting individual connections then by having more than one connection operating at a time you can more effectively utilize your internet bandwidth but perhaps more importantly than that is that if an individual part fails, then you just retry the individual part. You don't need to retry the entire upload,** so let's have a look at this in practice. I'm going to go back to my terminal, and this time I'm going to make a larger file, so I'll use the same command mkfile. I'll specify -n again. This time, though, I'll specify a 10 gig file, and I'll call the file 10GB.data. So I'll generate that file. **Now multipart uploads are done from the command line.** So I'll need to configure the command line tool. So I'll do that. Now I'll run AWS configure then I'll move back to the console because I'll need my access key information. So I'm going to click on services, go to IAM, go to my user, so I'll select my user, go to security credentials. I'll delete the access key that I've already got assigned to this IAM user and I'll generate a new access key and then configure my command line tools with that access key. So I'll do that first I'll create access key I'll copy the access key ID paste that in. I'll show the secret access key, paste that in, I'll select the default region as U.S.-East-1, and then for default output formart I'll put none. Now I'll run an AWS S3 ls just to verify that this has connected successfully, which it has and then once I've done that, I can run AWS S3, copy the 10 GB data file, and then the name of the bucket and proceed to automatically use multipart upload by copying the file from the local machine to the S3 bucket. Now, this will be using the full speed I've got available my internet connection. You'll notice that the upload speed is gradually increasing. So once all of these individual uploads these individual parts take place, it'll more effectively use my full internet upload speed. If any of these individual parts fail, there'll be automatically retried, and I won't have to do the entire operation again. So this is an effective way of uploading data to S3 by breaking it into individual parts and then uploading each of these to the S3 bucket. **Now multiple upload is required for anything beyond 5 GB object size but AWS do recommend that you use multipart upload for anything over 100 MB in size.** So keep that in mind for the exam but that is everything that I wanted to cover in this lesson. I just wanted to make a brief lesson just describing the differences between the single put upload and the multipart upload because you will need both of these for the exam. You might get a question on when to use single put versus multiple upload and key facts you need for the exam is that you do it need to use it for anything beyond 5 GB in size and AWS recommend because of the faster uploads and more reliable uploads that you do use it for anything beyond 100 MB. So at this point, go ahead, mark this lesson as complete and when you're ready, you can join me in the next.