# **Amazon Web Services Machine Learning Essential Training**

### 

### **Welcome**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Langit] Hi, welcome to Amazon Web Services for Machine Learning. I'm Lynn Langit. In this course we're going to start by looking at machine learning itself. We're going to take a look at concepts such as algorithms and methods of processing. We'll then look at patterns and also service types that are available on the Amazon platform. Speaking of service types, we're going to start by looking at the newest services, and these are API services, you can even think of them as serverless services, services like recognition, Polly, and Lex. There's really a lot of new ones that have come out recently and we're going to cover them all. Then we're going to look at platforms like AWS SageMaker, which allows you to setup Jupyter notebooks as a service and to use Docker containers to optimize your model training. In addition to that we're going to look at using machine learning using Amazon hosted virtual servers. We'll look at the machine learning optimized AMI, and we'll also look at some of the deep neural networks such as MXNet, and the newer, higher level languages to make working with these complex techniques more usable such as gluon and other languages. We have a lot to cover, so let's get started.

### **About using cloud services**

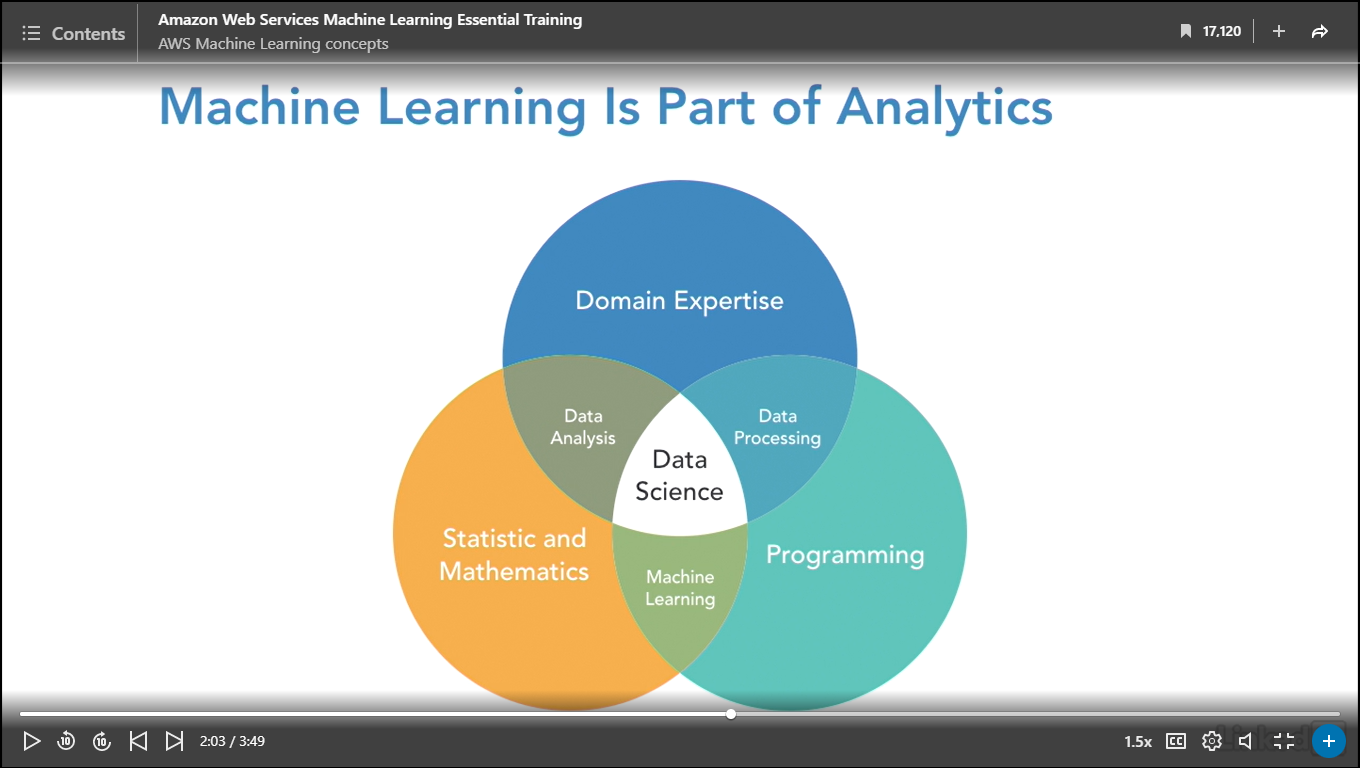
Selecting transcript lines in this section will navigate to timestamp in the video

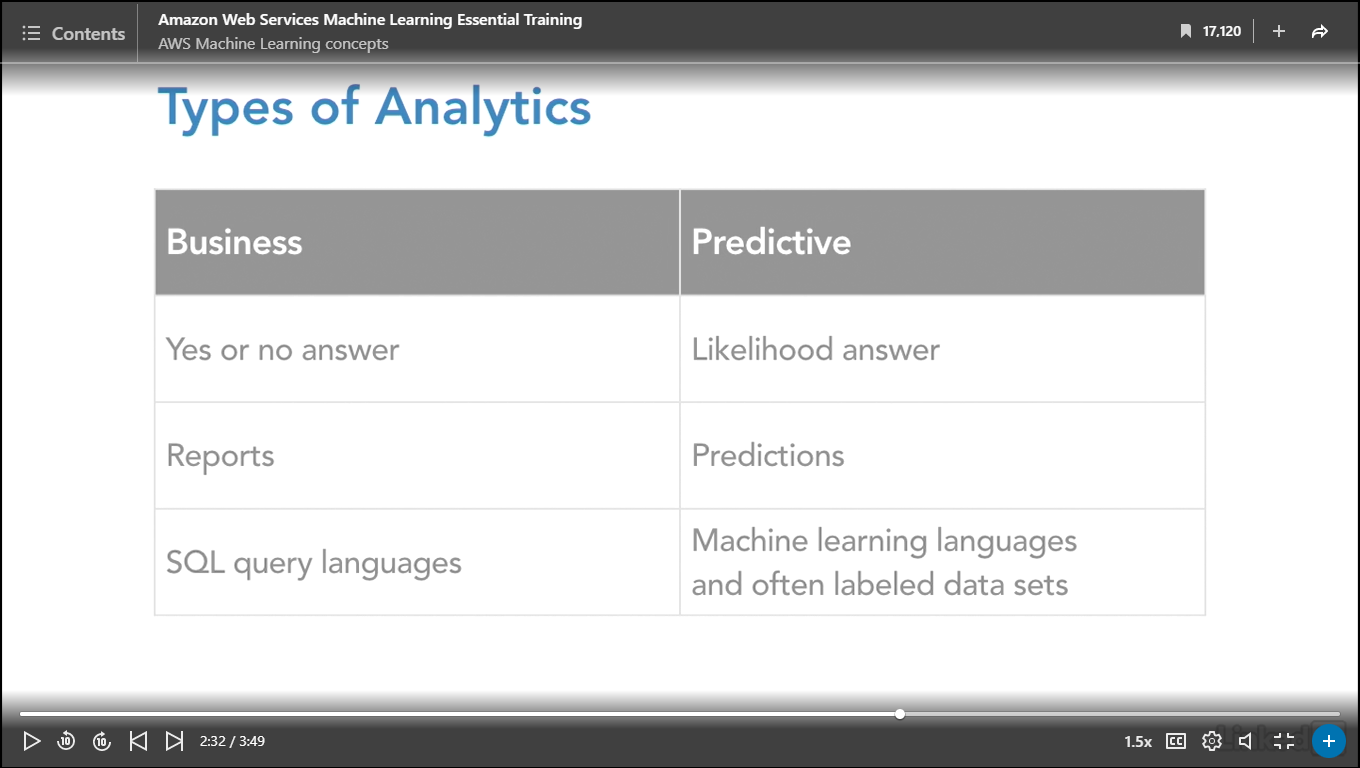
- [Instructor] In this course we're going to be working with AWS cloud based services. Now there are some best practices that I like to share. When you're learning it's always best to use a dedicated user account. In fact, it's best to use a dedicated Amazon account if you can setup a separate account. It's not always he case, but that really allows for a clean separation between any test and production environments. At minimum you want to use a unique user login. Although many of these services will be included partially in the free tier so that you can try them out, not all services are included so you really want to understand what kind of charges you could be racking up if you turn the services on, and the biggest tip I'll give you is it's usually not expensive if you just try something out quickly. Where it can become expensive is if and when you forget to turn the service off after you're done learning about it. To round out this discussion of tips, I want to take you out to the Amazon console and show you where the billing dashboard is. So here is the Amazon console, and if you click on the dropdown next to where your login is shown, you'll see my billing dashboard. Now there's many different ways that you can get service cost information from Amazon, but this is probably the simplest. Now you do need to have a high level of permission, so if you click on this link and you don't have permission to view it, then you want to talk to your organizational administrator, but this will help you, and you can see I've done some other testing on this account, and I've had some charges occurring over the past period. So you can scroll down, and you can see where the charges are coming from, which of the services. In addition to viewing the dashboard, a best practice is to set a budget. So you just click on budgets and fill in the information there to get notified if your service charges go over a certain amount. You really want to follow these best practices particularly when you're working with computationally intensive services like those we're going to work with in machine learning 'cause you can run up charges pretty quickly if you forget to turn high powered servers off for example.

### **AWS Machine Learning concepts**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] So as we get started with AWS machine learning, I'm going to throw a bunch of words at you because that is the landscape of working with machine learning. It's a lot of heighth and a lot of words. So we need to look at the key words in the domain and start with some definitions. You may have heard words like MX Net or Tenson Flow, Statistics, Predictive Analytics, Algorithms, Data Mining. It's really confusing out there because as more and more people and companies are moving into machine learning, there are different definitions for these key terms. So let's start by defining machine learning. Machine learning is an application of artificial intelligence or AI. It provides the ability to automatically learn and improve from experience without being explicitly programmed. And of course, it provides our application disability. And it uses computer programs that can access data and use it to learn for themselves. This is a key difference in algorithmic complexity versus basic computation. In machine learning, it's the combination of one or more statistical algorithms or predictive algorithms, and data most commonly. It's not required. There's some algorithms that don't use labeled or trained data. But most commonly, you will be working with a combination of data and algorithms. Another important aspect of working with machine learning is to understand that machine learning is part of an analytics solutions. Given all the hype, you'd think that every problem can be solved with machine learning. And this is really far from true. As you may know, in addition to working as an educator, I work as a practitioner. So I actually help companies build analytics solutions. And I'll tell you in my practice over the past couple of years, more and more companies are starting to use machine learning as part of their solution. Don't buy into the hype. Machine learning should not be your first go to for analytics solutions. It is a set of tools for a specific set of problems. And you'll see in the diagram that it's a part in combination with other key concepts and tools like statistics, data analysis, data processing, regular programming and domain expertise. Speaking of analytics. Broadly, there's two types of analytics. There is good ole' fashion business analytics. Which is commonly called reporting. This is getting information out of traditional, relational data storage usually. Although it can be non relational and you usually use sequel query languages. It's answering questions like, how many red sweaters did we sell in the east coast during that month in December when it was so cold? And it will result in what's called a deterministic answer. A yes or a no. Or a specific number. The new game in town, machine learning area that we're going to be working with is predictive analytics. This is based on the application of statistical algorithms and you're going to get a percentage of likelihood answer. If you had statistics, it's a P-value. It's a probability. So rather than, how many red sweaters did we sell in that store in the east coast? What is the probability that we will sell the things that we sold with the red sweaters last year? Maybe we sold black mittens. What is the probability in that area based on the algorithm and labeled or trained data that we have from last year. So they're two complimentary flavors of analytics, business and predictive.





### **Business scenarios for machine learning**

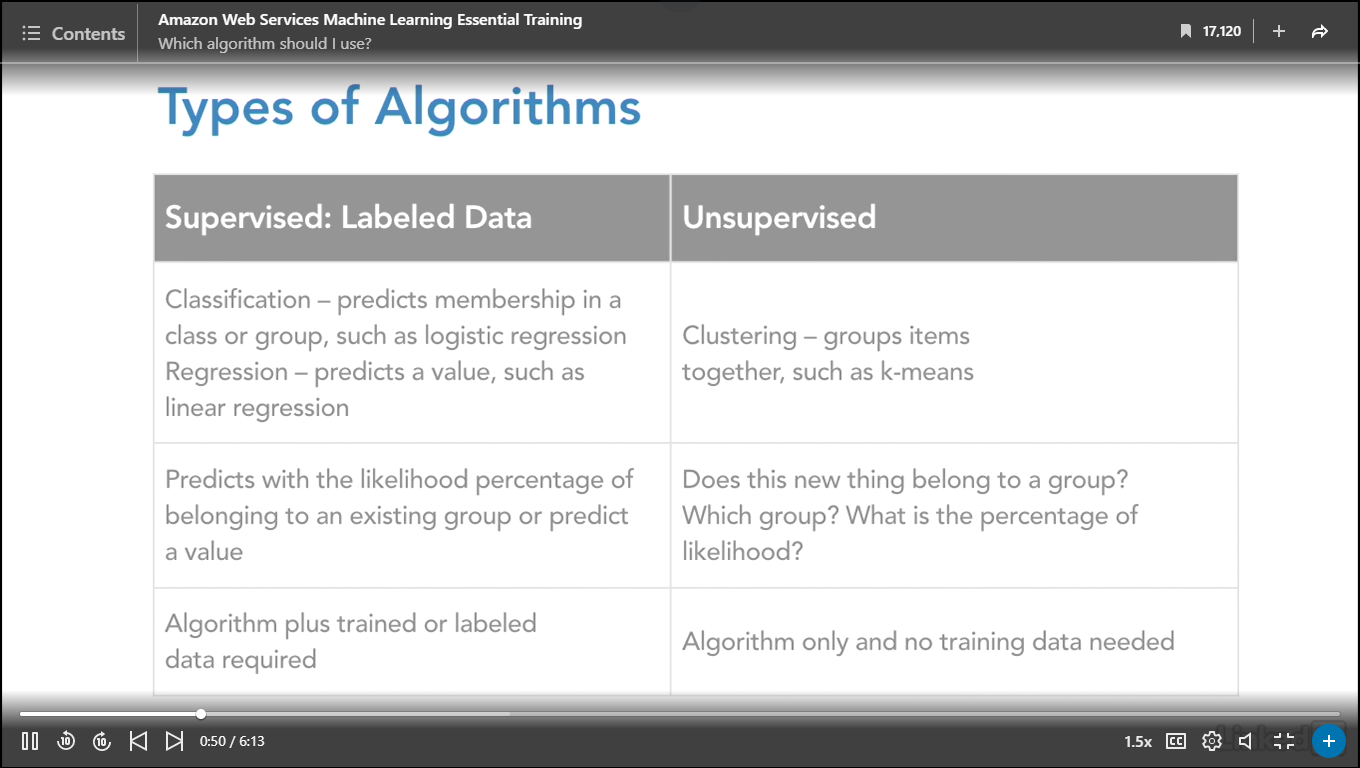
Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] So let's go to the key question, when is machine learning needed for an analytics solution? When you're considering using machine learning in analytics, you want to use the common starting points for any analytics project, whether it's predictive or not. You want to always start with the business problem, and this might seem obvious, but especially given the hype these days around the potential power of machine learning, I know I've gotten lots of calls from clients saying, "Help me implement MXNet." Or "Help me implement TensorFlow. "I think I need it. "I read an article, and it seems just really compelling." And while the new technologies are powerful and they can be a fit for some business problems, I'm going to through up a caution here that it's extremely important to always start with focusing on the business problem and then figuring out which technologies apply. Just one more tip from the real world. I've lately had a number of clients calling me asking me to implement machine learning and they didn't actually need it, they needed business analytics, they needed plan old reporting. So you always want to make sure that you start with the business problem and we're going to go into some examples here in just a second. Second, you want to aim to produce a minimum viable report or a minimum viable output, that a subject matter expert can validate. I've seen a number of developer teams become enamored of these new technologies and not share the output of their work with the subject matter experts for too long of a period of time and what can them happen is, the developers can be working with very sophisticated algorithms but producing output that's not providing any value to the business. So it's really key that you tighten up the feedback cycle and you try to get a minimum viable something out at least every week, if not more frequently. You want to use the simplest possible technology. We're going to have a discussion about algorithms, which is a key aspect of working with machine learning and it's well known from practitioners in the industry that the simpler algorithms, such as linear and logistic regression that have been around for a really long time, solve a lot of the machine learning problems, and last, but certainly not least, to use clean and understood data. Now because in this course, we're going to be focusing on understanding and working with the many different Amazon possibilities for machine learning. We are going to start with clean data, but of course, in the real world, that's a huge step. Accounting for and setting up time in your project to verify that your data is clean and understandable by your subject matter experts can be the difference between success, that is getting value out of your analytics project and failure. Now let's talk about some ideal usage patterns for machine learning. The first one is to enable applications to flag suspicious transactions. A way to address this with machine learning would be to build a machine learning model that predicts where a new transaction is legitimate or fraud. Another usage pattern would be to forecast product demand. You'd input historical order information to predict future order quantities. This is similar to the red sweater analogy I talked about on an earlier movie in this series. Yet another usage pattern is to personalize application content to predict which items a user will be most interested in and get those items in real time. This is really, really common in online shopping and amazon.com was really one of the first companies to personalize it with the you might like suggestions. Some additional usage patterns include predicting user activity. You can analyze user behavior to customize your website and provide a better user experience. Facebook is well known for this, and they in fact do machine learning in combination with what's called AB testing, where they test small changes in their features to a small number of users, get feedback, have their model learn, and then rollout features which drive business results that they're after. And then yet another one, is to what we call listen to social media. So ingest and analyze social media feeds that potentially impact business decisions.

### **Which algorithm should I use?**

Selecting transcript lines in this section will navigate to timestamp in the video

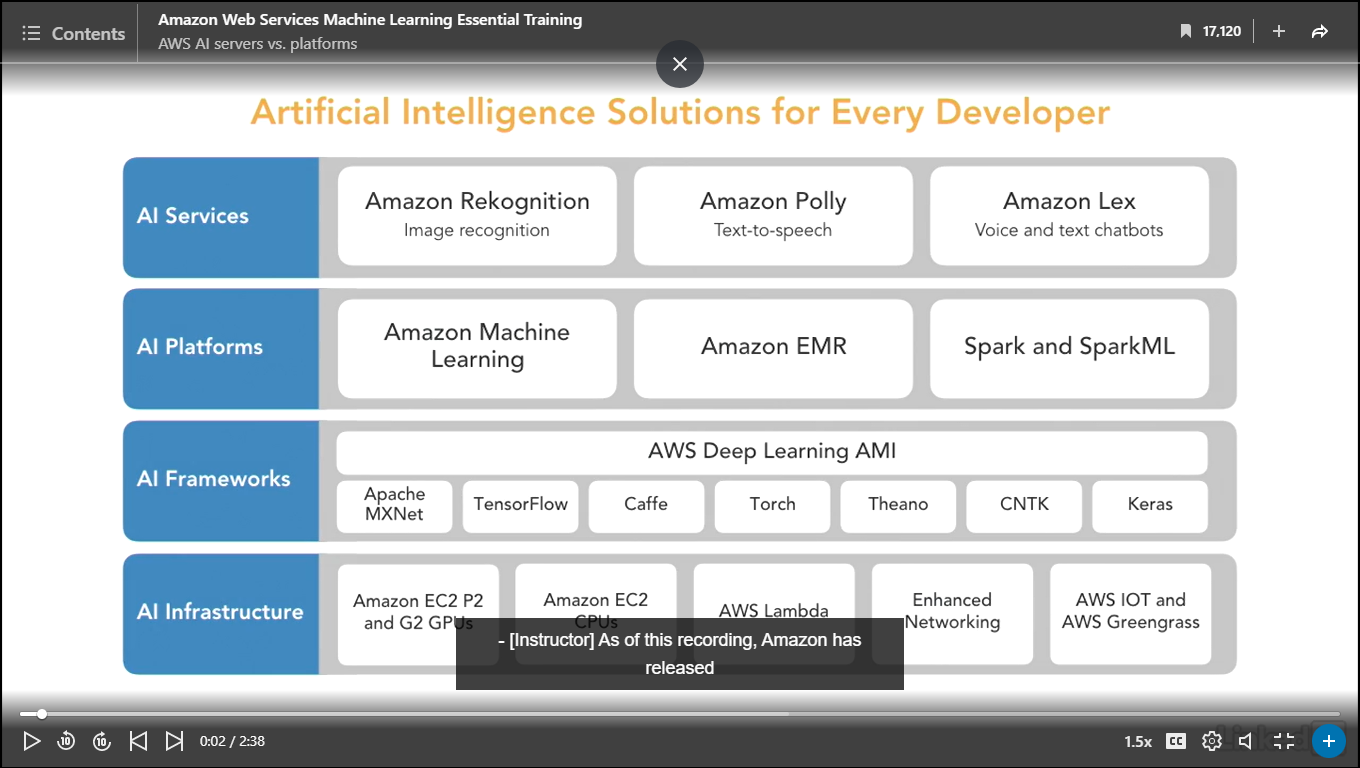
- [Instructor] After you've defined your business problem, and decided that it's a match for machine learning technologies, checked your data to make sure that it was clean and understandable, the next difficult challenge in working with machine learning is to determine which algorithm or, more properly, algorithms should I use? So to get us started, I've categorized the types of algorithms into two broad categories, supervised and unsupervised. And I've shown a couple of categories and examples of common algorithms. There are many, many more but this just kind of gets us started on this path. So in the supervised area, we commonly work with algorithms that do classification or regression. Classification results in a prediction as to whether a piece of data has membership in a class or a group. The most common business situation of this, or example, is categorizing photos. It's often of, is it a dog or is it a cat. That's sort of the Hello World of machine learning. An example algorithm is logistic regression. And I know it's strange, it's called regression even though it does classification, but that's the way it goes. The second type of supervised algorithms are regressors, or regression algorithms. They predict a value, and an example of this is good old linear regression. You probably remember back in high school algebra, when you were drawing a line in an XY axis between some points, to figure out if the line was going up, down, or if there was a line. And that's where this algorithm comes from. So it's predicting a value. It could be predicting the temperature of a Nest thermostat, or it could be predicting a price at which you would feel comfortable buying that new pair of shoes. It's a number or a value. So supervised algorithms are going to predict with likelihood the percentage of belonging to an existing group, or predict a unique value. And they most commonly are going to include the algorithm plus a trained or labeled data set. So in the case of classification, if you wanted to create an application that categorized which animal was in a picture, you would most typically need to have a training data set that was already labeled with which animal was in the particular picture. Now the other major type of algorithms is unsupervised, and the category here is clustering. And the idea is to group items together to discover natural groupings in data. An example algorithm is k-means. So the types of questions that you would answer is does this new thing or item belong to a group? Which group, what is the percentage of likelihood of belonging to this group? So to make this a bit more concrete, you might think of real estate buyers. So real estate buyers might have traits that they could be grouped together by. So urban buyers might have different interests. They might consume different media, they might prefer different types of food, so on and so forth. And this generally is the algorithm only, with no training data needed. Now I am making this a little bit oversimplified. In the real world, there's something also called semi-supervised, but again, I want to get us started in our journey of algorithms. So a great next step if you're new to this, is to work with the open source set of resources on Scikit-learn, and I have a link to the main site here. This repository is often used at universities to train students in basic machine learning, and you can see here's a expansion of my previous categorization of the algorithms. You have a start path, and you have classification algorithms. You have clustering, you have regression, and then you have a new category called dimensionality reduction. What that means in English is, if you have input, that's what's called very wide. Think of a table with a lot of columns in it, if you want to remove some of those columns of data because they're not relevant to your model. And again, the people at Scikit-learn have a little bit of a sense of humor because you can see at the bottom in the center, you get to a certain point, you're at the tough luck stage. So what that means from a practical point is not all problems are solvable with machine learning, and that is actually really important to remember as you're journeying down this path. One more concept for us to think about when we're understanding algorithm choices in the AWS machine learning ecosystem. There's a particular subset of machine learning that's really hot right now, and that's called deep learning. This was popularized first by Google's open sourcing of their version of this, which is called TensorFlow. Amazon recently has also supported deep learning, working with a library called MXNet. And you can see this representation shows that traditional machine learning relies on input. Here we have the dog example, dog and cat. So we have some input, picture of a dog, and in traditional machine learning feature extraction, which is a way of saying labeling a data set and saying whether the picture is a cat or a dog, was done by a person and then classification was done by an algorithm, and then you get the output. One of the reasons why deep learning is so exciting is you'll see that this manual process of feature extraction is, in part, encapsulated by the algorithms of deep learning. Now how this is done is pretty, pretty computationally complex, and when we get to that section of the course, we'll review in a little bit more detail how exactly this happens. But this is why there's so much excitement around this. Any time that a very manual intensive human process, expensive, error-prone, can be automated, of course the results that you can get back can be dramatically retrieved more quickly, and potentially have more business value.

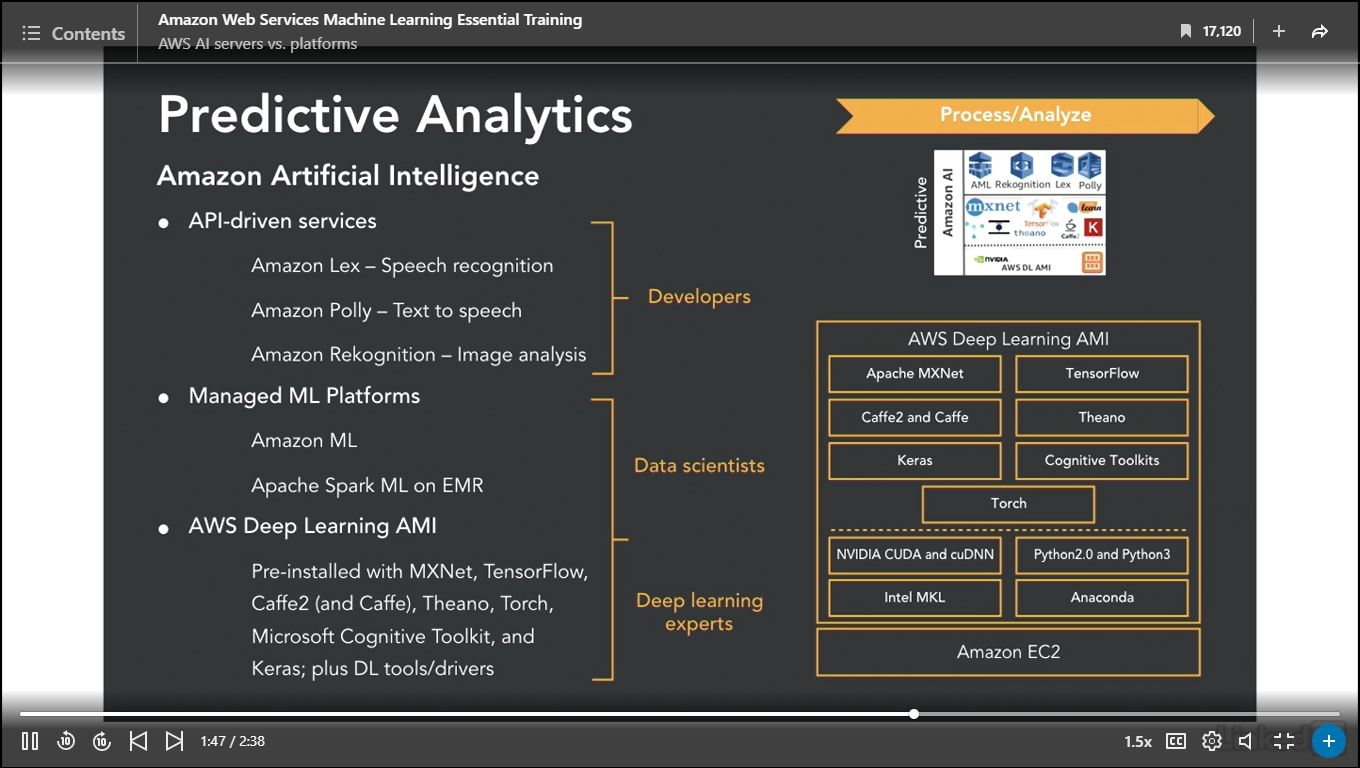


### **AWS AI servers vs. platforms**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] As of this recording, Amazon has released a very large number of solutions for not only developers but anybody working with machine learning analytics. So this chart shows a subset of the currently released products, but it is grouped together roughly in the order that we're going to be addressing it in this course. At the top level, we have AI Services, and you can think of these as endpoints. You provide input data, Amazon performs the machine learning analysis and provides you with the result. A result can be a label on an image, a cat or dog situation, it can be speech when you input text, and much more. At the next level, Amazon provides you with platforms, and these vary from server-based platforms such as Elastic MapReduce, which is ManageToDupe in Spark, to a new platform called AWS Sagemaker that's not even shown on this diagram. This is Docker-hosted machine learning containers. At lower levels, if you're coding and making your own machine learning models, Amazon supports AI frameworks for Deep Learning and is providing increasingly powerful infrastructure, particularly by adding GPUs, or graphics processor units to virtual machines, which is required for some of the algorithms, most notably Deep Learning algorithms. Now when you think about working with predictive analytics, Amazon groups their solutions by the intended users. They intend for developers to work with the APIs such as Lex and Polly, speech recognition and text to speech, and they provide the platforms for those who have expertise in algorithms and machine learning, such as data scientists and Deep Learning experts. You can see in this diagram in particular, the Deep Learning AMI or Amazon Machine Image, which can be used to quickly create EC2 instances for data scientists and Deep Learning experts. It includes a large number of Deep Learning frameworks such as MXNet, TensorFlow, Keras, and many more. It also is suggested to be run on very powerful hardware. So I'll remind you, when we get to that section of the course to work with the Deep Learning AMI, that you want to pay attention to the service charges if you choose to spit an instance along with me in the course.

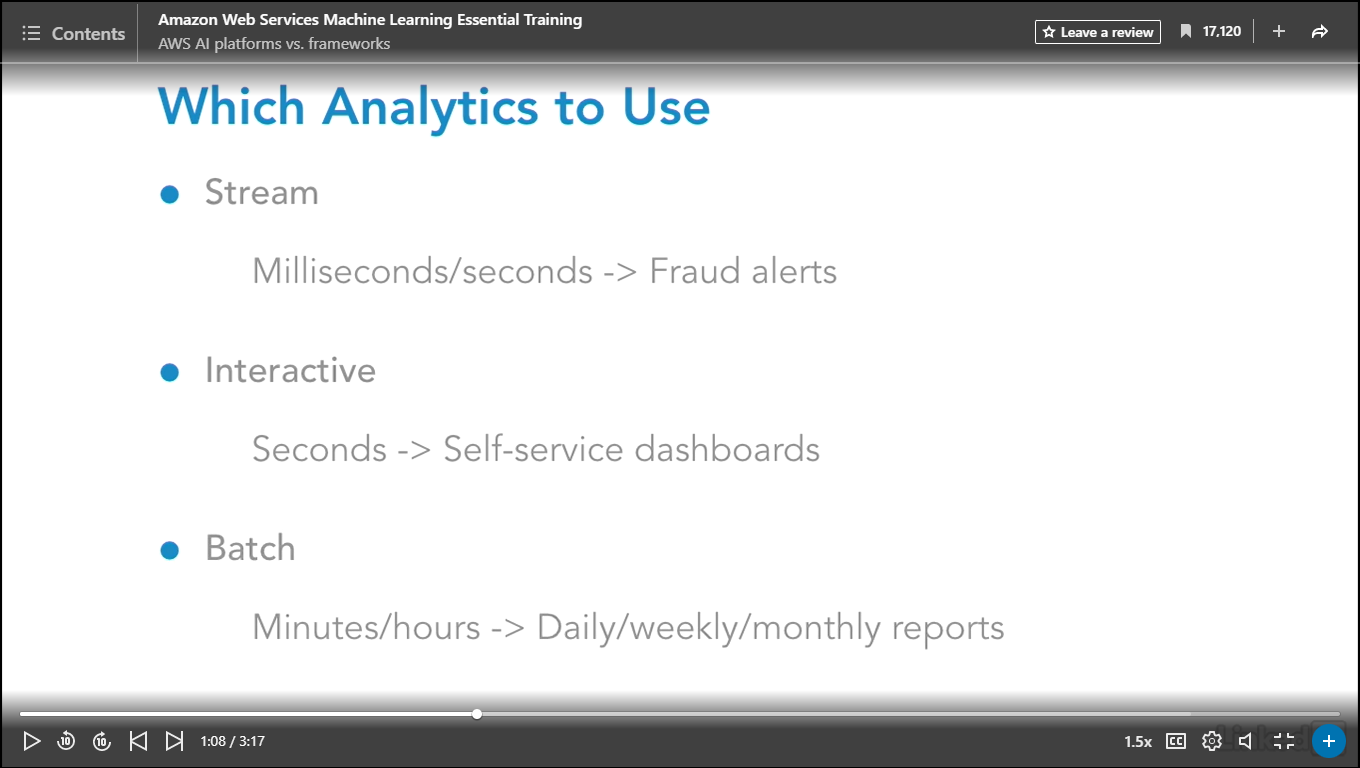


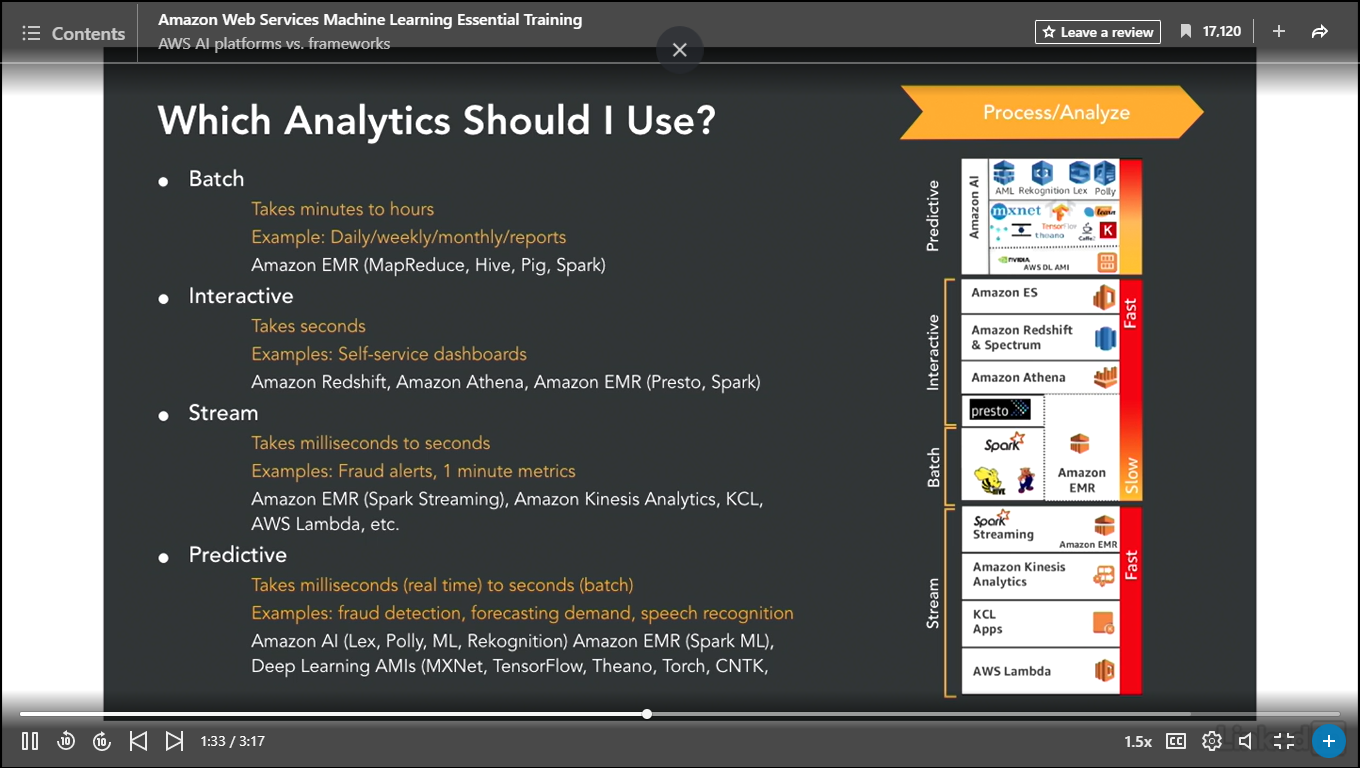


### **AWS AI platforms vs. frameworks**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] As I mentioned earlier in this course, it's becoming increasingly common in production to layer on machine learning as part of complex analytics solutions, and in fact, at the end of the course, I'm going to show you some sample and reference architectures that do just that. I thought it would be interesting, though, to consider speed or time when we're thinking about which type of analytics to use, whether business or predictive, and kind of putting the two together, because this is how it goes in the real world. So if we consider first how quickly we need to get the data back, and we think then about how we're going to get the data into our system, that can drive what we select in terms of our algorithmic processing around that data. To be succinct, it can drive the choice to whether we use machine learning or not. So considerations include, is it milliseconds or seconds? Something like fraud alerts? That's usually streamed data. Is it some kind of dashboard, and the data needs to be refreshed in terms of seconds? It's called interactive. Or is it batch? The data comes in every few minutes, every few hours, even every day. And this is traditional business reporting. Amazon matches all of these requirements to different types of products, so you can see, for batch, they recommend elastic map reduce, which is managed Hadoop and Spark. For interactive, they recommend Redshift, which is managed data warehousing, Postures, the new Athena, which is SQL queries on top of S3 data, or EMR again. For stream, they recommend EMR with Spark streaming, kinesia analytics, Kinesia client library or Lambda. And for predictive, they recommend Amazon AI, which is their set of APIs that we're going to look at, Lex, Polly, so on and so forth. EMR, with Spark machine learning, or the deep learning AMI, with some of the deep learning libraries. Speaking of deep learning libraries, a question that I get asked frequently by developers is, what programming language should I learn, so that I can become fluent in machine learning and help build models? I've pulled up screenshots from MX.net and Tensor Flow to help us understand where the vendors are going in terms of language support. If I had one language to bet on, I would bet on Python. You can see, front he most popular deep learning libraries, Mxnet and Tensor flow, they both support Python. Of course, they both support C++ as well. But between those two languages, I would go for Python. Not surprisingly, a language that is supported in Tensor Flow is the Google language GO, and I do see 'Tensor Flow applications being written in GO. For Mxnet, it's interesting to see that, rather than Java, Scala is supported. Also very interesting, the very dominant data science language R is supported. Not yet supported in Tensor Flow, in the main Tensor Flow. Also on MX.net, Julia is supported, and to be complete, Purl.





### **A classifier in action: Amazon Macie**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] To round out our understanding of machine learning concepts I thought it would be interesting to look at an application. This application uses both traditional analytics and machine learning. It's actually built by Amazon. It's a service called Macie. Now I'm just going to demonstrate how it works. If you're interested in setting it up, in the library there's a separate course that I made and if you can just click the link and follow those steps then you can set up a demo that'll look similar but not identical. What I want to show you here is the business value that you get out of a properly designed solution that includes machine learning. So what does Macie do? Macie gives you a dashboard that alerts you to security problems and risks in your Amazon account. So you can see that I've got a nice dashboard here that tells me from a high level what's going on in my account. And I have different levels associated to my users. And then I have a slider around risks. So you can see that I have risk set up here and what's being analyzed in my account is my data in my S3 buckets. And this is a really interesting business use of machine learning. As more and more companies are moving their data into S3 buckets and then doing the analysis on top of that rather than using relational databases. And of course it's much more difficult to put constraints and monitoring on S3 buckets. So this is a new application of machine learning. So you can see I've got some problems. I have aws access keys and secret credentials and secret keys up in some files in my account. Now if I want to see that you can see that I can drill into that data. I can also set the level of tolerance here and this is directly related to machine learning. So I can say I want to see items that have a risk of three, I want to see items that have a risk of 2. I can click in and I can actually see these items. If I want to see where those keys are I can click down and I can find out which bucket has those keys in it and I can actually go all the way down to the bottom and I can see that this particular file has it in it. And I can see there's the key. And in this case we're using actually traditional analytics. We're using a boolean regex. Now I can do research and I can look at my S3 objects and I can type a query. And I can see that I've got a problem really quickly. Now where does machine learning come in? So far I'm doing basic analytics. Well if I click into the settings you can see that this entire set of services is running what's called classifiers. Now how the classifiers work are either with simple analytics or machine learning. If I go into themes you can see that I have a set of predefined themes. Traditionally confidential information like credit cards, financial information. But you see I have a pencil over here. What I can do is I can further define via a custom machine learning model what is a corporate proposal in my company and I can monitor who's working with that information in S3. Are you beginning to see how a combination of traditional analytics works with machine learning in effective solutions? It's really interesting to see how Macie is addressing a new customer challenge by applying both of these types of analytics. Just to be complete I'll show you the other item that's monitored here which is called could trail. Check this out. You can see from where in the world you have accesses yo APIs that are critical for your system. Again a combination of traditional analytics applied with custom machine learning is extremely powerful when it's tied to business needs.

## **Question 1 of 2**

Machine learning typically combines \_\_\_\_\_ and \_\_\_\_\_.

* analytics; processing
* language processing; data analysis
* data; algorithms  
  Correct  
  Machine learning combines statistical or predictive algorithms with data.

## **Question 2 of 2**

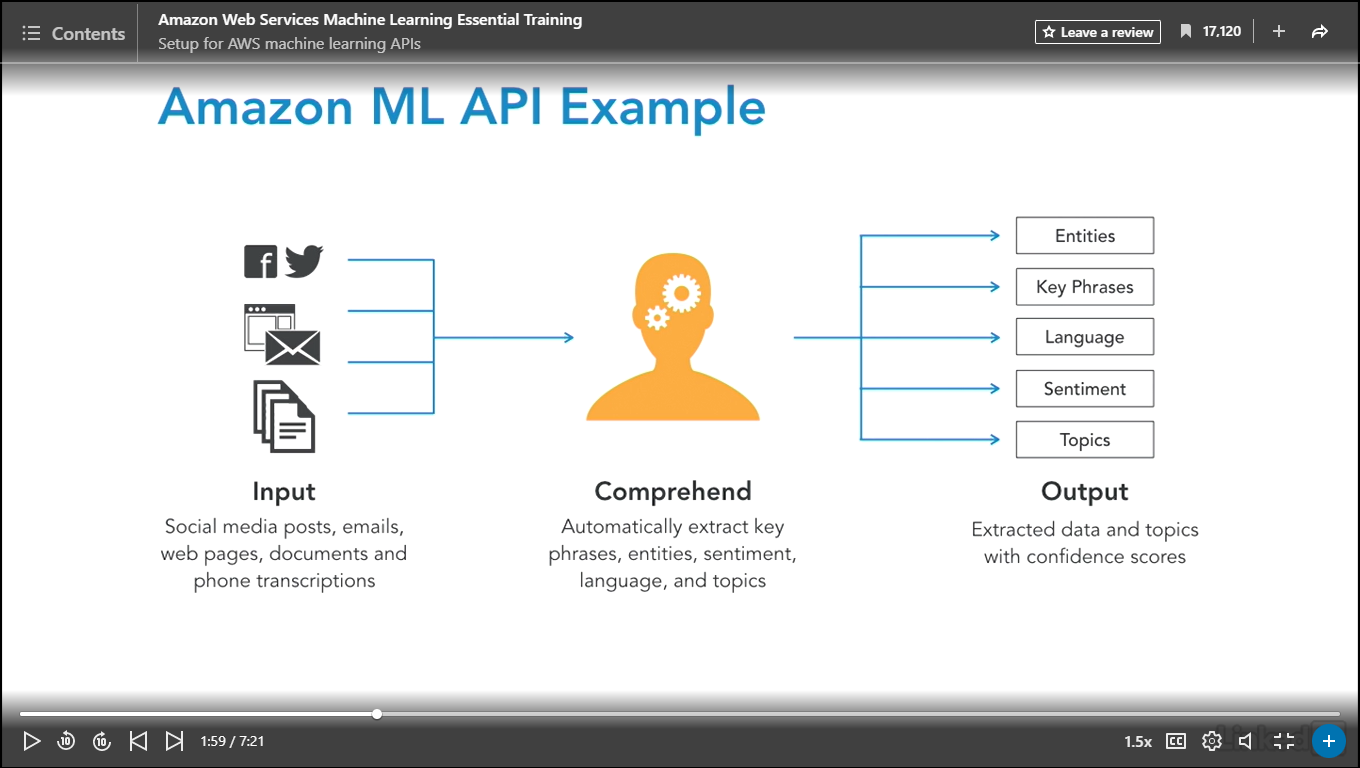
AWS ML APIs are designed to access data stored \_\_\_\_\_.

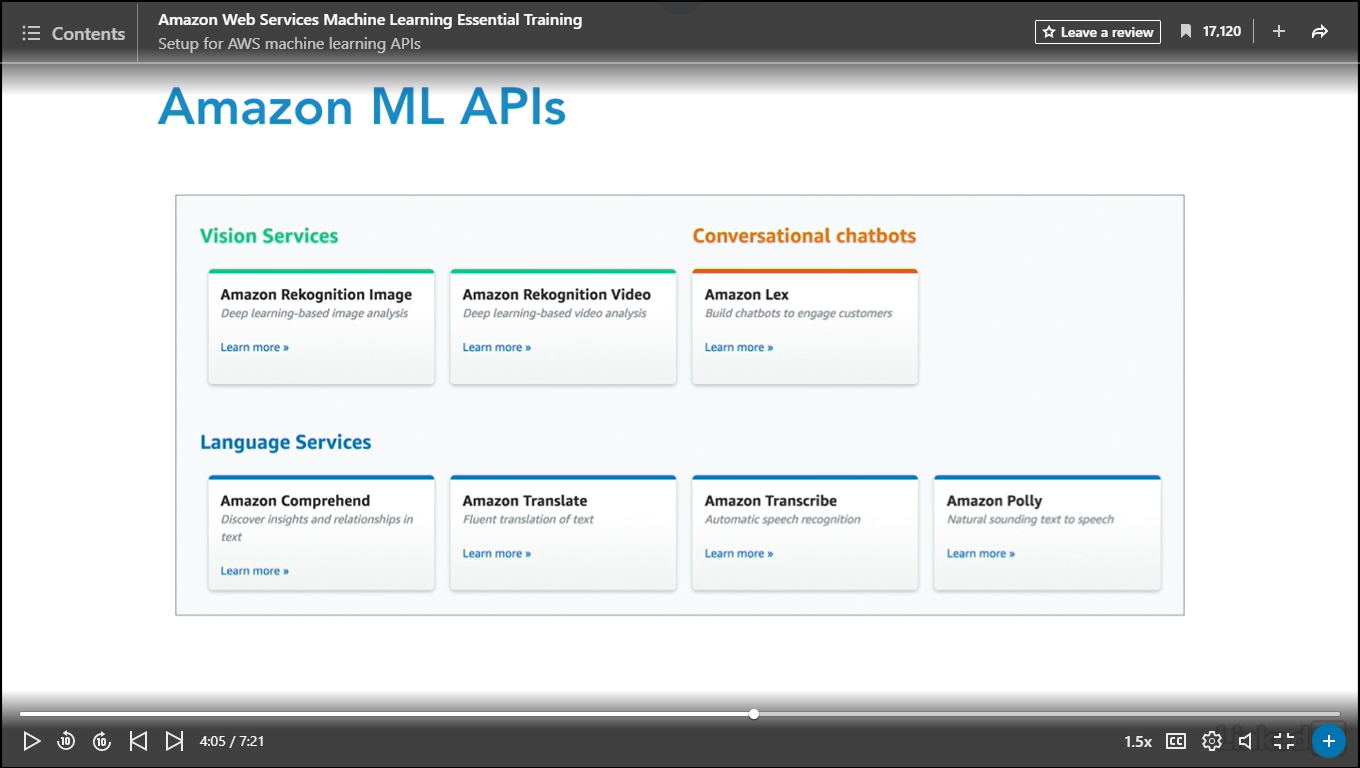
* in S3  
  Correct  
  API's access data stored in Amazon's S3 data.
* in RDS
* locally
* in Redshift

### **Setup for AWS machine learning APIs**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Narrator] We're going to get started working with Amazon Machine Learning APIs, or Application Programming Interfaces. So what are these? They are a set of tools that allow you to work with machine learning in an easy way. You can get up and running pretty quickly. You'll see when we start working with the specific APIs. They consist of an algorithm or a set of algorithms plus importantly, Amazon's own labeled data models. So let's make this concrete. One of the APIs is recognition, and what this does is image labeling along a number of different feature vectors. So specifically it can do things like identify faces in photos. Even to the extent of identifying celebrity faces in photos. And this is where using the Amazon APIs allows you to create applications more quickly because they've got a massive labeled dataset. They use this for their own applications on Amazon.com and also to help manage AWS better. So you're benefiting from their labeled data set. In addition, the algorithms are pre-tuned. You basically just supply your input data so your unlabeled photos for this example, and you retrieve the predictions. So the likelihood that the photo is superstar A or superstar B and then you can incorporate that into your business application. So let's take another example because in the machine learning space, this presentation of APIs is very new among all the cloud vendors. So looking at another one of these APIs called comprehend. The way that it works is you supply the input data and you generally will put it in S3 in the form of files although some of these APIs can work with some of the relational data source and some of the other services up on Amazon. The key is your data has to be in Amazon. So some example input for this particular API are social media posts, e-mails, web pages, documents, phone transcriptions. Comprehend is an API that does natural language processing. So it will extract key phrases, entities, I like to call nouns, sentiment, or the tone of the particular text, language, so what human language is predominant, and topics, so subjects. And then you can take these outputs, and they're shown over on the right, so entities, key phrases, language, sentiment, and topics. And the scores, cause remember this is predicted. It's always a percent likelihood that it's an entity, or percent likelihood that it belongs to a topic rather than an absolute yes or no. And you can incorporate that into your application. Now as we go through these APIs because this is such a new area and it's difficult not only to process all the APIs that you can work with, but I think more importantly what you would do with them in terms of applications. So Amazon's done a good job talking about customers who are already using these different APIs and I'm going to highlight some of these customer stories from Amazon and my own experience as we go through the APIs. This is an active area of product development among, not only Amazon, but all the major cloud vendors. So be sure to check in the console and Amazon documentation we are actually working with this. As in the time of this recording, Amazon released a large number of these APIs to production just a couple of months ago at their annual reinvent conference in November of 2017. So by the time you watch this there might be more because this is an area where the cloud vendors are just continually putting out products. Because the usability of a machine learning API versus building a model from scratch is quite profound. And one of the big challenges out in the tech market is that there is not enough skilled labor to create machine learning models from scratch. So this is basically a practical consideration democratizing or making machine learning more accessible by exposing the APIs and really to a large extent, abstracting away the black box of the machine learning algorithms. And to that end this won't work for all use cases because in some use cases it's not the best business situation and sometimes there are even regulatory challenges that require that you have to be able to provide what is exactly happening in the processing of the algorithm. There is for example some legal consideration around situations like deciding whether to grant financial loans or not. There are laws regulating the fairness of this. And these kind of pre-baked APIs, until they have some visibility to the inner working, which I think will come, they are not the best fit for those kind of use cases. So anyway, what are the APIs? What are we working with? They're broadly categorized in the Amazon ecosystem in three categories: vision, conversation, and language. So in vision, it's called recognition and recognition of both image or video based content. And note that it's categorized using deep learning. So it's an implementation of deep learning or neuro-nets that is much simpler to use than a custom model, which we will also cover later in this course, but is a good entry point to see what's possible. In the area of conversational chatbots, we have Amazon Lex, which is designed to build chatbots to engage customers. This is also the technology behind the very popular Amazon Alexa device. We have a set of language services. Amazon Comprehend, insights and relationships in text. Translate, fluent translation of text. Transcribe, automatic speech recognition. And Polly, natural sounding text to speech. As of this recording, translate and transcribe are in private beta so you'd have to sign up to see what they do although we will talk about the information that Amazon has shown publicly. So from a practical point, when you're working with APIs, the first thing is you have to understand what they do to figure out if they can help you solve your business problem cause as I mentioned at the beginning of this course we always want to focus on using the proper technology for the business problem we're trying to solve. And Amazon has provided test harnesses in the console and we'll be looking at all those in the subsequent movies so that we can understand core functionality. Once you understand whether or not the functionality's a fit then the first level of automation would be to be scripting calls with batches of data through tools like the aws-cli. Taking it to the next level when you're incorporating this functionality into your business application, you'll be using the AWS SDK. As a general pattern you're going to be processing data stored in S3, which is going to require IAM user permissions on your bucket or buckets which you're interacting with. Although you can interact with other data sources such as Kinesis Streams and a RedShift Data warehouses and so on and so forth. We're going to just keep it focused on the APIs, so we'll be working primarily with S3 in this section.

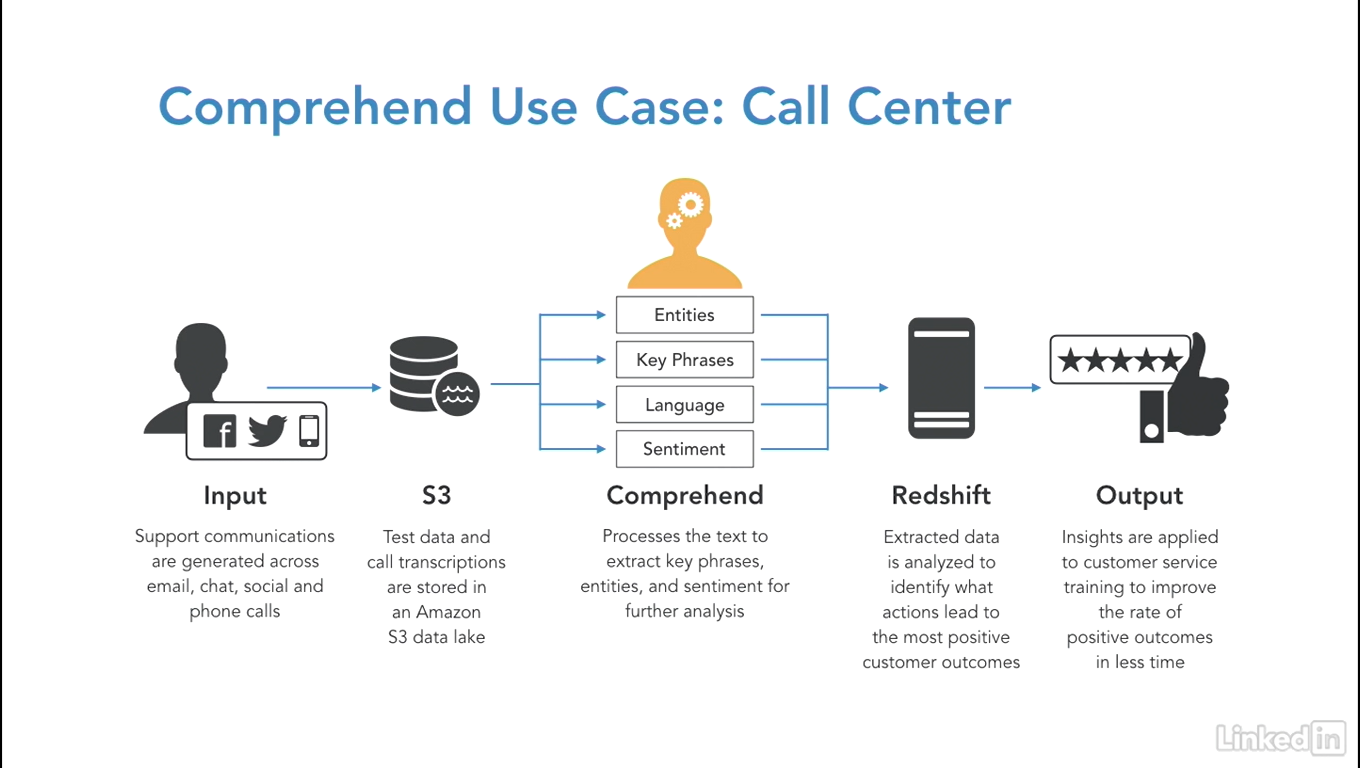




### **Predict using AWS Comprehend for NLP**

Selecting transcript lines in this section will navigate to timestamp in the video

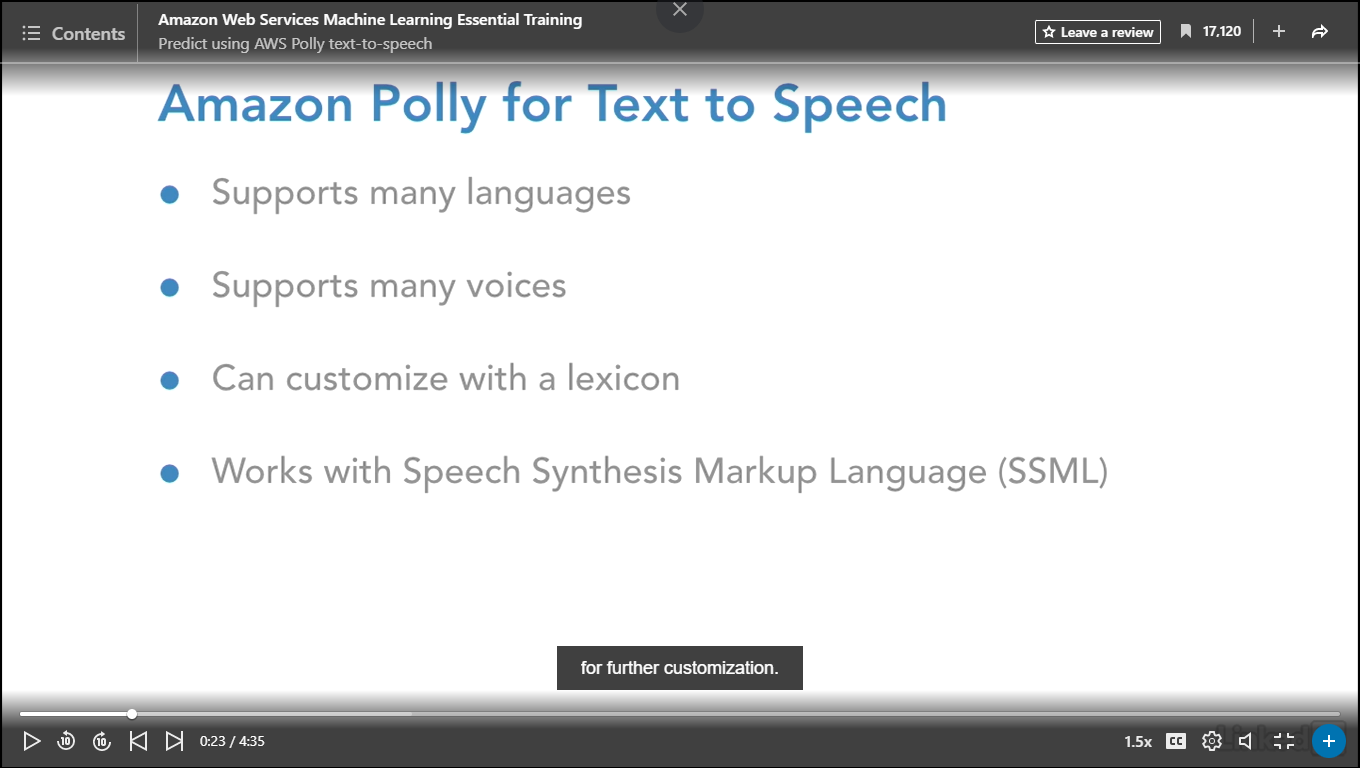
- [Instructor] To start, we're going to work with AWS Comprehend for natural language processing. It will identify in text entities, key phrases, language and sentiment. It also can do topic modeling, and as of this recording, the interface is in English or Spanish. So I type comprehend, and click Get started. And this takes us immediately into the API explorer and presents us with the results of the analysis. So you can see that we've got some sample text here on the left side in the API explorer. On the right side, we have Entity information, so we can see the category, these are types of entities, the count and here's the confidence, or the P value or the predicted score. If I scroll down, we have key phrases, same situation. Scroll down, here's the language. And here is the sentiment. So it's a neutral tone. So I'm just going to type some text in here, and then we'll reanalyze it. All right, and I'm going to click Analyze. Have I spelt Carpenteria right? Probably not, I'll just say Santa Barbara. Can never spell my city. And click Analyze. And you can see there's been some additional detections here. So today, Lynda.com, Santa Barbara, California, and if I scroll down, the sentiment is now positive, because the weather is beautiful and I feel happy to be teaching. So that is how this API works in terms of identifying the various entities, key phrases, language and sentiment. Now the second aspect of this API is working with the Topic modeling. So let's click on that. And let's click Create. So I'm going to use some of Amazon's sample data, and pre-populate this. I'm going to call it demo-job. And then I'm going to select an output bucket that I've created in advance. So this one. And then I'm going to need an IAM role, because I've got the service interacting with S3. And I'm going to call this demo-ml, and click Create job. All right, my job's in progress, so I'm going to click into it. And now my job has completed. And if I scroll down, under the actions. There's the output data location, which is right over here, and there's my output, which I've downloaded and opened in an editor. And you can see we have two files. The first one is document topics, and the second one is topic terms. Notice that in the first file, we have docname, topic and proportion; in the second, topic, term and weight. All right, as I mentioned when in introduced the top of Amazon machine learning APIs, I said in addition to exploring what they do, I would share at least one use case for how they can be integrated into business scenarios. So let's do that now. So a business scenario that can work with Comprehend is managing a call center. And here is an example architecture. So you have some input coming in, maybe from social media or from somebody's phone. And that information is stored in S3. So the text data and the call transcriptions are in an S3 data lake. And then Comprehend processes the text to extract key phrases, entities and sentiment for further analysis. In this scenario, the results are dumped not into S3, but into Redshift. And this is done so the data can be analyzed in aggregate to understand what actions lead to the most positive customer outcomes. And of course, this can be used by business analysts to determine how to provide more emphasis to the positive outcomes and to reduce the emphasis on the actions that are resulting in negative outcomes.

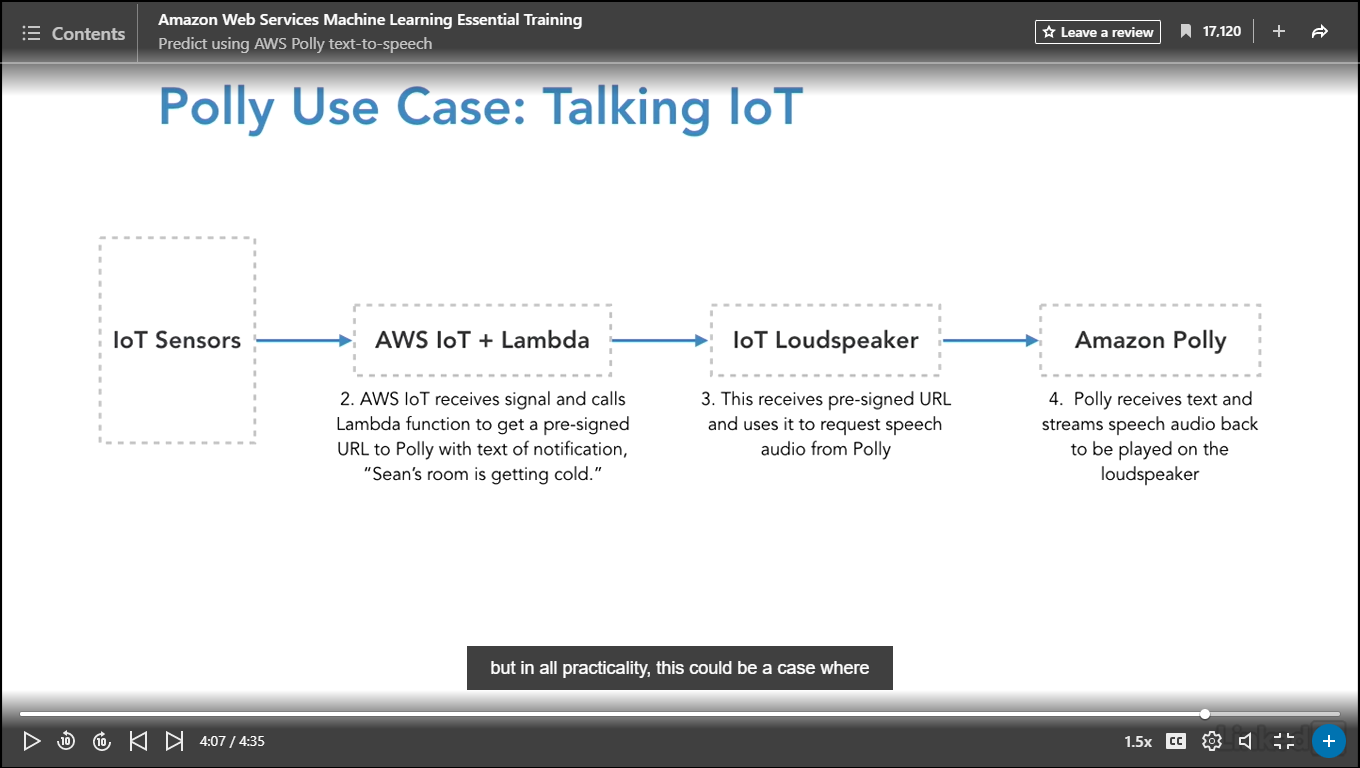


### **Predict using AWS Polly text-to-speech**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Lynn] The next machine learning API for us to look at is called Polly, and it works with text to speech. It supports many languages and many types of voices. You're going to see it's pretty fun to play with, actually. You can customize with a lexicon and it also supports a standard in this domain called SSML, Speech Synthesis Markup Language for further customization. Let's jump over to the consol and work with it. So, you can see I have two inputs here. I have plain text and SSML. So, Alright, so, what can we look at here? So, by default we have US English and we have Joanna, who's the female and we can listen to the speech or download MP3. So, let's listen to the speech. - [Joanna] Hi, my name is Lynn Langit. I will teach you how to learn with Amazon machine learning. - [Lynn] Alright, so I'm going to pick a different voice here. I'm going to be Kendra now. - [Kendra] Hi, my name is Lynn Langit. I will teach you how to work with Amazon machine learning. - [Lynn] Now, I'm going to work with language and region. I will be Welsh and I guess I'm going to be a male. - [Geraint] Hi, my name is Lynn Langit. I will teach you how to work with Amazon machine learning. - [Lynn] Oh, I rather like that. Let's see if there's anything else we have in here. Oh, just for fun let's try a complete different language. Let's try Icelandic and I'm going to be Dora. (speaking Icelandic) (laughing) Oh, I have to admit. This is kind of amusing. So, customize pronunciation. You can work with lexicons, like I said. You can modify by uploading and applying lexicons, which maps words, written representations, and pronunciation. And to be complete here, we have SSML. And let me switch this back to, let's see what else do we have in here. Let's do Spanish. And I will be Miguel. Alright, and this is Miguel speaking Spanish here. (speaking Spanish) Okay, and you can tell I'm just having too much fun with this. So, I'm just going to do one more for my Brazilian friends out there. Let's see, I'll do Victoria. (speaking Portuguese) How much fun can we have, huh? In any case, so the next part of this is to work with lexicons. So, we would upload a lexicon and provide further customization. So, what can you do with Polly? A fun Use Case, I think, is Amazon sample, which is called Talking IoT or Talking Internet of Things. And the example here shows IoT Sensors so, I don't know about you, but I have a Nest thermostat so that's kind of what I'm thinking here or some type of Smart light bulb or, you know, all kinds of the IoT stuff that we have in our houses more and more frequently these days and you can associate the messages that are coming out of those devices with Amazon data collection with Amazon IoT and subsequently process those messages with Amazon Lambda. And then we have an IoT Loudspeaker. So, this is some kind of device that would be amplifying the sound and then Amazon Polly receives the text and streams the speech audio back to be played on the loudspeaker. So, assuming that's like an Alexa or something like that. So, I think I'm actually going to have to build this. This just seems too much fun talking IoT, but in all practicality, this could be a case where maybe there were mobility challenges by the person living in the building or the house. Maybe they were not able to reach the devices to adjust them, maybe they had visual impairments, so they couldn't see the controls or they needed to have the controls read to them. So, there's a lot of really interesting Use Cases, just in this one domain. In Talking IoT and lots more across other domains. So, that is Polly.





### **Predict using AWS Lex for chatbots**

Selecting transcript lines in this section will navigate to timestamp in the video

[- Instructor] The next service we're going to look at is called AWS Lex and it's used for conversational interfaces, sometimes called chat bots. This is the same sort of services that's used by the really popular AWS Alexa and it uses machine learning in combination with both voice and text to create these conversational interfaces. There are a lot of concepts involved so before we explore on the console, let's take a minute to look at the concepts. So each set of services is called a bot. So here we have an example of ordering flowers. A bot has several objects associated with it. The first object is called an intent, and I think of this like a verb or a method. In this case we have delivery order, so this is a particular goal that the user wants to achieve. Then we have utterances. Here an example is, "I'd like to order flowers", spoken or typed phrases that invoke the intent. That basically called delivery order, in this case. Then we have the slots, the data that the user must provide to fulfill the intent. Of course, Lexa is responding here after the customer says, "I'd like to order flowers." What kind? And the customer says, "A dozen roses please" Then there are prompts which are in response to utterances. So once the customer says, "A dozen roses" Lex answers, "Where should we deliver?" And then the customer gives the address and then the last object is fulfillment. The business logic required to fulfill the users input. Now there are a lot of concepts here, a lot of functionality is prepackaged for you. The best way to try this out, like with the other machine learning APis is in the console and the demo really is pretty good. Let go over and take a look. I'm in Amazon Lex, and I'm going to click the blue get started. I have different samples that I can work with. I could do a custom bot, I could book a trip, I could order flowers or schedule an appointment. Since we looked at order flowers previously lets look at book trip. I'll call this book trip demo. You can see that I have intents, utterances, slots, prompts and fulfillment. I'd like to book a hotel. Sure, which city? New York. What day do you want to check in? There it is, and then book the hotel. Notice, there are two other aspects of this an IAM role, it's automatically created and then COPPA which is a US regulatory set of standards around children's online privacy protection. It's not going to be subject to COPPA because children can not book hotel rooms in the US. Let's click create. Now we have our sample bot and you can see we are in the section around the utterances. So these are the phrases that the bot can respond to and you can add more inside of here. Then we have Lambda initialization validation. Then we have slots. This would be the information that the person who is interacting needs to supply. So the pick up city, the pick up date, the return date. So on and so forth. And we're in book car, which we get book car and book hotel. Then the confirmation prompt and then the fulfillment Whether it's something that Amazon can ship, so if you're ordering a product that Amzaon.com or if you're going to get an interaction with some other service provider. Let me test this out, by clicking on test. I'm going to type an utterance. I'm going to test this out by typing, book a car, and then pressing enter. Notice I get the response, "In which city?" I'm going to say, "New York City" Then I'm going to say, "On what day?" Now the reason my microphone is grayed out is because I would have to enable interaction with my microphone, then I could speak to the bot. I'm just going along the process here. Typing in based on the prompts. Ready for fulfillment. You can see that there's a lot of customization capability inside of here. You can go into the settings and you can set up aliases. I'll work with the voices and you'll probably recognize this probably from the previous movie. These are the voices of Polly the output voices. Then you can set channels to integrate with popular third party APTs such as Facebook, Kik, Slack or Twilio and then you have monitoring here. Our bot's successfully built because we were able to test it. The next step would be to publish it. This is the last step before we can connect our bot to a mobile app or a chat bot. I'm going to pause here and go back to the use case. So an example use case for this would be patient bookings. We looked at booking hotel rooms, ordering flowers, and booking rental cars. But another flavor of this would be to book patient appointments. You can see the input device, the patent requests an appointment, and then Lex recognizes that scheduling an appointment has been requested. Then Lex is going to respond to the input, asking for the preferred day. Then using subsequent downstream processing with Lambda the appointment time is reserved. The message is sent back through Lex and the patient is notified that the appointment is made. The data generated form the downstream processing in Lambda can be persisted in your choice of persistent stores. In this scenario they're using Dynamo DB and they're using SNS and SES and other services.

### **Predict using AWS Rekognition for images**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Next up we've got AWS Rekognition for Images. This API can detect objects and scenes. It can be used for image moderation. It can detect facial information, comparison between faces, and it can detect celebrity faces. It also can detect text in images. So I'm going to type in Rekognition, and I'm going to click on the blue try demo button. Now you can see on the left side here, we have demos set up around image detection, and we're going to cover those in this section, and video analysis in the next section. So we've got some default images here to start with, and the first set of labeling that the API can produce is object and scene detection, so you can see that we've got this sample image, and you can actually try this out with your own images, too. It's really actually super fun. So I recommend if you're following along, you do that, once you look at the Amazon demos, and you can see that it shows the likelihood of the objects and the scenes, and I click down show more, and you can see all the information here in the likelihood. All the labels basically in the declining probabilities, or decreasing probabilities. So I'm going to go ahead and collapse this, and then you can see the request being passed as a JSON object, and the response which is the labels. Now if I want to switch images to this one, I can just switch to this one, and again I can see the same pattern. So object and scene detection and gives a confidence score. The next capability of this API is image moderation. So what this does is it detects explicit or suggestive adult content in images, and provides confidence scores. So the images blur by default, and nothing has been detected here. So family picnic. The response is no moderation. Now the next image is a woman wearing swimwear, and that could be considered a suggestive in certain business situations so that was labeled that way, and then you can try this on your own images as well. The next demo is on facial analysis. So you can see here that we get some information about the face, if it's a face, female or male, age range, smiling, happy, glasses, and then showing more information. Kind of amuses me about mustache and beard. It reflects a, I think, a male bias in the training. That is actually a subject for a different discussion around machine learning. It is interesting though to try out the demo to see what capabilities are being checked for, and to see if that matches your business situation, and then if you want to go to more than one face, you can use this image. And the way the API works is it gives you an array, so you can see here's the guy's face, and if we click this little button, and it's the next face and then it's the next face. And again, the input is just the file, and looking for all attributes, and the output is all the information about the faces that were found. The next one is celebrity recognition. I think it's interesting who Amazon sees as a celebrity. Not surprising, Jeff Bezos and Andy Jassy. I guess they worked hard and they earn it. I've from L.A. so my idea of a celebrity is slightly different. It'd be interesting to see how they trained this to see if the more traditionally thought of entertainment celebrities are identified. I'm sure they are. I'm just kind of joking. But you can see celebrity recognition here. Then the next one is face comparison. So you have a reference face and you have comparison faces, and then you have whether or not they match. So we can see that if we selected this face. Course one of the things when you're exploring these APIs is here it's telling you about constraints around the provided inputs. So in this case, image has to be JPEG or PNG. No larger than 5 megs. And your image is not stored. It's just scanned and then the results are given to you, and again the same thing for the comparison faces. And then the last one is text in image. So you can see this identifies the text in the image, and this one's kind of cool. The license plate. So you can see that there. So, lots of different classifiers or labeling. So what we're coming down to is would the classifiers that are provided and the data that's provided. I was kind of joking around with the celebrity recognition. I'm sure it works as expected. I just thought the reference samples were amusing. In any case, you look at how the API performs by testing it with your data, and then does it provide the labels that you're looking for, for your business case? So let's go back and consider our business case for recognition for images. So a business use case could be working with real estate. So in this case, you've got an image capture coming through your phone and then it's uploading to S3. Then the Lambda function is basically an invoker. It's like a trigger on the S3 bucket. The pattern is similar to the old world of relational database triggers. It's just that the underlying database here is S3. And then on upload this Lambda calls Rekognition, which then retrieves the image from S3, and returns the labels for the detected image. Now in this case, it's going to be object and scene detection. And then this information is passed to another Amazon service Elasticsearch. So, this labeling is done relatively automatically through Rekognition, and then Elasticsearch is further search criteria that's integrated into the application. So, it looks like in this case near to the space needle in Seattle is desirable in the case of real estate, based on the illustrations in the pictures. So in any case, the labels are generated relatively automatically and then the downstream processing is done with Elasticsearch here.

### **Predict using AWS Rekognition for video**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] The next part of this API is Rekognition labeling for video input. As of this recording, this API can detect objects and activities, and can be used with moderated labels. Again, this is an active area of development, so please check the actual product and documentation for additional features when you're watching this video. So we're back in the console, and we're in Amazon Rekognition, and we're going to click on Video Analysis. So as it says here, this service analyzes video, and detects objects, activities, people, celebrities, and more, in videos and livestreams, and we've got this sample video here. Now, if you upload your own video, it has to be a really small size. For this demonstration I looked up the documentation, it was like one minute, and the idea of course is to keep the sampling capacity down, 'cause this is a service that you would pay for, and this sample is free. So if you're going to test this out, you're going to need to move into production to work with videos, unless your videos are only one minute long. So you can see, here's a short video, and if I go through the video, you can see it's two leaders of Amazon interviewing each other basically. That's what's going on in the video, and it's a 18 second video. So you can see that there are 13 results, and two people. So Werner and Jeff, and then they're both celebrities, and here are the labels, and there are no moderated labels. And then you can download this response as a JSON file. So it's similar conceptually. I think the difference probably is just showing it all at once here, so the objects, the labels, so on and so forth. So a use case for this could be around celebrities. So we would have input. So a user would upload a new video, either mobile app or some other application, and that would go into S3 again, so we've got the familiar Data Lake pattern, and then Lambda would be triggered on upload of a new object into an S3 bucket, and then Lambda would call Rekognition Video, and the video API would analyze the footage, and it would return the names, the timestamps, the IMDB links for all the detected celebrities, and then the metadata is indexed in

### **Predict using Transcribe and Translate**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] The next two APIs, Transcribe and Translate, are still in private beta as of this recording. I'm going to describe their functionality, and their developer guides are already available and I've linked them here so you can understand a little bit how they're designed to work. So Transcribe automatically takes sound files and recognizes the human voices and translates the speech into text output. It can be used to timestamp audio files, such as MP3, WAV, et cetera. And as I mentioned, as of this recording it's in private beta. AWS Translate translates using natural language. The currently supported languages are Arabic, Chinese, French, German, Portuguese, and Spanish. Again, please check the documentation when you watch this recording. I'm sure there'll be more as this moves from private beta into public beta and GA. AWS Translate is designed to integrate with other machine learning APIs. For example, Polly for speech output, S3 for document repository translation, and Comprehend to extract named entities, just to name a few. As we complete this section around machine learning API services, I want to remind you this is a really active area of development, and when you watch this, there'll probably be new capabilities added to the services that we looked at and possibly even new services. The other thing that I want to emphasize is because this is so new, I see a lot of solutions being modeled without even considering or trying out these types of services, not only from Amazon, but from other vendors, because this is a new way of working with machine learning. So as I mentioned at the start of this course, a best practice with any analytics solution, but particularly machine learning is to use the simplest possible solution that will fit your business use case. And I'm really impressed with the capabilities of these services right out of the box. So I know as an implementing architect, I will be urging my customers to take a look at them. And I consciously put them in the beginning of this course because I think that they have great potential for democratizing and allowing more of us to add machine learning capabilities to our analytics applications. Now, that being said, there are other situations and use cases for which these services do not fit. Those would be cases where there are limits on the amount of data that you could input. There are limits on what's output. I spoke specifically around looking at the labels that come out and recognition as an example. If those labels don't fit for your business case, then you might have to create a custom machine learning model. So how do you do that? You go lower in the stack, and that's what we're going to be doing in the rest of this course. We're going to be working at the platform, framework, and then discussion the infrastructure level. So that's what's coming up.

## **Question 1 of 4**

AWS \_\_\_\_\_ analyzes video and can detect objects, activities, people, and celebrities.

* Rekognition  
  Correct  
  Rekognition is AWS's video and image detection service.
* Redshift
* VOP
* Lex

## **Question 2 of 4**

Using AWS \_\_\_\_\_ you can extract text sentiment values.

* Comprehend  
  Correct  
  AWS Comprehend is a natural language processor that can extract key phrases, language, and sentiment.
* NLP
* Redshift

## **Question 3 of 4**

Which API can translate using natural language processing?

* Translate  
  Correct  
  Translate currently supports Arabic, Chinese, French, German, Portuguese, and Spanish.
* Polly
* Transcribe  
  Incorrect  
  Transcribe is a predictive API that creates a text output from audio files.
* WAV

## **Question 4 of 4**

Which API can recognize human voices in sound files and translate that speech into a text output?

* Transform  
  Incorrect  
  There is no such API.
* Translate  
  Incorrect  
  Translate uses NLP to translate languages.
* Lex  
  Incorrect  
  AWS LEx is used primarily for conversational interfaces like chatbots.
* Transcribe

### **Understanding ML platforms**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] In this next section, we're going to take a look at platforms that are available when you're selecting AWSML for your particular business scenario. Now, in this drawing, we have a subset of the platforms that are available, and the reason is, as at the services level of AWSML services, there have been a number of new services that have been released within the past 12 months of the time of this recording. Same goes for platforms. In fact, there's so much activity here, that I'm actually splitting this into two sections in this course so that you can select, understand, comprehend, and choose the appropriate platform, if that's the level at which you want to work. To that end, I've separated out the Platform section into two subsections, and in this part of the course, we're going to cover what I call the serverless or the container-directed section of AWS machine learning services. 'Cause these, as you would expect, are mostly the newer services. So, there are three offerings here. One of them has been around for a little while, but we'll still cover it to be complete. It's the simplest possible partially managed or platform offering. It's called Amazon Machine Learning. It's kind of an unfortunate name, because it does not encompass all of the machine learning options that are available. It's really machine learning as its sort of simplest possible implementation. You can supply your own training data, which is the salient point, and then Amazon will actually analyze the type of data and select the appropriate model or algorithm, and then present you with the results. So, the concept here is that you have data, but you do not have technical talent on your teams who can work with machine learning. So, basically, it's optimized machine learning. Machine learning automatic, or for you. Now, if you don't have any use of predictive models, and you feel like you want to try it out, it's a great way to test. However, the implementation is lightweight, and it serves a subset of needs because of the amount of customization is pretty small. There are only two different models that are part of the implementation. But for some cases it might work, so we'll look at it. The second thing we're going to look at is the really, new, hottest thing that I'm most excited about. It's called SageMaker, and the idea is that you supply your data and pick or create your own model, and that the phases of model lifecycle are optimized around containers. It really reflects what's happening in the greater world of cloud computing, and has efficiency and economy built into it. So we'll dive into that pretty deeply. It's a brand new service as the time of this recording. We also, to be complete, could just use ECS or EKS, the new container services, but it's kind of a separate topic. Just pure containers would be differentiated from SageMaker in that you could host any container that you wanted. Of course, you would have to configure all the networking security and configuration to have the communication across containers, if needed, and so on and so forth. So, it's kind of a separate topic, but just to be complete, you could use your own containers. In the section subsequent to this one, we will look at platforms being hosted on managed virtual servers. And that would be Elastic Map Reduce or managed Hadoop and Spark, and in this case, Spark Machine Learning, as well. AWS Batch, which is managed EC2 spot fleets, and we'll take a look at vendor solutions. One that I've done some work with is Databricks, which actually runs, now, on top of either Amazon or Azure VMs. But the important point is it is a type of managed Spark, and for many of my customers, they want to look, when they're looking at virtual servers and Spark, at both EMR and vendor solutions. So, reflects the real world.

### **Understanding and using AWS Machine Learning**

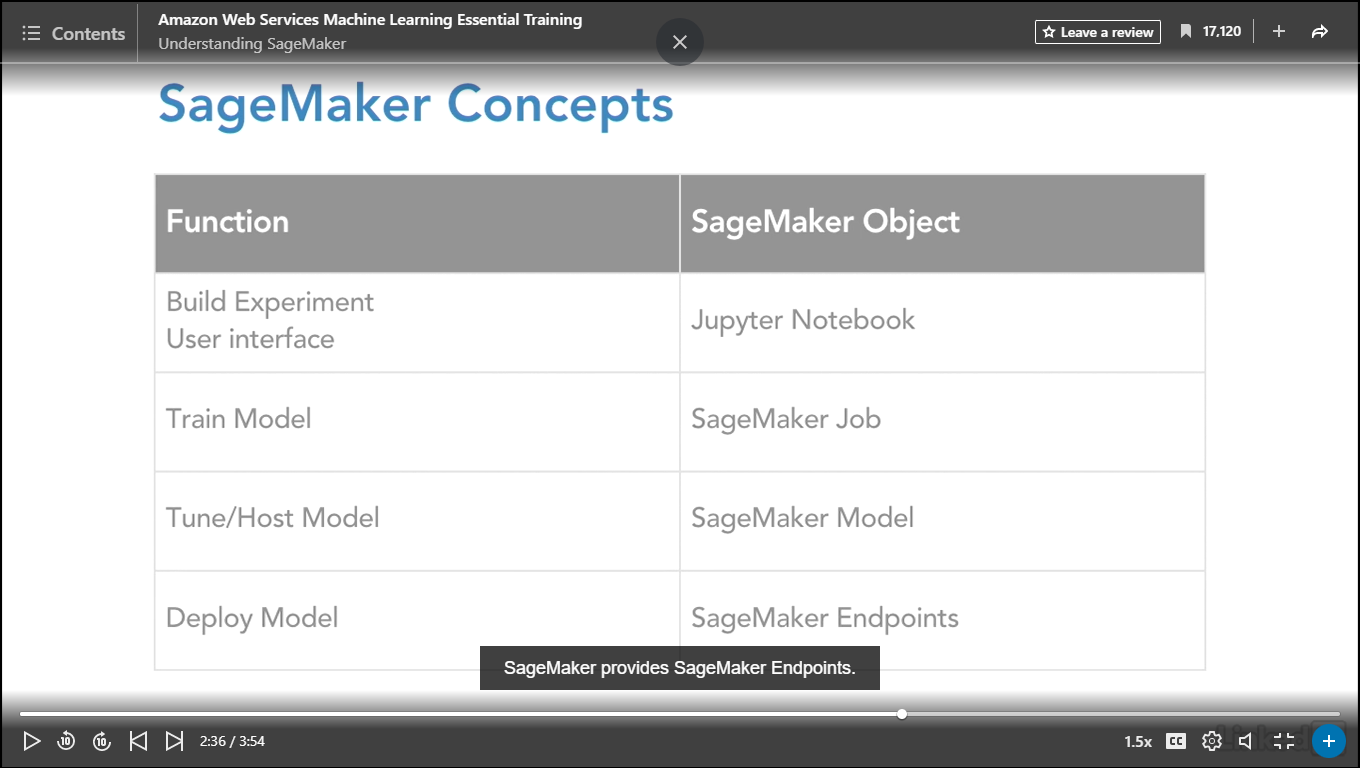
Selecting transcript lines in this section will navigate to timestamp in the video

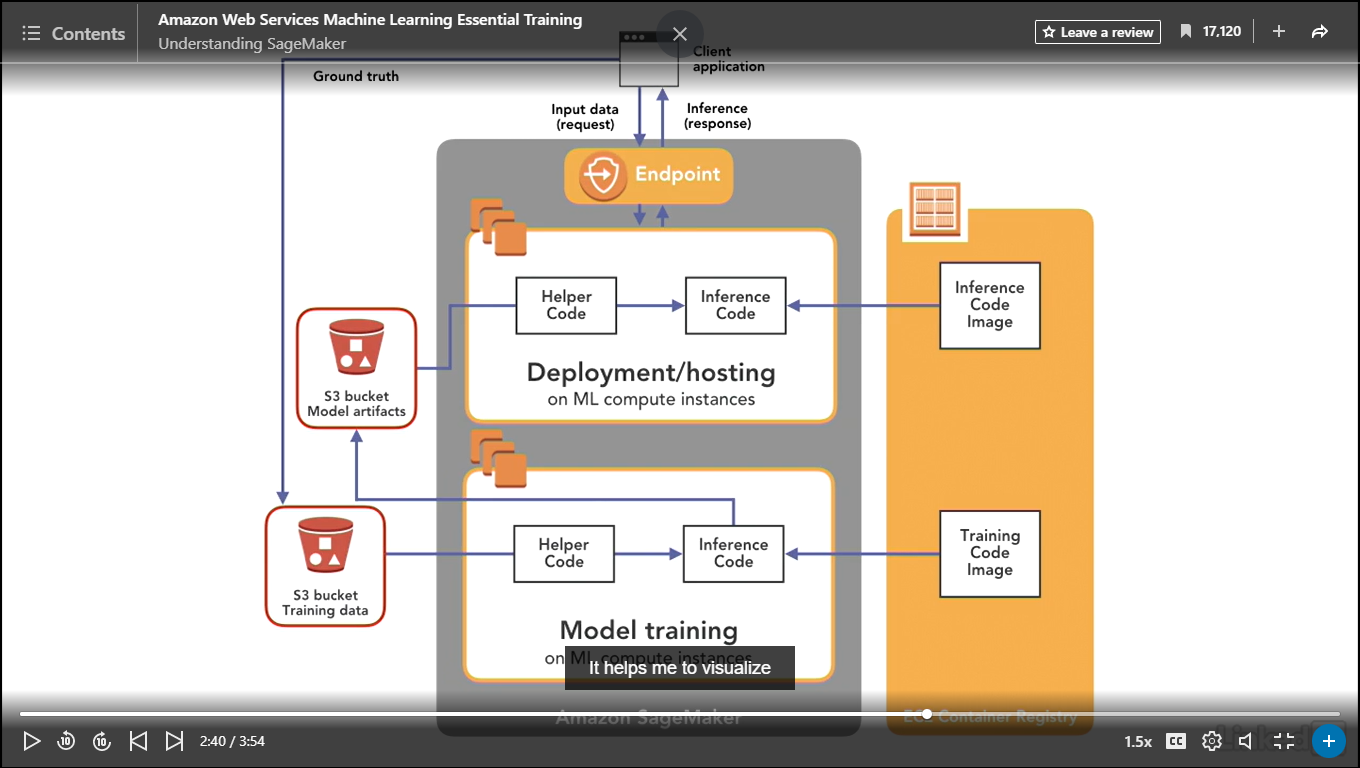
- [Instructor] We'll start by taking a look at the AWS Machine Learning Service. The service includes two prebuilt machine learning models. And it will suggest which of them or which algorithm to use based on the type of input data. It's integrated with S3. So here we are in the console, let's click Get started. We're going to click the blue Launch button. And we're going to use the banking data as our input data. Now I've already downloaded it and I'll show you what it looks like. You can see, it's csv and the top row is the column names. The value that we're trying to predict is whether or not each person in this list has gotten a loan in the past and we'll use that to create a model based on the attributes of the person in the list. Whether or not we should give a loan to a new person who's applying. So we're going to use Amazon sample data. And notice, you can pull data from S3 or Amazon Redshift the data warehouse. So we'll select this one and we'll call it banking. And click verify. You'll see that the algorithm detected the schema and created the schema file, let's click Continue. And here's the parsing of our information. So Amazon ML scanned the input and inferred the columns and the data types. Here, I can review and edit the data type to make sure there represents the data. And this is required for Amazon ML to read the input data correctly. So if I scroll down, I can see that that my choices for data types are binary, one or zero, categorical. I would group based on that, numeric, a number, or text. Just some text information that I would not group on. So let me just scan this and make sure it looks correct. I'm going to set default to text. Click on the next page and the rest click OK. I'm going to click Continue. It's informing me that we're using the sample banking data. I'm going to choose y as the target. When I generate predictions, get another dataset that doesn't have the information and that it will be predicted for me. And it's informing me that I've selected a binary attribute and y is the target, of course, it did for me automatically. Machine Learning Models trained on this target use logistic regression to train a binary classification model. And what that means in English is, logistic regression, regression means a line being the data and logistic means, a category. So is it a yes or a no. Binary means yes or no. It's a single category. The I click Continue and the data does not have an identifier. And I'm going to click Review. And then I'm going to click Continue. So now I'm in the world of my model. So I have six steps here. My input data, my model settings, my recipe, advanced evaluation, and review. The service sets the parameters by default. You can override that by selecting custom and then modify the training parameters. An example will be, how much of the input data you want to be used in training versus validation that's called a split. We're going to go with the default of 30% and click Review. And notice in the Recipe section, recipes help Amazon Machine Learning find patterns in the data. If you don't provide a recipe, Amazon Machine Learning will generate one for you. And then create model. And here we have a successful model creation. Now it will take a couple of minutes for this model to process so that we can see the results of the algorithm. And what we're going to be looking at is the predicted results using a split of the input data. So the input data, you remember is labeled. It says whether or not the person got a loan. So when you split the data, what's happening is, when the models being evaluated, the split data is being run against the model algorithm and the results are being calculated versus random guessing. And it will show you a graphical output and basically has a line through the middle of it and anything above the line is an improvement over random guessing. So it's called the AUC curve. So we'll evaluate quality of the model and then we can try realtime prediction. It takes a couple minutes so I'll start this up again once the model is created. Now that we can see that our model is complete, and we could download the log of the steps that were used to create it. Here's information about our data source. And here's the section we want to look at next which is the evaluation. And this is what I was talking about earlier in this movie, where in addition to processing the model, it's important you look at the evaluation of it. So how is this performed is by using the labeled data that was split and then running the algorithm against it. And looking at the results that the algorithm predicted versus the actual results, the true, the known good, or the known true results. And this is evaluated versus random guessing. So this is pretty good, 0.936 and this is called the Area Under the Curve. And to understand this, if we open this up under evaluation, we look at Summary. You can see it's telling us we're in the green. The area under the curve is 0.936 and this is significantly better than random guessing which will be of course 0.50. So the difference is 43% better. So what you can actually do in terms of tuning is you can ingest the score threshold by exploring performance. So this has to do with the grid of true positives, true negatives, false positives, and false negatives. This is called the score threshold. So it's actually easier to show so inside of here, you can see predicted to be zero, predicted to be one. And if we scroll down, we can adjust the slider to indicate how much error we can tolerate based on our needs. So what would be an example of this? Well, if we're doing some social media campaign, probably this is good enough. But if it's some dispensation of some drugs that are life-saving, then, 9% errors could be completely unacceptable. So how does this work? You can see if I scroll down a little bit, move the score threshold to the right will decrease the number of false positives but increase the number of false negatives. So if i move this score to the right, watch what happens, you see. And that adjust my trade off based on my threshold score. If I move it to the left, and you can see at some point, your value correct goes down, here it is. What was from 91 to 90 and the same thing for the 10% error rate but the composition of the errors is what's changing. And now we can go up, we go up to a certain point here. And we'll get passed, there it is. So it's within this range that we can adjust for true positive, true negative, false positive, and false negative for a particular use case. Now the last thing we can do is try a realtime prediction. We can try it out once we feel like the model is performing properly for us. So the way that we do that is we paste a record in. So if I click here and I got one in my buffer, and I click Submit, that will paste it in. And then, if I create a prediction, and you remember that the value is one or zero. We didn't do a translation to yes or no for the attribute that we wanted to predict. And so we can try this out in this interface of course you will most probably test this using the programmatic tools of the CLI or using the SDK so that you can evaluate it with multiple numbers of records. Then once you're done with the model and you feel comfortable with it, then you can integrate it using those same programmatic methods into your applications

### **Understanding SageMaker**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] As we get started working with our next service, AWS SageMaker, let's take a look at a typical machine learning model workflow. You can see in this diagram, there are three high-level steps. To generate example data, train a model, and deploy the model, and then there are substeps of fetching, cleaning, and preparing the data, training and evaluating the model, and deploying and monitoring the deployed model. SageMaker has components that fit with each of these parts. SageMaker is a relatively new service at the time of this recording. It's only been in general availability since this last year's reinvent, so three months ago, and it has four general steps. The steps are a component that supports building, so it allows connection to other AWS services, and transforming data using something called a SageMaker notebook. Then training. You can use SageMaker's algorithms and frameworks, or you can use your own for distributed training. Distributed is really key here. The underlying framework in the training component of SageMaker allows for very flexible and powerful scaling, which is really, really important as we're working with more sophisticated algorithms that are more computationally complex, such as the deep learning algorithms MXNet and TensorFlow, which are supported by SageMaker. The next component supports tuning, and the keyword here is SageMaker is automatically capable of tuning the model that you provide by adjusting multiple combinations of your algorithm parameters or hyperparameters, and this again is really significant in terms of getting your model deployed into your production system more quickly. Traditionally, tuning of hyperparameters has been a human activity, and I'm sure it will continue to be, but people plus computers tuning hyperparameters is going to result in a faster time to market. And then the last component is deployment. After your training is completed, your model can be deployed to SageMaker endpoints for real-time predictions. Now I'm going to map those functions to SageMaker objects or concepts. For the Build Experiment or User Interface function, SageMaker provides an Apache Jupyter Notebook. For the Train Model function, SageMaker provides SageMaker Jobs. For the Tune and Host Model function, SageMaker provides SageMaker models, and for the Deploy Model function, SageMaker provides SageMaker Endpoints. I always like to see a diagram. It helps me to visualize all the components working together, so here's one from Amazon around these SageMaker components. You can see on the bottom left, the training data is held in an S3 bucket. During the training phase, it's passed into SageMaker ML Compute instances that work with both helper code and training code. The training code is instantiated from an EC2 container registry from a training code image. Once the model is trained, the results are placed into a new S3 bucket. It's called the Model Artifacts. And then for deployment, you have new ML Compute instances that once again work with helper code, and this time, they work with inference code or prediction code, and this is generated from an inference code image from an EC2 container registry. Now, this can be Amazon's algorithms or your own. This is exposed to the client via endpoints, and then the client application, which is most typically a Jupyter notebook, but it's not required to be, is going to interact with the deployed model and then produce results that often are recycled and added back into the training data so the model continues to learn.

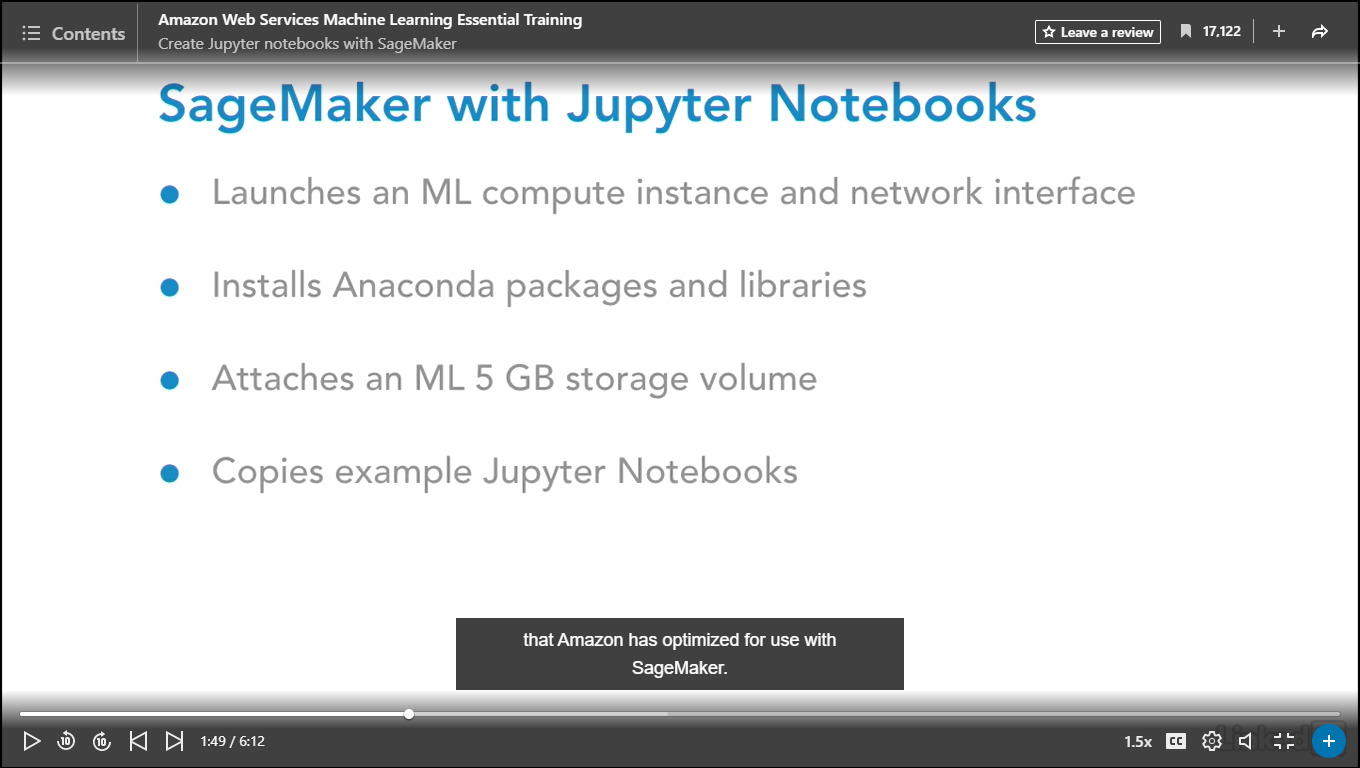




### **Create Jupyter notebooks with SageMaker**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] So as we get started working with SageMaker, we're going to work with Jupyter Notebooks. So if you're not familiar, these are based on an open standard, Apache Jupyter, and they're presented as web interfaces, so web pages. And they're designed as an alternative IDE, or integrated development environment. They are most commonly used by data scientists, or people working with machine learning. Why would you use something like a Jupyter Notebook? Well, I think of them as a more powerful terminal. You can document your data science experiment and you can document it very thoroughly, because Notebooks allow for you to work with text in the form of markdown, so you can annotate, or write in English what you're doing; code, code frameworks, and you can see the example Notebook I have here is set up to integrate with Python3, so you can actually run code, so like a terminal; and most importantly, visualizations. This is really key. When we're working with so much data and complex algorithms, having the visual component is a key aspect of both presenting our findings and also as we're working with the data in our experiment to understand what our data looks like. Some aspects of SageMaker Jupyter Notebooks are the following. When you create an instance of a Jupyter Notebook, SageMaker will launch a machine learning compute instance and it's associated network interface. It'll install Anaconda packages and libraries for a number of run times. It'll attach a five megabyte machine learning storage volume, kind of a scratch disk for you to work with your data. And it includes a large number of example Jupyter Notebooks with the included algorithms to help you to understand how to work with the algorithms that Amazon has optimized for use with SageMaker. So in order to work with Jupyter Notebooks, I'm in the Amazon consol at SageMaker. And I'm going to click on the orange Create notebook instance. I'm going to give this a name of Demo. And you'll notice when I scroll down, I can select from three different instance types. I've run this previously, so when I did that I created the IAM role by default. Now when you're accessing services outside of SageMaker, and that's typically data, which could be in S3 if it's files, or it could be in, for example, Redshift if you have data, warehouse data. You need to set up your IAM role with appropriate permissions for those external data stores. You can optionally associate your notebook with the VPC, and you can optionally encrypt. I'm going to click the orange Create notebook instance button. And now my instance is creating. I'm going to close this dialogue box, and you can see that the state is pending. It takes a couple of minutes for this to set up, so I'll come back once this is set up, and we'll open it and look at a Notebook. Now our instance is in service, so let's go ahead and click Open. And we're in our Jupyter environment. We're going to click the dropdown by New, and you can see that we have a number of run times or environments available here. We have connectors for Spark, so Sparkmagic for PySpark and PySpark3, and regular Spark and SparkR. We have the libraries for MXNet, 27, 36, Python2, Python3, TensorFlow 27 and 36. So that's pretty powerful in and of itself that we have all these run times available. So I'm going to go ahead and create a new folder, and rename it to Demo. And then open it. And inside of that folder, I'm going to create a new Python3 Notebook. And I'm going to rename this HelloJupyter. And I'll start with my code cell here. And then when I'm done with my cell, I can press Shift + Enter, or I can click this button to run any cells. And there's my code output. Now if I wanted to insert a new cell, I could say Insert Cell Above. And I can continue to work in Code, or I could switch this to Markdown. And then execute that by pressing Shift + Return, and that will render the Markdown. So when I'm done with this, I can work with it like a regular file, so I'm going to Save and Checkpoint. And notice that I can download this as an IPython Notebook, Python, and the rest of the file formats, as well. Now it's a best practice when you're working with Jupyter Notebooks, to save your work. Typically, integrate with your source code repository, so I work with GitHub or Git, and this is a really important best practice when you are working with your machine learning models, because then, of course, you have a history of all of the iterations. If you're new to Jupyter Notebooks, another thing that I want to point out is we have this Kernel menu item here, and in certain cases you're going to need to either Interrupt, Restart or Reconnect, so on and so forth. So you want to consult the Jupyter documentation to learn more about Notebooks if you're new to them. But it's just a tip from working with the real world, sometimes you need to clear out what is running in the Kernel in order for your Notebook to work properly, particularly when you're experimenting.



### **Get data with SageMaker notebook**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Narrator] So now that we've set up our HelloJupyter notebook, let's work with some of the sample notebooks that Amazon provides. So we're going to close this. And you can see by the little green icon that this notebook is running. So it's just a best practice to shut down when you're done. And then we're going to click on the folder to go up a level. And we're going to click into sample-notebooks. And we're going to click into sagemaker-python-sdk. And we're going to click into the first example which is demonstrating the Amazon built-in algorithm called kmeans. Kmeans, if you're not familiar, is an unsupervised, means it doesn't need any labeled data, algorithm that does clustering. It looks for what's called natural groupings in the data. So I'm going to click to open this folder. And I'm going to click to load the notebook. And let's start working with it. We do have to click on the Not Trusted button. Click okay. And there, our kernel is ready. So in addition to understanding how notebooks work, this is a well written, well formatted notebook. There is a syntax and a structure to notebooks that are used in the data science community so I do recommend spending some time looking at these examples. Amazon did a good job setting them up. So, you'll notice first here it has an index that's hyperlinked. And so this talks about what we're going to be working on, so a classification problem. This is a very common problem in machine learning. It's called the digits problem. So the idea here is we have images of handwritten digits from zero to nine. And we're imagining that the dataset doesn't have labels so we don't know what the true answer is. That's unsupervised. So we need to get the digits classified. We need to get them labeled. Are they a one, are they two, are they nine, whatever? So the common method is clustering. And we're going to use kmeans. So each point belongs to the cluster with the closest mean and the data is partitioned into number of clusters that is specified when framing the problem. There's 10 clusters. We have no labeled data so that means this algorithm is a good fit. It's really important, it's actually critical to pick the best algorithm or algorithms for the job. So if you're new coming into machine learning, read all these introductions when you're working with these notebooks. It's tempting to just skip down to the code sections. It's really important... You have to understand why the algorithms are being used before you understand how to use them. So we're going to scroll down. So we're going to set up the authentication to AWS services. So, we have a role and then we're going to have an S3 bucket. So we need a bucket name here. So let me go back and open up and get a bucket name. And I'll use demo-for-lynnlangit-jan. Now fill this in. And then to execute, shift enter. The next step here, we read the dataset from the existing repository into memory for preprocessing prior to training. We're going to use the MNIST dataset that has 70k 28 by 28 pixel images of handwritten digits. So they say you can do this processing by Athena, Spark, in EMR, Redshift, so on and so forth. The next step would be to transfer the data to S3 for use in training. For small datasets, we don't... Reading into memory isn't a big job. So, this is a Magic command. In a Jupyter notebook the percent sign percent sign and we're importing some libraries. And we're going to go ahead and load the dataset here. And you can see here we have the result. And once the dataset is imported, it's typical as part of the machine learning process to inspect the data to understand the distributions and this is the visualization I was talking about and determine what type of preprocessing might be needed. If you need to, what's called reduce the features, you have some features from machine learning are really columns from the relational data world. So if you want to remove some columns because they have data that's either extraneous or a lot of nulls or just want to make the model smaller so it's more efficient to process. So you can perform these tasks right here in the notebook. And as an example, let's go ahead and look at one of the digits that's part of the dataset. So here we're using matplotlib and we're importing that and then we're going to plot one of the values here. So we've got a method, show a digit, and from the training dataset, we're going to just show this digit. And there it's a three. So as I mentioned in an earlier movie, the data that we work with in this course is unusually clean and it's artificial, but it's so that we can focus not on data cleaning, which is what you have to do in the real world, but focus on learning the algorithms and the services. So one aspect of working with these sample notebooks is the data is exceptionally or unusually clean. And it's just a big difference from working with this in the real world. You could do something very quickly in these notebooks because you have clean data and you really want to have a sense of the data that you're working with and the quality of the data that you're working with because it can take a lot of time to clean it up properly and also, as you're cleaning through it, you may determine different patterns in the data which may lead you to using different algorithms. So, this step of ingesting and cleaning the data is very non trivial in the real world. For the purposes of this course, we're focusing really more on the next part which is setting up and training the model.

### **Train model with SageMaker job**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] All right, continuing with our example. Now that we have our data, we're going to train the K-Means model. And it says here since the data is relatively small, it's not really designed to show off the performance of K-Means training algorithm. But Amazon's reminding you, what they've done with these implementations is they have tested and they are telling you that they've got a system that's optimized to scale well with multi-terrabyte data sets. This is really a key reason to look at using the SageMaker Services because of this pre-optimization that Amazon has done. So the next step is setting the training parameters. Then we're going to start the training and pull for status until the training is completed. It's going to take between seven and 11 minutes. So you can see from SageMaker, so they've written an API here, Import K-Means and here's the data location, and there's the output location. And then they're just printing out a statement, the data will be uploaded to where and that training artifacts will be uploaded to the location. And then here's the meat of this, for the K-Means variable, to K-Means here are the parameters. You've got the role, that's the IM role, and the training instance count, that would be the number of optimized machine learning images, and then the instance type, which is machine learning instance types, c4.8xlarge, the output path, the K is ten, that's the number of values we're looking for, so classify by 10, because our digits are zero through nine, and then the data location. So we'll go ahead and kick this off by clicking in this aisle and pressing shift+return. And there it's giving us a little bit of information. We're using this time value here, and then for K-Means, we're calling fit and on K-Means for the record set, the training set, the first one. And let's run this. And then we'll return once this is completed. So it took eight minutes for this to complete. So you can see here's the bucket information and then if I scroll down here, we have a big log. So not going to take the time to go through all of it but you can see there's some red, and then there's some green, yours might be a little bit different of course, because your instance will have different names and that kind of stuff. Go all the way down here and you can see, Job Complete. And it took eight minutes, and then if you want to verify you can look in your bucket. So here's my bucket, so high level example, there's my data so there's the data. Okay, we are complete with this step and then the next step we can set up hosting for the model that we built.

### **Deploy and host model with SageMaker model**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] So I'm going to scroll down, and this tells us here we can deploy the model we just trained behind a real time hosted end point, and again, these examples are not optimized for production. They're just to get us started, so really don't take this as the performance of this service. Hence why Amazon's telling you this here. So we're going to go ahead and call deploy, and pass in an instance count of one, and there's our instance type, and then we'll click on the cell and press Shift + Return to execute that, and what's really kind of amazing about this is how much of the infrastructure Amazon has abstracted away. Now, while this is running, just to be complete, I'll show you that they actually do have, in their GUI console interface, the capability to click and do this. So let me just go back over here and you can see that we have instances and we have jobs, and of course, that's the job that we just ran. So that's model training, if you remember. If we click in here we can see information about it and we have some monitoring available. So the step that we're on now, of course, is models. So if we click into models, you can see that we've kicked this off. And how you choose to work is really what's comfortable for you. If you wanted to click in the console when you're first learning, you can certainly do that, but most of Amazon's examples will use the notebooks. So I find when working with customers, particularly if they have a background with Jupiter notebooks, they usually don't use the console that much. So it really depends on whichever one you're comfortable with. The examples are set up in notebooks. As Amazon has noted, this can take seven to 11 minutes to complete, so I'm going to go ahead and wait for this to complete and then come back. And we can see here, the output, that it took 11 minutes and this is done. And again, just to connect us back to the objects I'm going to go over to the console. If we start with the dashboard in SageMaker and we scroll down, we have one notebook that we're working with. We have one job, we have one model, and now we have one end point. And in the next step, we can actually use our model for prediction.

### **Use model from SageMaker endpoint**

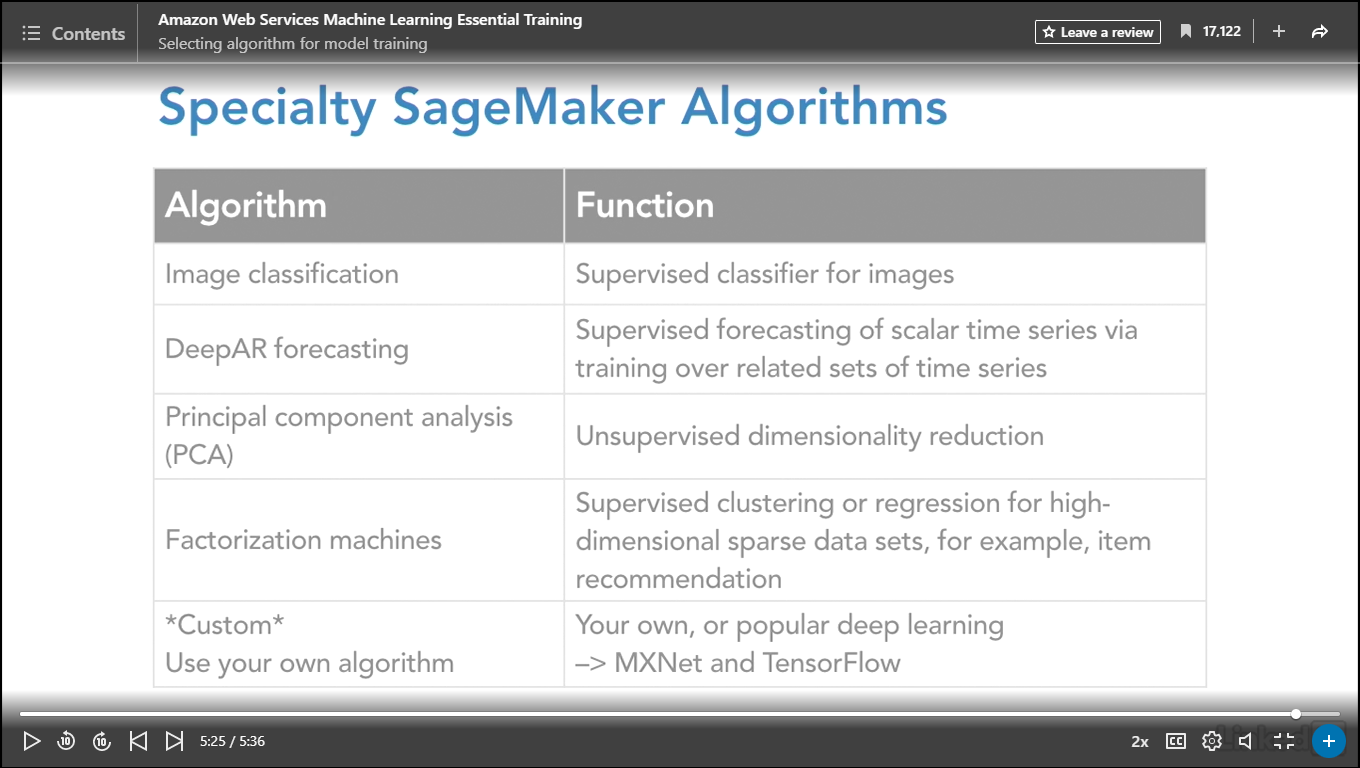
Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Now we're going to validate our model for use. We're going to generate a classification from a single observation from the train model using the endpoint. So you can see in this code cell we're setting a result variable equal to K-Means predictor calling predict against the training set and we're going to print the result. And I'm going to click in here and press shift-return to execute this cell. And you can see that we get this output in the format adjacent. The key is closest cluster. The value is a float 32 tensor and it's a 3, which is what we're expecting. And the label is the distance to cluster and the value is a tensor. This is the result that we expected. So a single prediction works. So next we can do a whole batch and see how well the clustering works. So we're going to scroll down and use this time value again and set the result to K-Means predictor predict from a set. And we're going to set a clusters variable equal to a label the closest cluster for r in the result. So let's execute that. And then for the cluster in range 10, because number we have ten digits. We want to print the output. And for the digits the image for 1 image in the zip. The clusters, the valid set of zero if cluster 1 is equivalent to the cluster. The height is the length of the digits minus one divisible by 5 plus 1 and the width is 5. Then we want to plot the parameters, the figure size, the width and height. And then we're using subsequent plotting here to show the result. And there's our first cluster. And our second cluster. And our third cluster. Our next cluster and our next one. Our next one. So algorithm is performing okay like you can see we got all the sixes there but we got them into two different clusters. So it's not perfect. Let's scroll up again to see. Yeah like this one is not perfect. But as it says here, K-Means clustering is not the best algorithm for image analysis but we do see reasonable clusters. So what's that really mean. What this really means is by using this process of a notebook we were able to quickly see. Yeah we got some result but it's probably not good enough so we probably need to use some other type of process and in particular we might choose to use a supervised algorithm which you might remember would be a combination of applying an algorithm plus some label data which is typically how this problem is solved. So again this is more an example to show how the process works with SageMaker than to show the more optimal result. In fact, Amazon takes this digits problem and it uses several different algorithms throughout its examples and its a great way to further your learning when you're working with SageMaker to work with some of the other notebooks that use the same examples so you can understand how you can use supervised algorithms that work with label data and compare the results. It's always a best practice to clean up when you're done. As it says here when you're ready to be done make sure that you run the cell below. This is going to remove the hosted endpoint and help you avoid any charges which is just the best practice. So click in this cell and press shift-enter. That removes the endpoint. And then you complete the deletion by clicking into the last cell and pressing shift-enter.

### **Selecting algorithm for model training**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Part of the power of SageMaker is the number of algorithms which Amazon has included. And a great way to explore them is to look at the samples. Many of the use the same example we had in our early movies of MNIST, the digits, and this is in Introduction to Amazon Algorithms. So you can see you've got, for example, you've got factorization\_machines, you've got linear\_learner, you've got xgboost. Now if you're new to this topic of algorithms, in addition to working with these notebooks, as foundational learning I would recommend that you work with the scikit-learn website. Now this is similar types of algorithms, but these are written really for university students and to run on laptops or desktop machines, but conceptually it can really help you, because as you can see, they're broken down to types of algorithms and you can compare types of algorithms, understand how they work. I've used it personally and it's really been helpful. So that we understand our algorithm choices with SageMaker, let's consider the four types. These are the built-in, Apache Spark, Deep Learning Frameworks and custom types. The built-in types, as of this recording, include 10 common machine learning algorithms and Amazon has worked really hard to optimize these algorithms to run on the SageMaker environment, so you should always start with those if they suit your problem. If you have existing Apache Spark code, for Spark or Spark ML, integration with SageMaker is also possible. And you'll want to read the referenced developer guide to understand how to set that up and make that work. Additionally if you have sets of problems that would warrant using the deep learning libraries TensorFlow or MxNet, integration is already included with SageMaker and we'll be looking at that in a subsequent movie. And finally, you can make your own using custom Docker images. You can create, package, and use your own algorithms. Another way to think about how to pick the best algorithm or algorithms for your particular case, is to fit your selection to your business situation and your data. So generally we have four things to think about here. Are we looking for our model to answer the question that is going to produce discrete categories? So yes or no? Is it a member of a class or not? Category one, category two, category three. It can be more than a yes or no. It can be many classes, like we had with our digits. Which of the 10 classes, or categories, between zero through nine did the drawing of the digit best fit? Another type of an answer is an answer which is quantitative. Which one? How much? So it's a number. What do we think the temperature will be in the future from our smart thermostat? How many of a type of car will we sell? At what price will a type of real estate listing sell? Yet another type is insight with no labeled data. And in fact, that was the example we worked with with K-means in the earlier movies of this course. Are there any natural groupings in the data? And you remember in the results, for some of the groupings, it got all of the hand drawn digits basically correctly, so all the eights together. But for others, sevens were mixed with maybe fours or ones. It didn't get all of them. And this is a simpler type of a processing, because we don't have to supply it with labeled data, a set of known truth or known correct data. In addition to that, there's a set of specialty algorithms and these support particular types of data, so processing of image data, video data, or text data. So optimized algorithms in SageMaker for discrete categories are K-means for unsupervised and that's the example that we looked at with the digits. And that means no labeled data needs to be supplied. Linear learning for supervised classification or XGBoost for supervised classification, which can be binary or multi-class, multiple categories, so A, B, C, D and ranking using gradient boosted decision trees. Algorithms for quantitative values are linear learning using supervised regression and XGBoost supervised regression. Now you might be surprised why linear learner and XGBoost can do both classification and regression. And it's a parameter setting. So again, you're going to want to read the documentation when you're working with these algorithms in production. There are a number algorithms for text processing and they do everything from finding topics in data to translating tokens for machine translation to translating words into vectors. And the final category of SageMaker algorithms I call specialty. So these include algorithms for particular data types, such as image classification for particular activities such as dimensionality reduction, PCA or principal component analysis is available for that. And to complete this section of course, you can use deep learning such as MXNet, TensorFlow, or even your own customized algorithm.



### **Advanced use of SageMaker**

Selecting transcript lines in this section will navigate to timestamp in the video

- In addition to what we've seen already around supporting the machine learning life cycle in Sagemaker, there are several scenarios for advanced usage. Now I just want to point out a couple of my favorites here. So, one of them is deploying multiple variants of a model to the same endpoint for A/B testing of different varieties or flavors of a model. It could be trained with different data. It could be set with different hyperparameters. So you can easily do that with Sagemaker. Additionally, you can update the endpoint by providing an updated endpoint configuration to change the machine learning computer instance type or the distribution of traffic among the model variants. You can also log endpoint access with CloudTrail. My personal favorite is using only the needed components of Sagemaker. And to that end, I worked with my college-aged daughter, who is using Sagemaker, specifically the component around hosted Jupyter notebooks, to help her with her bioinformatics research at university here in the United States. So she and I worked together and we worked with a use case where she was pulling a notebook that was done by some other researchers, and working with that notebook by quickly getting an instance of it up in Sagemaker. So she wrote this article talking about this particular use case. And this was from CSIRO Bioinformatics in Sydney, Australia and it was to support a tool in a genomic variant processing called GT-Scan, and this is an underlying article that that team wrote. So here Samantha was writing about how Sagemaker is providing scalable hosting for Jupyter notebooks, hence the Jupyter use case, but, in this case it's a notebook that she got from the other researchers and set up as a custom instance. So here she talks about how she did that, and then once she was working with the notebook, this is the original results, the visualization and then she added a visualization. So it was really quick for her to do. It took around 20 minutes. There was nothing to install on her laptop and when she was done working, she was able to just turn off the instance. So it was really a great use of notebooks as a service and this functionality I think has a great amount of usage from a large number of researchers worldwide, so I wanted to share it here.

## **Question 1 of 5**

Jupyter notebooks allow you to work with text in the form of \_\_\_\_\_, so you can annotate or write in English what you are doing.

* in-line  
  Incorrect  
  In-line is not used in this context.
* markdown  
  Correct  
  This can be incredibly helpful.
* open source

## **Question 2 of 5**

SageMaker supports \_\_\_\_\_ model testing.

* A/B  
  Correct  
  You can deploy multiple variants of a model ot the same endpoint for A/B testing.
* B/A
* neither of these

## **Question 3 of 5**

To predict a value you use a \_\_\_\_\_ algorithm.

* Regression  
  Correct  
  Regressions are used to make predictions of values.
* Classification  
  Incorrect  
  Classification is way to determine categories, not predicting values.
* Clustering
* Deterministic

## **Question 4 of 5**

What kind of predictions can you run with SageMaker?

* none of these answers
* single prediction and batch  
  Correct  
  SageMaker supports both kinds of predictions.
* only batch
* only single prediction

## **Question 5 of 5**

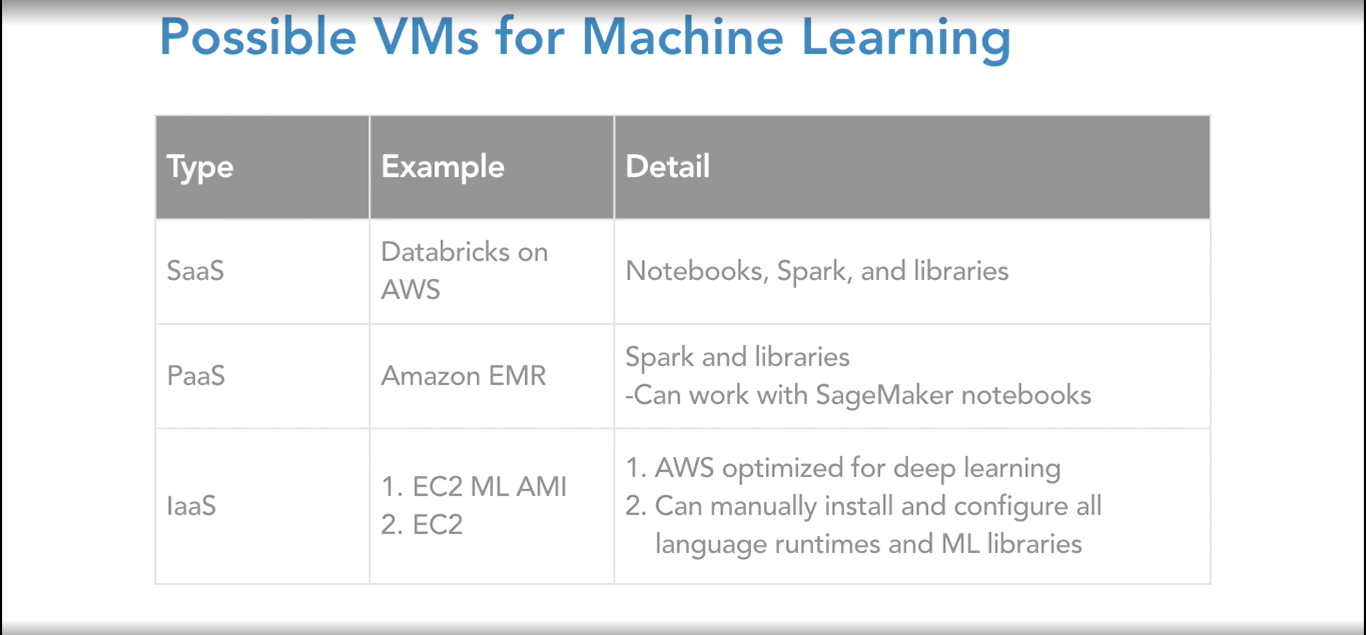
You can use your own \_\_\_\_\_ machine learning algorithm with SageMaker.

* linear regression
* supported
* custom  
  Correct  
  This is a wonderful feature offered by SageMaker, enabling developers to create, train, and deploy machine-learning models.

### **Understanding ML virtual servers**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Narrator] In this next section, we're going to address the question, "Should I use a server, virtual machine or, actually, a server cluster for machine learning?" It's really a key question and it's quite timely as the vendors, including Amazon, are upgrading their services to reflect changes in underlying compute options. What do I mean by that? Well, I started the course not with servers, which would be typical for machine learning a couple of years ago, rather with APIs. We looked at just working with endpoints and being charged by the service call for things like automatic image labeling and text-to-speech translation, common machine learning tasks these days. Next, I looked at SageMaker, which is a scalable set of componentized services that you can use in any combination to suit you. Of course, there still is a lot of server-based machine learning work out there, but my point is to get you thinking and to consider using some of the newer options because they can allow for a lot more flexibility and agility and help you get your machine learning models to market and to provide business value much more quickly. That being said, let's look at servers. I have four different ways to work with virtual machines in the Amazon ecosystem for machine learning. The first is software as a service and I'm going to highlight a vendor that I've done lots of production work with. They're called Databricks. It's a third party service, so you pay them and then you access underlying Amazon resources. And they are well-known for their implementation of Spark as a service. Their implementation includes their version of a notebook. It is a Jupyter-like notebook, but it's a Databricks notebook. A optimized addition of a Spark cluster and the ability to install additional libraries. So we'll be looking at that in this section because the underlying technology is servers even though the management is very much abstracted away which I see as a positive. The next level is Platform as a Service and Amazon's offering there is Elastic MapReduce, which is managed Hadoop and Spark. It comes with the ability to install common libraries, such as Spark and Hive and Pig and other types of libraries, just by clicking when you install when you're setting a flag if you're doing it via script and you can also optionally install additional machine learning libraries, such as TensorFlow and MXNet with bootstrap actions. Interestingly, Amazon has already installed, in SageMaker notebooks, the environments, they're called Sparkmagic, so that a connection to an external cluster of Spark, including EMR, can be easily made. Now the third possibility is infrastructure as a service. This is a little bit tongue-in-cheek. Most people would say, "Well, you can have an EC2 machine learning or deep learning AMI or image or you can just use EC2." And, yes, I think you can use a machine learning AMI. It's optimized for deep learning and all the libraries are already pre-installed. I actually would not recommend you use EC2 because you must manually install and configure all the language run-times and machine learning libraries and I have seen this task take people days or even weeks to set-up at a cluster of machines. We've really progressed beyond that state and there needs to be a really compelling reason for you to consider, so much so, that I actually updated the slide and the way I recommend to my customers, the three options are use the Amazon optimized EC2 image, use EMR, or use some vendor implementation. Databricks is the one I typically use, but there's lots of them out there.



### **Understanding deep learning**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] So when I'm working with my clients in trying to figure out whether or not using server clusters is the right approach for working with machine learning, there are a couple of considerations. One is the size and complexity of the input data. And the second is the complexity of the algorithm or algorithms that are going to be used. Very commonly, if customers are interested in trying out deep learning, we'll use server clusters. So what is deep learning? And how is it different from traditional machine learning? If we consider the use case of trying to build a model that labels which animal is in a photo, the cat or dog, sort of classic use case, in traditional machine learning, you'll use a classifier algorithm. But that classifier needs a whole bunch of trained or labeled data. And that has to be done by people, and that's called feature extraction. This process of creating this trained or labeled data can really slow down the building and implementation of the model, because, of course, people have to sit there and manually label the photos. There could be errors. This takes time, someone might need to review it. And so on and so forth. How deep learning differs is the feature extraction process is handled by the algorithm itself. Now this might seem like magic, but it's not. But it can be magical in its business results if it works properly. Because what it does, is it speeds up the time to market. That's why there's so much interest in it. Let's look at an example to understand more deeply. In this example, we're performing analysis of people spaces. So what happens first is the input information is encoded. In other words, the picture information is translated into a set of information that can be computed against. Some numbers, basically ones and zeros that represent the pixels and how the pictures are set up. Each circle represents a unit of computation. So we start with the input layers, and then we have a number of what are called hidden layers that perform mathematical computation to find patterns in the information. And then the result is an output layer. So the hidden layers are really the magic, or the secret sauce, of deep learning. Now in order to work properly, there are a set of inputs that can be tuned, and these are called hyper-parameters. And the output of deep learning is that deep neural networks learn hierarchical feature representations.

### **Work with Gluon for MXNet in SageMaker**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] In order better understand deep learning, we're actually going to jump back into the world of SageMaker, and look at a sample notebook. We're going to click on sagemaker-python-sdk, and we're going to click on mxnet\_gluon\_mnist... and open up the notebook. So this is our same example that we used earlier, but with a different type of an algorithm, and I think this is great for comparison, so you can either review, or think about when we worked with the K-means algorithm, and you can contrast that in working with MXNet, which is one of the many popular deep learning algorithms. Now we have a little bit different flavor to this, because there is a lot of complexity to working with this, Amazon has actually written a language called Gluon, which is a higher level language, which reduces the amount of code that you have to write to work with a deep learning network. So we're going to start with our example with Gluon, and then subsequently we're going to look at the more low level MXNet, so you can understand if you need to do model tuning or manipulation, how you can access it at that level. So let's start with Gluon. So we know what the dataset is. It's the same 70,000, 28 by 28, grayscale images of handwritten digits, and the dataset's split into 60,000 training images, and 10,000 test. There's going to be 10 classes, 'cause it's zero through nine, and this is going to show us how to work with Gluon. So, here's our import statements. We're going to import SageMaker of course, and then bring in MXNet, and then Gluon, so this is just set up. And we're going to click into the cell and press shift return to execute it. And then we're going to download the test and training data. And then we're going to use the SageMaker session upload to upload our data into an S3 location. I'm going to wait for my notebook to catch up here. All right, now we're going to implement the training function. We need to give a training script that can run on SageMaker, and we should be familiar with this from when we ran through the K-means example in earlier movies. So we're just going to cat this file so we can see what it looks like. So here's our training function. Here we're setting the context to CPU. You also could use GPU, or graphics processing units, and we'll discuss that in more detail in the subsequent movie. We're setting the hyper parameters here. How big the batch is, for example. We're loading the data, defining the network. Collecting the parameters. And then starting the training. So we have all this training that we're running. Then we're training and validating. We have our test data. Here's our hosting methods. So we've got it all together in one script. Again, we're going to go through this in more detail subsequently. So you can see that we've created an m, a variable, to pass in the parameters to run our script, and set up our machine learning instance of c4 size, and setting our batches and so on and so forth. So we'll run this. And then we'll fit it using the data we've uploaded to S3. Now our training is complete. And we're going to use the MXNet object to build and deploy and MXNet predictor object, and that's going to create a SageMaker endpoint, and we can perform inference then. We're going to run this cell. Now it took a bit of time to setup the endpoint, and again, I just used the console, 'cause it was quick, and you can see it needs to be InService. Think it took about ten minutes there to set up. So, it's just a quick way, when you're running this demo, you can switch back and forth to the console. So now we can use the predictor to classify handwritten digits. Drawing into the image box loads the pixel data into a data variable in the notebook, which we can then pass to the MXNet predictor. So let me draw a four. And the predictor runs inference on our input data and returns the predicted digit as a float value, so we convert it to int for display. And it's a four. So, it's not obvious in looking at this implementation why it would be superior to K-means. You'd have to subsequently test it with a batch. We basically just got it to the point of making sure that it could identify a digit, but the next step that you would do to compare algorithms is you would compare MXNet with Gluon, to any other algorithm you might be working with, such as K-means, or anything else, and look at the percentage correct. Now one thing that you'll want to do is, you'll want to run the cleanup so that you don't incur excessive service charges from Amazon if you're just testing. So you're going to want to run this delete endpoint when you're done testing.

### **Work with MXNet in SageMaker**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Next, for comparison, in SageMaker in the sample notebooks, we're going to run the mnist sample using MXNet. So, again, it's going to be the similar kind of input dataset. It's the images. Handwritten digits. We have 70,000 of them. So, we need to define a few variables. And we're going to have to use a bucket for this. So let me get a bucket-name. Of course your bucket-name will differ, depending on which bucket you decide to put this in. So we've got two bucket locations. One to save the custom code in a tar format. And one where the results of the model training are going to be saved. And then we need an IAM role. And we're going to click in the cell and press Shift + Return to execute it. And then we're going to use a script as we did in the previous movie. So we're going to cat that script so we can read it. And we can see here we're using MXNet itself. We're not using Gluon. So it's a lower level. So we're loading data. We're finding the file. This is a key section right here. We're building the graph. So this is where you have access to constructing the actual graph itself. And this is really for a small subset of situations. One of the reasons I actually decided to make this course is 'cause I see too many machine learning courses that start with creating a graph with a deep learning language, such as MXNet or TensorFlow. And while they certainly have their place and they're very powerful, to me this is sort of at the opposite end. It's after you've exhausted using all the APIs and all the pre-built stuff because you really have to have a very, very deep grasp of the mathematics behind this, which is linear algebra and calculus and mathematics of that level in order to be able to translate this programmatically. That being said, if you have that and you need that, this is how you do it. You're creating layers here, basically. So you have fully-connected layers, activation layers, so on and so forth. Then here's your training. As I mentioned earlier, it's very common to optimize for GPUs because the price of GPUs has been coming down and it's a way to scale out the computation and deep learning algorithms are actually written to take advantage of GPUs. So it's a hardware and software optimization that is speeding implementation and causing customers to take a look at this. So now we'll look at the MXNet estimator class. So we can run a single machine or distributed training using CPU or GPUs. And, again, this is very advanced, but when you are needing something this sophisticated, of course you want to use the appropriate hardware. So you spend appropriately in terms of service cost and you get the results back quickly. So, when you create the estimator, we pass in the filename of our training script, the name of the IAM role S3 location, we provide a few other parameters. The train\_instance\_count and the train\_instance\_type. So this is familiar territory here, and because we're just testing, we're only passing in one and in a machine-learning optimized instance that's m4.xlarge. And then we certainly, if we were in a production situation, we would be using more instances. So we're going to set this up. And then we're going to run the training job. So, after we've constructed our MXNet object, we can fit it using data stored in S3. We're going to run SageMaker training on two inputs, train and test. During training, SageMaker makes the data stored in S3 available in the file system while the script is running. The Python script simply loads the train and test data from disk. And, again, I assume this is going to take several minutes because we're only using a single instance. So, once this is completed, hopefully this is becoming familiar now, the next step is we'll be creating that inference Endpoint. So really when we're looking at this, what's different between this instance and the previous movie? It's how the model was constructed. So, you're going to use the right tool for the job. If you have a situation where you need to construct the layers of the model because, for example, you're not getting out the quality of model predicting that you're needing for your business, then you may need to go to a lower level to do adjustments to the number of layers, the hyperparameters. However, if you just want to work with a deep learning network, higher-level languages like Gluon for MXNet and there's a similar one for TensorFlow, it's called Keras, are really recommended so that you can get started leveraging the power of these deep learning networks without having to get all into the kind of nitty gritty of it. I like to say, MXNet and TensorFlow are kind of like C++ in terms of object-oriented programming. There certainly are situations for which they're the best fit. But, just like you probably wouldn't use C++, most of us, anyway, to write a user interface, you shouldn't use MXNet and TensorFlow to solve every machine learning problem. It's really a key takeaway of this course. You want to use the right language, the right tool, at the right time so you can get business value. So I'm going to wait for this to get set up and then come back and complete the example. All right, our training job is done. And, again, let's use our validation practice of going over to the console, refreshing, looking at the dashboard, scrolling down. And then we can go into the jobs. And we can see that our job's completed. So now we're going to go back to our notebook. And the next step is going to be to create an inference Endpoint. So this is similar to what we saw with Gluon on MXNet. After training, we're going to use our estimator to build and deploy a Predictor. And that's going to create the Endpoint. And, of course, the arguments we pass are going to set the number and type of instances for the Endpoint. So let's run that. So now our Endpoint has been deployed. It took 10 minutes. You can see that the request-handling behavior of the Endpoint is determined by the script. In this case the script doesn't include any request-handling functions, so the Endpoint will use the default handlers provided by SageMaker. These default handlers allow us to perform inference on input data encoded as a multi-dimensional JSON array. Now we can make an inference request. And, in our case, this is going to be, the, hopefully, now familiar, classification of handwritten digits. So, the same process as before. Let's draw a different number this time. Let's draw an eight. And then, let's look at the prediction. You can see the raw prediction results include more data, but the most likely answer is eight. So it did get it right. Again, just like with the example that we saw with Gluon with MXNet, and, much earlier in this course, with k-means doing the same prediction problem, we would have to input a batch of data and then we would have to look at the percentage correct of the results across the different implementations of, in this case, two implementations of MXNet and then two different algorithms, k-means and MXNet, to determine which one was the better fit for our business situation. A final thought in working with MXNet. It is of note that I chose to work with SageMaker. And you'll see, when we work with some of the servers, that the setup that's involved, or, actually, not involved, in SageMaker because I simply could just use the notebook, the environment was already loaded. And you can see that from up here. This is part of what you get with SageMaker. It can really help you to iterate quickly through different algorithms, which is becoming increasingly important as there are more and more algorithms available for use. So don't discount this approach of using SageMaker when you're early in your model development, rather than setting up virtual machines. Virtual machines really are for production, in my opinion because of the amount of time and overhead and cost to set them up and maintain them. Speaking of cost, let's remember to clean up by deleting our Endpoint.

### **Databricks on AWS**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] So to get us started with working with clusters of virtual servers for machine learning, we're going to start with the software as a service or the vendor that I really work with most frequently, which is Databricks. Databricks offers Spark as a service, or Apache Spark. And the reason for this is that most of the committers to the open source Apache Spark set of libraries actually work at Databricks which is a commercial company. And they provide optimized versions of server clusters as a service for their commercial offering. So, you can select Databricks on either, now AWS or Azure, but we'll be focusing on AWS for this course. So their offering is a set of services that includes both sample notebooks, And their notebooks look like Jupyter Notebooks, but they're actually not Jupyter Notebooks. They're proprietary. And that's kind of important to understand. So the notebook is the interface and on the underlying server mechanism you still have some access to, although they optimize it and you'll see this in the interface. There are a range of operation supported including several flavors of machine learning. And one of the reasons it's fun and productive to work with Databricks, it's a very quick setup and you can try out a big variety of different libraries on different-size server clusters, which then can be quickly put into production. So it allows for quick time to market where you're processing large amounts of data. So, as I said, they support many types of machine learning, and machine learning code. So they support R. They support Spark, of course. Spark machine learning, which is a library. They support deep learning. And we're going to actually use deep learning. We're going to use MXNet just to continue our example, because I think it makes for a nice comparison to see MXNet across the different platforms. Also, conveniently, in addition to their commercial offering which you have to pay for, Databricks has a free community edition, which makes trying out working with their platform very, very convenient and easy. So, we're going to do some setup steps in this movie, and then in the next movie, we'll be working with MXNet. So let's move over to the setup steps now. So, I've already signed up for an account and your account might look different because I've dones some other work in this account. So when you look at my folder tree, it might look different than yours. And then I wanted to go into the documentation before we got started setting up for MXNet. So you can see, in the documentation, just for informational purposes, they support a lot of different libraries for machine learning and it really makes the platform very compelling for the different machine learning use cases. So if I open up the machine learning section here, you can see that they've got a lot of different examples and they have a lot of different example notebooks, similar to what we saw with SageMaker from AWS. The difference here, of course, the underlying mechanism is virtual servers, which you'll see when we set up the cluster in just a minute. And you can see for these various examples, if I click into, for example, binary classification, they have some great use cases with public data. So, it's a fantastic way for your own learning and studying. I met a lot of people that have worked with Databricks from a learning perspective and then have developed models that they were able to put into production for their business. So, really, really great platform for that. We're going to work with MXNet. So, what we're going to do is, we're going to grab out their notebook. So we're just going to scroll down here. And their general pattern is they have their version of the notebook and they'll usually put a link on the right side. So I'm going to copy a link to the notebook and import the notebook. And I'm just going to copy this 'cause we'll need to have this for the environment. So now, back in the environment, I'm going to open up the tree, and I'm going to make a new folder just for our work today. I'm going to call this demo\_mxnet and Create Folder. And then inside of the folder, I'll click the dropdown and I'm going to say Import from a URL the sample notebook. So you'll see up at the top it says Detached. And that means that we don't have a running cluster. So, this is a little bit different than what we saw with SageMaker for example. So, here we have to set up the cluster. There's not really that many steps to it, but we still have to do that. So let's go ahead and do that. Now, before we set up the cluster, in order to work with MXNet, MXNet is not preloaded. So what we actually have to do, is we have to set up a library. So I'm going to actually go back over here and into my folder. I'm going to click Create and say Library. And then the type of library is a Python library. And it's MXNet. And I'm going to click Install. And I'm going to make sure that this checkbox, Attach automatically to all clusters, is set up. And that way, when I create a cluster, this will attach automatically. So now I'm going to go back to clusters. And I'm going to create a cluster. I'm going to call it demo-mxnet. And I'll just accept the default version and I'll click Create Cluster. Now, for the community edition of Databricks, you can only have one live cluster at a time. And the compute part of the cluster, so the virtual machines, only stays up for a certain amount of time. I think it's about an hour, if I remember correctly. So this is really just for prototyping. Now, for the commercial version of Databricks, you're paying by the DBU or by the compute unit, and obviously your cluster then can be persistent and you can set up a cluster with a different number of worker nodes. You can take advantage of Amazon pricing optimization, such as spot pricing. Here they're showing you you have zero spot nodes in this particular cluster. And you can see the cluster is now running, but I need to check to make sure that the library is attached. I need to go in and make sure that my library is attached or we're not going to be able to use MXNet. So, I have some other libraries. Your screen would look different than this because I have done some other prototyping with some other libraries. But what we're looking for here is MXNet. So, that looks like it's there. Let me just check it. Yup, there are six. It just took a couple seconds to come up. And now I'm going to go back over to my notebook and I'm going to attach to my running instance. Now our cluster is set up and attached

### **Work with MXNet in Databricks**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Now an important consideration as we get started working on our machine learning problem here with MXNet is that because Databricks is designed for production and MXNet is designed to work with GPUs, or graphics processor unit type processors, this notebook is optimized for GPUs. Now our cluster does not have GPUs on it, nor can you get a cluster with GPUs in the community edition, so there's one small change that I'm going to make so that we can actually just run this. It's not going to be optimally fast because it's designed for GPUs, but at least we can see the functionality here. We know the problem now, the handwritten digit recognition and what we're going to do is get the data, so we're downloading the data again, this is nothing really new. I'm just going to go ahead and run this now. This notebook is a little bit different. We can do the Shift + Return like a Jupyter Notebook or you can use this little run button or there's keyboard shortcuts as well, so I'm going to just click run and you get a progress bar here which is kind of nice actually. Then we're going to plot the first 10 images and print their labels. There's our first 10 handwritten digits and then we're going to print their labels because we're using label data, of course, because this is a supervised algorithm. Now we're going to create data iterators for MXNet and this is similar to a regular iterator. It runs a batch of data in each next call and the batch contains images with the labels, and the images are stored in a 4D matrix with the shape batch size number of channels width and height. For our data set there's only one color channel and both height and width are 28 because of the size and they're mentioning that we shuffle the images used for training which accelerates the training process. Now here they're giving you a big warning: If you run this notebook on a non-GPU machine, then you have to make sure to install the correct package. Remember we installed the regular MXNet PyPi package as our library, so if you were running this with a GPU enabled set of machines in a cluster, you would run this library instead. Here we're working with MXNet and we're running that code. I particularly like the explanation of the deep neural network in this notebook which is one of the reasons I wanted to show it to you because it kind of is a dual purpose, we can look at Databricks and we can get a deeper explanation as we continue to work with this problem. Here's the math behind it, and this is the good all line formula so multilayer perceptron contains several fully connected layers and you might remember the layers in the initial explanation of deep learning where we had the multiple layers identifying the faces. Again, this is the math behind it so I'm not actually going to go through all the math, but it's nice that it's there if you wish to read it. What we're doing here is we're creating a placeholder for the input data and then we're flattening the data on line five from a 4D shape into 2D and then we're working with our layers. This is really basically identical to the code that we saw in a previous movie where we were running MXNet on SageMaker. The difference here is on SageMaker, you don't deal with the underlying compute resources, they're just spun up automatically, whereas in Databricks, you do set up the cluster. So I'm not saying one approach is better than the other, it's nice to have choice, but it's interesting how their approaches differ. Here we're setting up the layers in line eight, line 10, line 13, line 14, line 17, and line 19, so we're setting up our model. Now we can start training. Notice they're telling you again, "We recommend using "a GPU or a large CPU instance if needed." In our model, we have 10 data passes and the learning rate for us to cast a gradient descent is .1 which is a hyperparameter and then we have our fit which has got our training data and our validation data and we want to see output progress for each of the 200 data batches, that's our callback right here. Let's run that. We have some deprecation warnings because of the MXNet API is relatively new and so there are some updates to it, but it's still going to run, it's just deprecated values because the model code in this notebook could be updated to some of the newer API implementation. This is done. Then after training, we can predict a single image. Let's see what we predict on that, we're hoping a seven, right? 99% seven. You might be asking yourself, "Why did this go so much faster, this 10 seconds, "than on the SageMaker interface?" Well again, in all of these demonstrations, I'm not really optimizing for speed, it's more understanding functionality. There's a whole set of considerations around optimizing for speed, so really don't take this as important information, it's really more understanding the mechanics and the possibilities of the different platforms. Each of them can be optimized. All right, and we can look at the accuracy by a given data iterator. 97%. All right. Now we're going to look at a CNN, or a convolutional neural network. I really like some of the illustrations in this notebook, it's one of the reasons I wanted to share it with you. It tells you in the previous fully connected layer, it simply reshapes the image into a vector in training. It ignores the spatial information that pixels are correlated on both horizontal and vertical dimensions. The convolutional layer, the additional layer, aims to improve this drawback by using a more structural weight instead of simply matrix to matrix multiplication, it uses 2D convolution to obtain the output. And so here it's showing you some visualizations of what it's doing which again, I think are really well done. Besides the convolutional layer, another major change of the convolutional neural network is the adding of pooling layers. A pooling layers will reduce an N times M, often called kernel size image patch, into a single value to make the network less sensitive to the spatial location. It's a fancy way of saying reducing it. Again, I really like the illustration here because it shows you it picks the highest value in this case. Here we're again constructing our model with all of our various layers. You can see we've got layers on three, four, five, seven, eight, nine, 11, 12, 13, 15, and 17, so really granular level of control. Let's go ahead and run this. The explanation here, I think it's a bit ironic. They say, "Note that LeNet is more complex "than the previous multilayer perceptron, "so we use GPU instead of CPU for training." Again, this notebook is written for GPU and they're giving you the big warning. So the simple thing here is it's looking for a GPU, so if you're in a CPU, you just change it to CPU basically. Then you're going to go ahead and run this. All right, you can see that the training is done, and that took almost 10 minutes. Now let's work with the validation accuracy and see how we do. 98%, not bad. We close up. The comment that "With the same hyperparameters, "LeNet achieves 98.9%," well in our case 98.8, "validation accuracy, which improves the previous "multilayer perceptron accuracy of 96." I think we got a little bit higher than that, let's look. We got 97, but still, 98.8 versus 97 is an improvement. Again, this is an important concept of comparing different implementations of models and this is a use case when you would want to work at the MXNet level of this particular algorithm so that you could have access to hyperparameter tuning and to working with additional layers and constructing your model to improve the accuracy.

### **Set up the AWS Deep Learning AMIs**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Narrator] Next step we're going to look at AWS deep learning Amazon machine images running on EC2 or AMIs. And as mentioned previously the reason that these are so great to use is because they come pre-installed. There's three versions. Conda AMI that has preinstalled pip packages for deep learning such as TensorFlow, MXNet, Gluon, and many more. The base AMI which is a clean base image. And an AMI with source code so the deep learning packages and their source code. Now in addition to selecting the AWS deep learning AMI, typically when you're using EC2 you'll consider using GPUs or graphics processor units. And the reason for this is that most deep learning algorithms are optimized to take advantage of the parallelism available via GPUs. The chart shown here shows a performance analysis conducted by an independent group and there's a reference to the blog post. Showing the speed up in model training. It's really significant. To launch our instance we'll go to the console and click launch instance. Click AWS Marketplace. Type deep learning conda linux. And we'll select this instance type. This is information about pricing. We'll click continue. We'll scroll down to see the default selection. The default selection is a p3.2xlarge and this has GPUs associated with it. We really don't need something that extensive so we're going to go with the smallest possible GPU enabled instance which is a p2.xlarge. When working with any GPU instance you'll need to verify that this instance is available for your Amazon account. If you try to use it and you get an error, you'll just need to call Amazon so that they can enable your account for usage. I'm going to click review and launch and then click launch. And select a key pair. Now because this takes a couple minutes to launch, I've already launched another instance but these are the same steps basically. And here's my p2.xlarge instance. So I'm going to connect to it. Now I'm using a Mac and so for connecting I've already run the chmod on my key. Your key name might be different. And I'm going to connect to my instance using the ssh command, again yours will be different. Also I want to point out that you need to change root to ec2 user. So here I am in the terminal. And I'm going to connect to my instance with the ssh command passing in my key and the ec2 user and the IP address of my instance. I'm going to scroll up so you can see here I'm in EC2 deep learning AMI and the first thing that I need to do is I need to select which environment I want to work with because as mentioned this instance comes preinstalled with a number of environments. So these include MXNet with Keras, Python two or three, TensorFlow with Keras, Theano with Keras, Pytorch with Python two or three, CNTK with Keras, Caffe2, Caffe, and base Python. So now that we've set up our deep learning AMI, how do we use it? I'm actually following the steps in this blog post and we've done some of them so I'm going to scroll down a bit. So we've got our instance set up and all the samples require Jupyter Notebooks. So in order to set up Jupyter to work with this instance, you need to follow these steps. So find the instance public IP. And then rather than using a regular ssh, using the ssh command with -L you're going to create a mapping between the Jupyter notebook instance on Amazon and your local browser. So you're going to fill in the values here, now we used an Linux instance, so in my case the user's going to be ec-2user. And also you do not want to put quotes around your pem file. I'll show you what this looks like. So you'll connect using this command and then once you're connected you'll run the Jupyter Notebook command in the terminal. And then you'll be given a new URL which you'll paste into your browser. So I've already done these steps so let me show you what this looks like. So in terminal, you can see that I've switched to the directory where my key is. My key is demo-lynnlangit.pem and I have specified the ec2-user with my IP. And I've created this mapping with the ssh -L command localhost:8888 to localhost:8888. Then once I was connected I ran the Jupyter notebook command to start the Jupyter notebook on the AMI instance. Then once the notebook was started this URL which includes a token is what I used to open the Jupyter instance in a local browser. So I'm going to switch to that. So here's Jupyter which is running on the remote instance mapped to a local browser. In the next movie we're actually going to work with a sample. So let's get ready for that. We're going to go to the MXNet site and we're going to download a notebook, the MNIST sample that we're so familiar with. We're going to go back to Jupyter and we're going to upload our notebook. And then we're going to open it. We'll select our environment. And set the Kernel. In the next movie we'll work with this notebook.

### **Work with EMR for machine learning**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Narrator] So the next service we're going to look at is Elastic MapReduce, which is managed Hadoop, Spark, and other type of library clusters of virtual machines. So, the question is, why should we use virtual servers when we have API's, docker containers, and many other options? Well, the answer is, you shouldn't always. But there are situations for which you need the level of control. Could be security requirements. Could be custom setup steps. Could be the amount of data. I've been working with some bioinformatics customers, and in processing genomic sequencing result output, the data is huge, and taking advantage of the economies of spot pricing on EC2 is really critical for some machine learning workloads. Amazon Elastic MapReduce is platform as a service. It's Hadoop clusters, so master and worker nodes, that are customized EC2 instances, that are designed to run Hadoop and its associated libraries, such as Spark, SparkML, or machine learning, and other workloads. Many data processing libraries such as Spark and SparkML, can be installed just by selecting the option in the console, or setting a flag when running a script. You can also bootstrap using scripts to further customize your EMR cluster. And of course, to me one of the most important considerations is you can utilize spot instance pricing, or spot fleets, to optimize scaling for really large size data processing workloads, which include machine learning and regular processing. It's a regulator on the size. There is an interesting use case that I wanted to setup and show you, because it pulls together some of the different components that we've looked at throughout this course. I really think that one of the big takeaways, is not only selecting the right tool and the right algorithm to solve whatever business situation you're working with, but also to select the right interface. And, I've purposely focused very heavily on Jupyter notebooks, as opposed to using the traditional terminal, and just running batch jobs or scripts, because I really do think that they reflect the future for machine learning workloads. And to that end, I wanted to present in this section a very advanced implementation of working with a Spark cluster, and a Jupyter notebook. And this is a conceptual diagram, and we're going to look at this setup with EMR and a SageMaker Jupyter notebook in a minute. But you can see on the client side, we have Jupyter with something called Sparkmagic running inside of it. Sparkmagic sends code and SQL queries to an Apache library called Livy, which interacts with the Spark cluster, and sends the results back to Sparkmagic as text and JSON. So let's see what this looks like. So I roughly followed the steps in this blog post, to set this solution up. However I made a couple of changes, which I will point out. Now this took a couple of hours to setup, so I ran through the setup steps already, and I'm just going to show you the completed solution. So if you're going to plan to do this, especially if you're new to these technologies, it does take a good amount of time for you to run through the instructions, and for the services to get setup and connect. So, conceptually, you're going to setup EMR, Spark, and Livy. And you can do this by clicking in the console, which I've already done, and this just shows the screenshot of when you go to the EMR console, you just select the Livy and Spark libraries. Do pay attention to the configuration steps, because they are critical. As it says here, under network, select your VPC. And you're going to want to make a note of your EC2 subnet. And then you're going to create your cluster. Make sure you add keypairs, because you're going to need that for remote access. You'll also need the private IP address for the master node of your Spark cluster. Now it can take a good bit of time, depending on the machine type, and the location for the EMR cluster to setup, so like I said, I've already set mine up. So once your cluster is setup, then you want to find the master ID address to the private IP as it's shown here. And then you want to setup security groups, and open ports, so you're going to create a security group. And then you're going to add a rule, as it says here, to allow Livy to communicate across port 8998. And then you're going to setup your SageMaker notebook. And in SageMaker, you want to make sure to setup the VPC, where your EMR instance is, and subnet where it is, and the security group that you just created. Once your notebook is setup in SageMaker, you need to select the version of Sparkmagic that you're going to be working with. Now they use PySpark here, but I just decided to use regular Spark, because I'm more familiar with that. Notice your options are Sparkmagic Pyspark, PySpark3, Sparkmagic Spark, or SparkR. Then in the terminal, you need to type these three commands. CD .sparkmagic, and then wget, to get the configuration file. And then I followed these steps exactly. I used nano to open that file for editing. I used the control plus the backslash to enable editing, and find every instance of the word localhost, and then I replaced it with my EMR master private IP. A to accept. Control-X to accept. Y to accept. And enter. Then I tested my configuration, by running this curl command, with the EMR master private IP, and I got this result. And then I opened up the notebook, and I started working with it. So, let me show you what this looks like. So in the EMR console, I have my cluster. And I've been doing quite a bit of experimenting, so I have a number of applications. I was trying out local Spark shell, so I was SSH'ing into the instance, and just testing out some Spark, and I was also testing out PySpark. The items under action that have the heading livy, are different connections through the Jupyter notebook, and you can see that I have one live connection right now. In SageMaker, I created my demo notebook, as per the instructions, and then I opened it. And I opened it here. And then here is the actual notebook. So you can see, it's a demo EMR notebook, US east one, and if I run in my session here, I've got a Spark session. And then if I start my Spark session, or I check that it's started, and now just to give us a rest from all the complex machine learning we've done all through this course, I'm going to do something actually very computationally simple, compared to what we've been doing, because once we make this connection, then of course we could run any sort of machine learning, and in the case with Spark, that would probably be under SparkML library. But just to check this, it's a Pi estimator. So the digits of Pi. And this is just some Scala code that's running. And you can see in this estimation, here it is. So what I think's really interesting about this setup, is that, like I said, it was relatively easy to set this up. There were a few little tricky parts, and like I said, I chose to use Spark rather than PySpark because I was more familiar, but the idea of having a Jupyter notebook, rather than a terminal interfacing into my EMR cluster is really, really appealing to me. And, being able to use SageMaker to quickly set that notebook up, rather than having another EC2 instance, also was super appealing. So, I see this as a blending of kind of the old and the new. You've got the power of the EMR cluster, and then you have the flexibility of the new services, SageMaker being the example here.

## **Question 1 of 4**

Gluon is to \_\_\_\_\_, as Keras is to \_\_\_\_\_.

* SageMaker; MXNet
* Tensorflow; MXNet
* MXNet; TensorFlow  
  Correct  
  Gluon and Keras are the both machine learning libraries.

## **Question 2 of 4**

In deep learning feature extraction is done by \_\_\_\_\_.

* the algorithm  
  Correct  
  The algorithm performs the extraction.
* humans
* the GPU(s)
* as part of data preparation for input to the algorithm

## **Question 3 of 4**

Most deep-learning algorithms are optimized to take advantage of the parallelism available via \_\_\_\_\_.

* SPUs
* GPUs  
  Correct  
  When using AWS deep learning AMI you should consider using GPUs.
* AMIs

## **Question 4 of 4**

Databricks is designed for \_\_\_\_\_ and MXNet is designed to work with \_\_\_\_\_.

* GPUs; production  
  Incorrect  
  Databricks is not optimized for use with GPUs.
* GPUs; CPUs
* production; GPUs  
  Correct  
  Databricks is designed for production and MXNet is optimized to work with GPUs.

### **AWS ML APIs for conversational apps**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Now that we've completed looking at the various services available in the Amazon ecosystem to work with machine learning, let's look at some sample application architectures. And let's start first with some of the APIs that we looked at earlier in this course. So you may remember that there's a number of APIs. I basically call 'em black boxes, where you put some data in, the machine learning is applied, and you get some result out. And two of 'em that we looked at are used in this particular architecture, which is a conversational application. So the business usage is that you have some interaction with a customer and you want to make use of both voice and text. So you can see it starts with an end user is selecting a feature, and then they go through a permissioning phase, so IAM. And then information is retrieved from a calendar. Also at that same time, a request is sent to the Amazon Lex service to start a conversation. The next step, the request to Lex, kicks off a unit of compute within Amazon Lambda. So Lex is going to use the Lambda functions to get info from services provider or S3. So it could be external API or static information or data that's stored in S3. Once that's returned, then the output goes to Polly. And do you remember what Polly does? Polly, of course, turns text into speech. So it's a speech-enabled application. So it's interesting to see that this is a very advanced application that could be developed very, very quickly because the machine learning models basically are already completed. All you have to do is read the APIs, understand in which format the data needs to be sent in, and understand how the data will be returned so you can consume that with your application. It's been really interesting. A lot of development has been driven off the Alexa device in the Amazon ecosystem. There have been a number of hackathons and developer fests, and it's been really fascinating to see how conversational applications have been rapidly enabled through this medium of machine learning APIs. I think that it's really the most innovative part of the machine learning ecosystem, not only in Amazon, but across all the cloud vendors. And I would highly recommend that you pay a lot of attention to the future releases, because this is an active area of product development across all cloud vendors.

### **AWS ML service for IoT apps**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] This next application sample uses the platform as a service, Amazon Machine Learning service. Now this is designed for when you have data but no one on your team who's familiar with how to use machine learning. So remember from the earlier movies around this subject that there are two algorithms and Amazon will automatically select the appropriate algorithm, perform the training, and show you the results. So it's not always going to be useful, but this is a use case where it really might work. The data's pretty straight forward. We start with IoT sensors, so we would have lots of data because of the event data coming off of the sensors. Then a Python API would send that data to an Amazon Kinesis Firehose for streaming. The data would be staged intermediately in S3, and then copied into Redshift, probably aggregated. This flow is showing a direct copy. You may remember that Amazon Machine Learning accepts both Amazon S3 data or Amazon Redshift data as data sources. Once the data's supplied to Machine Learning, it will suggest the appropriate algorithm based on the data. You can of course adjust that, and it will try to infer the model. So the idea here is to be able to apply predictive analytics when you have bunches of data to see if you can get meaning out of it using the two most common algorithms. Now the easiest way to try this out is just to set it up to see if you get meaning. In some cases this is going to work for you just fine, it's going to be a really quick solution to be able to get further analysis on this sensor event data. In other cases it's a starting point. It gives you an idea of what's possible, and you may need to then either adjust some of the parameters of the machine learning model that is generated, or move to a different type of service such as SageMaker where you have more control over the model outputs.

### **Spark ML and Databricks AWS for real-time apps**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] In this next example we're going to consider a solution using Machine Learning for near real time predictions. Now this is a scenario where virtual servers are often used because you want to have a persistent running compute set of instances. You could of course use plain old EC2 or EMR with Hadoop Spark. In this case this architecture shows databricks, and you might remember that databricks is a commercial vendor that runs on top of either Amazon or Azure, and they're the home of the majority of the open source Apache Spark commuter community. So I've found because databricks packages their solution as software as a service, very easy to setup and use as you might remember from our movies earlier in this course. Databricks includes a notebook interface that allows you to quickly setup clusters and work with notebooks to try out your experiments, and then you can actually turn them into production jobs. In addition to that, they have a very optimized implementation of Spark because, of course, also the creators of Spark work there. So in addition to using Spark, often Spark Machine Learning will be used in these scenarios. So for this scenario, starts with Kinesis as our input pipe. So we have a realtime data feed, logs, pixels, sensor information. That's sent to a databricks cluster running Spark ML and Streaming. Micro-batches are sent to a NoSQL database, in this case riak. It could really be any NoSQL database for persistence, and then the prediction results are served up to AWS Lambda and delivered to the client through the API Gateway. Now as I've been mentioning throughout this course, there really should be a business justification for using servers as more and more lightweight services are becoming available. In particular, I spent a lot of time looking at SageMaker, which, although a new service, has already had very rapid customer adoption because of its infrastructure, which is based not on virtual machines, rather on the more lightweight Docker containers.

### **VariantSpark and EMR for genomic research**

Selecting transcript lines in this section will navigate to timestamp in the video

- For the last use case that I want to discuss in this course we need a little bit more background. This is one for which a custom machine algorithm is required. Here we're talking about genomic scale data. So this is an algorithm that was built by the team at CSIRO BioInformatics in Sydney, Australia. It's called VariantSpark. It's an implementation of a custom algorithm that's designed to process the amount of data that we get from sequencing human genomes. In case you're not aware, each sequence provides three billion with a B data points. So the idea is by getting this information and comparing the differences to reference genomes we can provide information that can be used in the creation of personalized treatments for medical conditions. What kind of medical conditions? Conditions like cancer. Again, I know this is a little bit of an aside, but I think it's very important for us to consider the possibilities that we have around the usages of machine learning and big data, and to me, this is the most important, to reduce the amount of suffering from cancer in the world. So I encourage you if you're interested in this domain, I know I have been. I've started to do some work actually with the team at CSIRO. There's a really pressing need in the industry for more people who have technical skills in all aspects of software product life cycles, particularly in machine learning to work with our bioinformatics professionals so that we can take the data that we're getting off the sequencers and putting it to good use. Because the technical implementation of VariantSpark is important to the architecture that we select, let's take a minute or two and talk about how the algorithm was implemented. It's an implementation of random forests which are sets of decision trees. The key differences, the input data is too large, it's both too wide in terms of number of features or columns, because remember the three billion samples, and also too deep because we have to have a large amount of populations to compare against. So what the team did is they implemented a custom random forest, they call it wide random forest, where they split the data both vertically and horizontally and then they recombine it so that the classification, as you can see at the bottom here, is generated from a consensus over all the trees, and then the association is produced. The association means which genomic variants seem to be of greatest interest. And why this is so important is optimizing a run of this algorithm gives researchers working with Cast Nine and technologies like it faster and more useful feedback so that they can work to find solutions around personalized medicine. So let's tie this back to technology implementation. This would be a use case for using an EMR cluster, and the reason for this is the amount of data coming through. Genomic scale data warrants using EMR clusters so that the teams can take advantage of spot pricing, because they need to be able to run at a very economical way very large inputs. So you can see in this architectural diagram there are two different clients, one client is using a Jupiter notebook, the other one is using the command line. Once they issue their commands then the analysis is running. The data persistence is in S3 buckets and the compute is being run on a Hadoop cluster with VariantSpark running on the worker notes.

### **Best practices for algorithms and architectures**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] So to summarize our learning in this course. First around algorithms, use the right algorithm for the job. Simple as possible always wins. I purposely reviewed the same problem using different algorithms and different platforms, the digit classification problem, because that's the way you should look at your analysis problems. You should run them across different algorithms and different platforms, always selecting the simplest one. You want to take advantage of AWS algorithm optimization. Again, I'm going to call out SageMaker here. Amazon has worked very diligently to optimize ten common algorithms, and it should be part of your testing to see if running on the SageMaker platform, with the automatic hyper-parameter tuning included, will help you get your machine-learning model to market quicker. You want to consider using the machine-learning APIs first. Look at Lex, Polly, Recognition, and the growing list. I'm sure there will be more by the time you watch this. This is a really active area of service development. If those work for you, they're really simple, they're battle-tested. Amazon uses them to run Amazon.com, and you can get to market super-quickly using those. Next up, you want to look at using SageMaker or Amazon Machine Learning service. As you can probably tell, I'm really, really excited about SageMaker, and I don't think I'm alone. It's pretty astounding how many customers have already picked it up, and it's only been in production about three months as of the time of this recording. It really is an exciting and usable service. I particularly like the fact that it's split into components. So you can, for example, use Jupyter Notebooks as a service and be done with it. You can use it for model training. You can use it for model hosting, or all of it. It's very well-designed. Of course, in some situations, using a deep-learning algorithm or a custom algorithm is really the only solution. And clearly, I've worked with that as well. I'm working with the team at CSIRO on the genomics use case, and that's a situation where a custom algorithm is the best fit. As a big tip, I included movies about working with the AWS Machine Learning AMI. I have seen customers waste days, or even weeks, trying to set up various deep-learning libraries, and also install GPU drivers on plain vanilla EC2 instances. Just don't do it. Use Amazon's Machine Learning API as a starting point. Also, you can use it to compare different frameworks. You can compare the number of GPUs on an instance. It's really great for testing. And really, to summarize the learning about algorithms, I'm just going to say one more time: use the simplest possible thing that will work. It's very, very important, and it's the best practice around machine learning. In addition to selecting the best-fit algorithm, it's important to select the best-fit architecture. So you want to use the right service for the job, we talked about that in terms of algorithms. And, you want to consider using the machine-learning APIs first, of course. SageMaker, or Amazon ML second. In terms of virtual servers, again, you want to use the simplest possible solution. Also, if you're using a solution that's spark-heavy, you really want to take a look at Databricks. They're very well-optimized for running both regular Spark, and Spark ML workloads. If you do choose to use elastic map reduce, or EMR, it's very important that you understand and utilize the Amazon economies around EMR. And what I mean by that is using Spot Instances. I've seen a number of customers not taking advantage, and it's really just silly. Because the reason you use EMR is you have huge workloads. So if you don't use Spot, you end up really overpaying. Remember, Spot Instances can save you 75 to 90 percent of the cost. So you need to take the time to understand how to use Spot Instances, and the new Spot Fleets, if you're going to work with EMR. It's just really, really critical. I really don't recommend using plain, vanilla EC2 for any machine-learning workloads. Now, I have had one customer over the years that had extreme security requirements, and they needed it controlled all the way down to the OS level. I guess that would be the exception case. But you're going to have to have a team of administrators to not only administer the servers, but also to take care of setting up all the machine-learning libraries, maintaining them, so on and so forth. So you need to have a business justification that's pretty significant for setting up machine-learning virtual servers that are plain vanilla, or infrastructure as a service. I really don't like to see that. And I do see that, and that's one of the reasons that I'm talking about it here. I always would start with probably Databricks, and then also look at EMR. And this should be for huge and complex workloads. Another aspect of this is I think now, with SageMaker, some of the workloads that were running on servers can now be migrated, or they can be simply piloted if it's a new type of implementation on SageMaker. I do think that the lightweight, container-based architecture, and the partitioning of services, is a very intelligent approach to a lot of the machine-learning workloads that I have had customers work with me on. The other thing, of course, is if you are working with servers, you want to take advantage of GPUs. Yes, they are expensive. So, you need to test. And as I said, when we were talking about algorithms, a great way to test is using the AWS Machine Learning AMIs so you can quickly set up and test, and tear them down, and try with different numbers of GPUs. Very, very important because you want to make sure, if you're needing to use GPUs, you want to figure out how many are going to be optimal, you want to figure out for what part of the workload. Again, there's tuning that is an important part of this. So there are lots of choices around architectures. Just like with algorithms, simple is better. Serverless is really leading the way, and that would be the APIs, so Lex, Poly and all that, followed by Docker, followed by Servers. And it is by design that I created the course this way. Most of the machine-learning courses that I see out there start with Servers because frankly, that's what we've had up until very recently. However, as with other aspects of cloud architectures, particularly Amazon Architectures, I'm working with customers to get economies of scale using the new serverless and Docker-based products, and it's really a competitive advantage, and it's one that I want you to have and understand for your machine-learning workloads.

## **Question 1 of 1**

AWS \_\_\_\_\_ is often used to stream data for real-time applications.

* Rekognition
* Kinesis  
  Correct  
  With Kinesis, you get a real-time data feed of logs, pixels, and sensor information.
* Comprehend

### **Next steps**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Lynn] Well, here we are already at the end of our tour of AWS Machine Learning. It is a vast world and there's always more to learn, so let me leave you with some resources that I like to use. The first one I mentioned before but I'll say it again, especially if you're new to machine learning in general, scikit-learn is a great place to learn about all the core algorithms. I find myself very frequently returning there when I have questions. A couple other resources; KDnuggets is a great, general website about machine learning, and last but not least, Kaggle Competitions. It's really fun to join a team and to work on some projects. Some of the projects are just for fun, and some are competitions that have prizes, money, trips, that kind of stuff associated to 'em. Kaggle Competitions are a great, fun way to get to know people interested in machine learning worldwide. Thanks for joining me on this tour of AWS Machine Learning. I'm Lynn Langit.