**Implementing a Machine Learning pipeline with Amazon SageMaker**

Machine Learning and Its Life Cycle in the Cloud

**Machine Learning** (**ML**) is a technique that has been around for decades. It is hard to believe how ubiquitous ML is now in our daily life. It has also been a rocky road for the field of ML to become mainstream, until the recent major leap in computer technology. Today's computer hardware is faster, smaller, and smarter. Internet speeds are faster and more convenient. Storage is cheaper and smaller. Now, it is rather easy to collect, store, and process massive amounts of data with the technology we have now. We are able to create sizeable datasets that we were not able to before, train ML models using compute resources that were not available before, and make use of ML models in every corner of our lives.

For example, media streaming companies can now build ML recommendation engines at a global scale using their title collections and customer activity data on their websites to provide the most relevant content in real time in order to optimize the customer experience. The size of the data for both the titles and customer preferences and activity is on a scale that wasn't possible 20 years ago, considering how many of us are currently using a streaming service.

Training an ML model at this scale, using ML algorithms that are becoming increasingly more complex, requires a robust and scalable solution. After a model is trained, companies are able to serve the model at a global scale where millions of users visit the application from web and mobile devices at the same time.

Companies are also creating more and more models for each segment of customers or even one model for one customer. There is another dimension to this – companies are rolling out new models at a pace that would not have been possible to manage without a pipeline that trains, evaluates, tests, and deploys a new model automatically. Cloud computing has provided a perfect foundation for the streaming service provider to perform these ML activities to increase customer satisfaction.

# Understanding ML and its life cycle

At its core, ML is a process that uses computer algorithms to automatically discover the underlying patterns and trends in a **dataset** (which is a collection of **observations** with **features**, also known as **variables**), make a prediction, obtain the error measure against a **ground truth** (if provided), and "learn" from the error with an **optimization** process in order to make a prediction next time. At the end of the process, an **ML model** is fitted or trained so that it can be used to apply the knowledge it learned to apply a decision based on the features of a new observation. The first part, generating a model, is called **training**, while the second part is called **prediction** or **inference**.

There are three basic types of ML algorithms based on the way the training process takes place – supervised learning, unsupervised learning, and reinforcement learning. A **supervised learning** algorithm is given a set of observations with a ground truth from the past. A ground truth is a key ingredient to train a supervised learning algorithm, as it drives how the model learns and makes future predictions – hence the "supervised" in the name, as the learning is supervised by the ground truth. **Unsupervised learning**, on the other hand, does not require a ground truth for the observations to learn how to apply the prediction. It finds patterns and relationships solely based on the features of the observations. However, a ground truth, if it exists, would still help us validate and understand the accuracy of the model in the case of unsupervised learning. **Reinforcement learning**, often abbreviated as **RL**, has quite a different learning paradigm compared to the previous two. RL consists of an **agent** interacting with an **environment** with a set of **actions**, and corresponding **rewards** and **states**. The learning is not guided by a ground truth, rather by optimizing cumulative rewards with actions. The trained model in the end would be able to perform actions autonomously in an environment that would achieve the best rewards.

## **An ML life cycle**

Now we have a basic understanding of what ML is, we can go broader to see what a typical ML life cycle looks like, as illustrated in the following figure:

Figure 1.1 – The ML life cycle


Figure 1.1 – The ML life cycle

### **Problem framing**

The first step in a successful ML life cycle is framing the business problem into an ML problem. Business problems come in all shapes and forms. For example, "How do we increase sales of a newly released product?" and "How do we improve the QA Quality Assessment (QA) throughput on the assembly line?" Business problems such as these, usually qualitative, are not something ML can be directly applied to. But looking at the business problem statement, we should think about how it can be translated into an ML problem. We should ask questions like the following:

* "What are the key factors to the success of product sales?"
* "Who are the people that are most likely to purchase the product?"
* "What is the bottleneck in throughput in the assembly line?"
* "How do we know whether an item is defective? What differentiates a defective one from a normal one?"

By asking questions like these, we start to dig into the realm of **pattern recognition**, a process of recognizing patterns from the data at hand. Having the right questions that can be formulated into pattern recognition, we are a step closer to framing an ML problem. Then, we also need to understand what the key metric is to gauge the success of an approach, regardless of whether we use ML or other approaches. It is quite straightforward to measure, for example, daily product sales. We can also improve sales by targeting advertisements to the people that are mostly like to convert. Then, we get questions like the following:

* "How do we measure the conversion?"
* "What are the common characteristics of the consumers who have bought this product?"

More importantly, we need to find out whether there is even a target metric for us to predict! If there are targets, we can frame the problem as an ML problem, such as predicting future sales (supervised learning and regression), predicting whether a customer is going to buy a certain product or not (supervised learning and classification), or identifying defective items (supervised learning and classification). Questions that do not have a clear target to predict would fall into an unsupervised learning task in order to apply the pattern discovered in the data to future data points. Use cases where the target is dynamic and of high uncertainty, such as autonomous driving, robotic control, and stock price prediction, are good candidates for RL.

### **Data exploration and engineering**

Sourcing data is the first step of a successful ML modeling journey. Once we have clearly defined both our business problem and ML problem with a basic understanding of the scope of the problem – meaning, what are the metrics and what are the factors – we can start gathering the data needed for ML. Data scientists explore the data sources to find out relevant information that could support the modeling. Sometimes, the data being captured and collected within the organization is easily accessible. Sometimes, the data is available outside your organization and would require you to reach out and ask for data sharing permission.

Sometimes, datasets can be sourced from the public internet and institutions that focus on creating and sharing standardized datasets for ML purposes, which is especially true for computer vision and natural language understanding use cases. Furthermore, data can arrive through streaming from websites and applications. Connections to a database, data lake, data warehouse, and streaming source need to be set up. Data needs to be integrated into the ML platform for processing and engineering before an ML model can be trained.

Managing data irregularity and heterogeneity is the second step in the ML life cycle. Data needs to be processed to remove irregularities such as missing values, incorrect data entry, and outliers because many ML algorithms have statistical assumptions that these irregularities would violate and render the modeling ineffective (if not invalid). For example, the linear regression model assumes that an error or residual is normally distributed, therefore it is important to check whether there are outliers that could contribute to such a violation. If so, we must perform the necessary preprocessing tasks to remedy it. Common preprocessing approaches include, but are not limited to, removal of invalid entries, removal of extreme data points (also known as outliers), and filling in missing values. Data also need to be processed to remove heterogeneity across features and normalize them into the same scale, as some ML algorithms are sensitive to the scale of the features and would develop a bias towards features with a larger scale. Common approaches include **min-max scaling** and **z-standardization** (**z-score**).

Visualization and data analysis is the third step in the ML life cycle. Data visualization allows data scientists to easily understand visually how data is distributed and what the trends are in the data. **Exploratory Data Analysis** (**EDA**) allows data scientists to understand the statistical behavior of the data at hand, figure out the information that has predictive power to be included in the modeling process, and eliminate any redundancy in the data, such as duplicated entries, multicollinearity, and unimportant features.

Feature engineering is the fourth step in the ML life cycle. Even with the various sources from which we are collecting data, ML models oftentimes benefit from engineered features that are calculated from existing features. For example, **Body Mass Index** (**BMI**) is a well-known engineered feature, calculated using the height and weight of a person, and is also an established feature (or risk factor, in clinical terms) that predicts certain diseases rather than height or weight alone. Feature engineering often requires extensive experience in the domain and experimentation to find out what recipes are adding predictive power to the modeling.

### **Modeling and evaluation**

For a data scientist, ML modeling is the most exciting part of the life cycle (I think so; I hope you agree with me). You've formulated the problem in the language of ML. You've collected, processed the data, and looked at the underlying trends that give you enough hints to build an ML model. Now, it's time to build your first model for the dataset, but wait – what model, what algorithm, and what metric do we use to evaluate the performance? Well, that's the core of modeling and evaluation.

The goal is to explore and find out a satisfactory ML model, with an objective metric, from all possible algorithms, feature sets, and hyperparameters. This is definitely not an easy task and requires extensive experience. Depending on the problem type (whether it's classification, regression, or reinforcement learning), data type (as in whether it's tabular, text, or image data), data distribution (is there a class imbalance or outliers?), and domain (medical, financial, or industrial), you can narrow down the choice of algorithms to a handful. With each of these algorithms, there are hyperparameters that control the behavior and performance of the algorithm on the provided data. What is also needed is a definition of an objective metric and a threshold that meets the business requirement, using the metric to guide you toward the best model. You may blindly choose one or two algorithm-hyperparameter combinations for your project, but you may not reach the optimal solution in just one or two trials. It is rather typical for a data scientist to try out hundreds if not thousands of combinations. How is that possible?

This is why establishing a streamlined model training and evaluation process is such a critical step in the process. Once the model training and evaluation is automated, you can simply launch the process that helps you automatically iterate through the experimentations among algorithms and hyperparameters, and compare the metric performance to find out the optimal solution. This process is called **hyperparameter tuning** or **hyperparameter optimization**. If multiple algorithms are the subject of tuning, it can also be called **multi-algorithm hyperparameter tuning**.

### **Production – predicting, monitoring, and retraining**

An ML model needs to be put in use in order to have an impact on the business. However, the production process is different from that of a typical software application. Unlike other software applications where business logic can be pre-written and tested exhaustively with edge cases before production, there is no guarantee that once the model is trained and evaluated, it will be performing at the same level in production as in the testing environment. This is because ML models use probabilistic, statistical, and fuzzy logic to infer an outcome for each incoming data point, and the testing, that is, the model evaluation, is typically done without true prior knowledge of production data. The best a data scientist can do prior to production is to create training data from a sample that closely represents real-world data, and evaluate the model with an out-of-sample strategy in order to get an unbiased idea of how the model would perform on unseen data. While in production, the incoming data is completely unseen by the model; how to evaluate live model performance, and how to take actions on that evaluation, are critical topics for productionizing ML models.

Model performance can be monitored with two approaches. One that is more straightforward is to capture the ground truth for the unseen data and compare the prediction against the ground truth. The second approach is to use the drift in data as a proxy to determine whether the model is going to behave in an expected way. In some use cases, the first approach is not feasible, as the true outcome (the ground truth) may lag behind the event for a long time. For example, in a disease prediction use case, where the purpose of ML modeling is to help a healthcare provider to find a likely outcome in the future, say three months, with current health metrics, it is not possible to gather a true ground truth less than three months or even later, depending on the onset of the disease. It is, therefore, impractical to only fix the model after obtaining it, should it be proven ineffective.

The second approach lies in the premise that an ML model learns statistically and probabilistically from the training data and would behave differently when a new dataset with different statistical characteristics is provided. A model would return gibberish when data does not come from the same statistical distribution. Therefore, by detecting the drift in data, it gives a more real-time estimate of how the model is going to perform. Take the disease prediction use case once again as an example: when data about a group of patients in their 30s is sent to an ML model that is trained on data with an average age of 65 for prediction, it is likely that the model is going to be clueless about these new patients. So we need to take action.

Retraining and updating the model makes sure that it stays performant for future data. Being able to capture the ground truth and detecting the data drift helps create a retraining strategy at the right time. The data that has drifted and the ground truth are the great input into the retraining process, as they will help the model to cover a wider statistical distribution.

Now that we have a clear idea of the basics of the uses and life cycle of ML development, let's take the next step and investigate how it can work with the cloud.

# Building ML in the cloud

Cloud computing is a technology that delivers on-demand IT resources that can grow and shrink at any time, depending on the need. There is no more buying and maintaining computer servers or data centers. It is much like utilities in your home, such as water, which is there when you turn on the faucet. If you turn it all the way, you get a high-pressure water stream. If you turn it down, you conserve water. If you don't need it anymore, you turn it off completely. With this model, developers and teams get the following benefits from on-demand cloud computing:

* **Agility**: Quickly spin up resources as you need them. Develop and roll out new apps, experiment with new ideas, and fail quickly without risks.
* **Elasticity**: Scale your resources as you need them. Cloud computing takes away "undifferentiated heavy lifting" – racking up additional servers and planning capacity for the future. These are things that don't help address your core business problems.
* **Global availability**: With a click of a button, you can spin up resources that are closest to your customers/users without relocating your physical compute resources.

How does this impact the field of ML? As compute resources become easier to acquire, information exchange becomes much more frequent. As that happens, more data is generated and stored. And more data means more opportunities to train more accurate ML models. The agility, elasticity, and scale that cloud computing provides accelerates the development and application of ML models from weeks or months down to a much shorter cycle so that developers can now generate and improve ML models faster than ever. Developers are no longer constrained by physical compute resources available to them. With better ML models, businesses can make better decisions and provide better product experiences to customers.

# What Is Amazon SageMaker?

# Amazon SageMaker is a fully managed machine learning service. With SageMaker, data scientists and developers can quickly and easily build and train machine learning models, and then directly deploy them into a production-ready hosted environment. It provides an integrated Jupyter authoring notebook instance for easy access to your data sources for exploration and analysis, so you don't have to manage servers. It also provides common machine learning algorithms that are optimized to run efficiently against extremely large data in a distributed environment. With native support for bring-your-own algorithms and frameworks, SageMaker offers flexible distributed training options that adjust to yourspecific workflows. Deploy a model into a secure and scalable environment by launching it with a few clicks from SageMaker Studio or the SageMaker console.

# Machine Learning with Amazon SageMaker

# This section describes a typical machine learning workflow and summarizes how you accomplish those tasks with Amazon SageMaker.

# In machine learning, you "teach" a computer to make predictions, or inferences. First, you use an algorithm and example data to train a model. Then you integrate your model into your application to generate inferences in real time and at scale. In a production environment, a model typically learns from millions of example data items and produces inferences in hundreds to less than 20 milliseconds.

# The following diagram illustrates the typical workflow for creating a machine learning model:

# 

# As the diagram illustrates, you typically perform the following activities:

# 1. Generate example data—To train a model, you need example data. The type of data that you needdepends on the business problem that you want the model to solve (the inferences that you wantthe model to generate). For example, suppose that you want to create a model to predict a numbergiven an input image of a handwritten digit. To train such a model, you need example images ofhandwritten numbers.Data scientists often spend a lot of time exploring and preprocessing, or "wrangling," example databefore using it for model training. To preprocess data, you typically do the following:

# a. Fetch the data— You might have in-house example data repositories, or you might use datasetsthat are publicly available. Typically, you pull the dataset or datasets into a single repository.

# b. Clean the data—To improve model training, inspect the data and clean it as needed. For example, if your data has a country name attribute with values United States and US, you might want to edit the data to be consistent.

# c. Prepare or transform the data—To improve performance, you might perform additional data transformations. For example, you might choose to combine attributes. If your model predicts the conditions that require de-icing an aircraft, instead of using temperature and humidity attributes separately, you might combine those attributes into a new attribute to get a better model.

# In SageMaker, you preprocess example data in a Jupyter notebook on your notebook instance. You use your notebook to fetch your dataset, explore it, and prepare it for model training.

# 2. Train a model—Model training includes both training and evaluating the model, as follows:• Training the model— To train a model, you need an algorithm or a pre-trained base model. Thealgorithm you choose depends on a number of factors. For a quick, out-of-the-box solution, youmight be able to use one of the algorithms that SageMaker provides. For a list of algorithmsprovided by SageMaker and related considerations, see Use Amazon SageMaker Built-in Algorithmsor Pre-trained Models. You also need compute resources for training. Depending on the size of your training dataset and how quickly you need the results, you can use resources ranging from a single general-purpose instance to a distributed cluster of GPU instances.

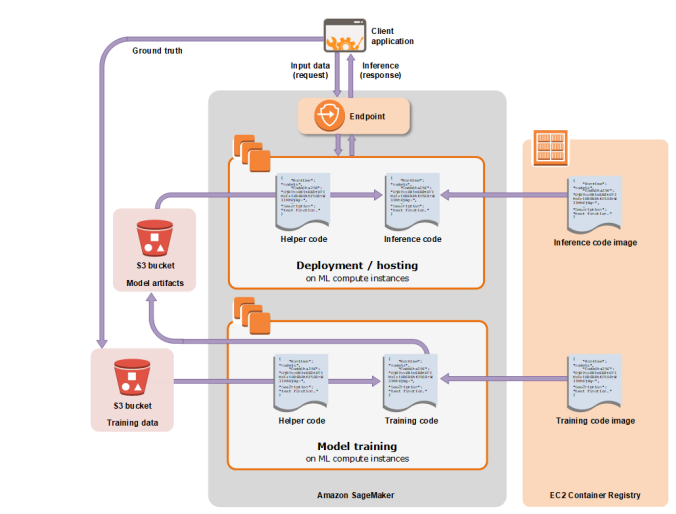
# 3.Evaluating the model—After you've trained your model, you evaluate it to determine whether the accuracy of the inferences is acceptable. In SageMaker, you use either the AWS SDK for Python (Boto) or the high-level Python library that SageMaker provides to send requests to the model forinferences.

# You use a Jupyter notebook in your SageMaker notebook instance to train and evaluate your model.

# 3. Deploy the model— You traditionally re-engineer a model before you integrate it with your application and deploy it. With SageMaker hosting services, you can deploy your model independently, , decoupling it from your application code. Machine learning is a continuous cycle. After deploying a model, you monitor the inferences, collect “ground truth," and evaluate the model to identify drift. You then increase the accuracy of your inferences by updating your training data to include the newly collected ground truth. You do this by retraining the model with the new dataset. As more and more example data becomes available, you continue retraining your model to increase accuracy.

# Exploring AWS essentials for ML

**Amazon Web Services** (**AWS**) offers cloud computing resources to developers of all kinds to create applications and solutions for their businesses. AWS manages the technology and infrastructure in a secure environment and a scalable fashion, taking away the undifferentiated heavy lifting of infrastructure management from developers. AWS provides a broad range of services, including ML, artificial intelligence, the internet of things, analytics, and application development tools. These are built on top of the following key areas – **compute**, **storage**, **databases**, and **security**. Before we start our journey with Amazon SageMaker Studio, which is one of the ML offerings from AWS, it is important to know the core services that are commonly used while developing your ML projects on Amazon SageMaker Studio.\



The area labeled SageMaker highlights the two components of SageMaker: model training and model deployment.To train a model in SageMaker, you create a training job. The training job includes the fol lowing information:

• The URL of the Amazon Simple Storage Service (Amazon S3) bucket where you've stored the trainingdata.

• The compute resources that you want SageMaker to use for model training. Compute resources are MLcompute instances that are managed by SageMaker.

• The URL of the S3 bucket where you want to store the output of the job.

• The Amazon Elastic Container Registry path where the training code is stored.

## **Compute**

For ML in the cloud, developers need computational resources in all aspects of the life cycle. **Amazon Elastic Compute Cloud** (**Amazon EC2**) is the most fundamental cloud computing environment for developers to process, train, and host ML models. Amazon EC2 provides a wide range of compute instance types for many purposes, such as compute-optimized instances for compute-intensive work, memory-optimized instances for applications that have a large memory footprint, and **Graphics Processing Unit**(**GPU**)-accelerated instances for deep learning training.

Amazon SageMaker also offers on-demand compute resources for ML developers to run processing, training, and model hosting. Amazon SageMaker's ML instances build on top of Amazon EC2 instances and equip the instances with a fully managed, optimized versions of popular ML frameworks such as TensorFlow, PyTorch, MXNet, and scikit-learn, which are optimized for Amazon EC2 compute instances. Developers do not need to manage the provisioning and patching of the ML instances, so they can focus on the ML life cycle.

## **Storage**

While conducting an ML project, developers need to be able to access files, store codes, and store artifacts. Reliable storage is crucial to an ML project. AWS provides several types of storage options for ML development. **Amazon Simple Storage Service** (**Amazon S3**) and **Amazon Elastic File System** (**Amazon EFS**) are the two that are most relevant to the development of ML projects in Amazon SageMaker Studio.

Amazon S3 is an object storage service that allows developers to store any amount of data with high security, availability, and scalability. ML developers can store structured and unstructured data, and ML models with versioning on Amazon S3. Amazon S3 can also be used to build a data lake for analytics and to store backups and archives.

Amazon EFS provides a fully managed, serverless filesystem that allows developers to store and share files across users on the filesystem without any storage provisioning, as the filesystem increases and decreases its capacity automatically when you add or delete files. It is often used in a **High-Performance Cluster** (**HPC**) setting and applications where parallel or simultaneous data access across threads, processing tasks, compute instances, and users with high throughput are required. As Amazon SageMaker Studio embeds an Amazon EFS filesystem, each user on Amazon SageMaker Studio gets a home directory for storing and accessing data, codes, and notebooks.

## **Database and analytics**

Besides storage options, where data is saved as a file or an object, AWS users can store and access data at a data point level using database services such as **Amazon Relational Database Service** (**Amazon RDS**) and **Amazon DynamoDB**. AWS Analytics services such as **AWS Glue** and **Amazon Athena** provide capabilities in storing, querying, and data processing that are critical in the early phase of the ML life cycle.

For an ML project, relational databases are a common source of data for modeling. Amazon RDS is a cost-efficient and scalable relational database service in the cloud. It offers six database engines, including open sourced PostgreSQL, MySQL, and MariaDB, and the Oracle and SQL Server commercial databases. Infrastructure provisioning and management are made easy with Amazon RDS.

Another popular database is NoSQL, which uses key-value pairs as the data structure. Unlike relational databases, stringent schema requirements for tables are not required in NoSQL databases. Users can input data with a flexible schema for each row without needing to change the schema. Amazon DynamoDB is a key-value and document database that is fully managed, serverless, and highly scalable.

AWS Glue is a data integration service that has several features to help developers discover and transform data from sources for analytics and ML. The AWS Glue Data Catalog offers a persistent metadata store as a central repository for all your data sources, such as tables in Amazon S3, Amazon RDS, and Amazon DynamoDB. Developers can view all their tables and metadata such as the schema and time of update in one place – AWS Glue Data Catalog. AWS Glue's ETL service helps streamline the extract, transform, and load steps right after data is discovered and cataloged in the AWS Glue Data Catalog.

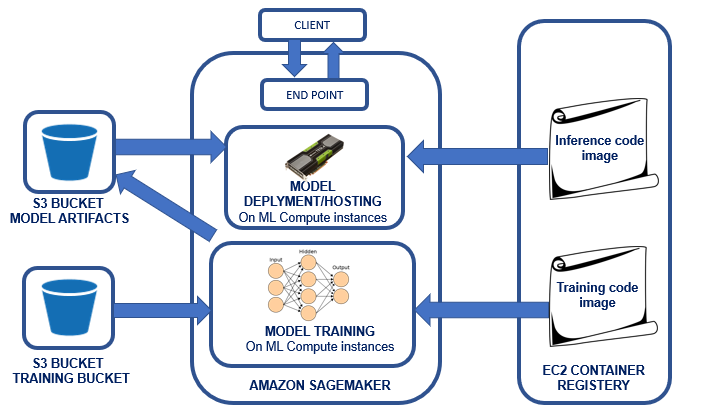
Amazon Athena is an analytics service that gives developers an interactive and serverless query experience. As a serverless service, developers do not need to think about the infrastructure underneath but instead focus on their data queries. You can easily point Amazon Athena to your data in Amazon S3 with a schema definition to start querying. Amazon Athena integrates natively with the AWS Glue Data Catalog to allow you to quickly and easily query against your data from all sources and services. Amazon Athena is also heavily integrated into several aspects of Amazon SageMaker Studio, which we will talk about in more detail throughout this book.

## **Security**

Security is job zero when you develop your applications, access data, and train ML models on AWS. The access and identity control aspect of the security is governed by the **AWS Identity and Access Management** (**IAM**) service. Any control over services, cloud resources, authentication, and authorization can be granularly managed by AWS IAM.

Key concepts in IAM are the IAM user, group, role, and policy. Each person who logs onto AWS would assume an IAM user. Each IAM user has a list of IAM policies attached that governs the resources and actions in AWS that this IAM user can command and access. An IAM user can also inherit IAM policies from that of an IAM group, a collection of users who have similar responsibilities. An IAM role is similar to an IAM user in that it has a set of permissions to access resources and to perform actions. An IAM role differs from an IAM user in that a role can be assumed by users, applications, or services. For example, you can create and assign an AWS service role to an application in the cloud to permit what services and resources this application can access. An IAM user who has permission to an application can securely execute the application without worrying that the application would reach out to unauthorized resources.

**AMAZON SAGEMAKER MODEL TRAINING AND DEPLOYMENT OVERVIEW**



**Two components are present in Amazon SageMaker:**

* + Model training
  + Model deployment

To start training an AI/ML model using Amazon SageMaker, you will need to create a training job with the following:

* + Amazon S3 bucket URL (training data): where training data is located.
  + Compute resources: Amazon SageMaker will train the model using instances managed by Amazon SageMaker.
  + Amazon S3 bucket URL (Output): this bucket will host the output from the training.
  + Amazon Elastic Container Registry path: where training code is stored.

Amazon SageMaker uses: (1) training code and (2) training dataset to train the model.

**S3 Bucket**

Amazon SageMaker saves the trained model artifacts in an S3 bucket.

* Amazon Simple Storage Service (Amazon S3) is a storage service that allows enterprises/individuals to store and protect any amount of data.
* Amazon S3 is extremely easy to use and allows enterprises to organize their data and configure finely-tuned access controls.
* Amazon S3 extremely durable to 99.999999999% (11 9’s).
* Amazon S3 is 99.9% available.
* Amazon S3 offers numerous enhanced features such as:

(1) Scalability

(2) Data availability

(3) Security

(4) Performance

* Amazon Simple Storage Service (Amazon S3) is built to be extremely simple and robust.
* Amazon S3 allows customers to store data in buckets or directories (much like folders).
* A bucket is a container for objects stored in Amazon S3. Objects are contained in buckets.
* Each of the buckets will have global (universal) unique name. So, you cannot have the same bucket name as somebody else!
* You can store an infinite amount of data in a bucket in which each object can contain up to 5 TB of data.
* S3 allows anyone to collect, store and analyze the data from anywhere and in any amount.
* Data is stored on 3 different availability zones to ensure data protection.
* S3 data is fully encryption to ensure compliance and security.

**Amazon Elastic Compute Cloud (EC2)** offers resizable compute capacity in the cloud.

* AWS EC2 can be used to acquire, configure and scale capacity in a very easy fashion.
* EC2 is a service that allows you to simply rent a server in the cloud.
* EC2 offers 7x fewer downtime hours than the next largest cloud provider.
* EC2 covers 22 regions and 69 availability zones all over the world.
* Note: EC2 is a compute service which is NOT serverless (lambda is serverless).

# Setting up an AWS environment

Let's set up an AWS account to start our cloud computing journey. If you already have an AWS account, you can skip this section and move on to the next chapter.

Please go to <https://portal.aws.amazon.com/billing/signup> and follow the instructions to sign up for an account. You will receive a phone call and will need to enter a verification code on the phone keypad as part of the process.

When you first create a new AWS account and log in with your email and password, you will be logged in as an account root user. However, it is best practice to create a new IAM user for yourself with the AdministratorAccess policy while logged in as the root user, and then swiftly log out and log in again as the IAM user that you just created. The root user credential shall only be used to perform limited account and service management tasks and shall not be used to develop your cloud applications. You should securely store the root user credential and lock it away from any other people accessing it.

Here are the steps to create an IAM user:

1. Go to the IAM console, select **Users** on the left panel, and then click on the **Add user** button:

Figure 1.2 – Adding an IAM user in the IAM console


Figure 1.2 – Adding an IAM user in the IAM console

1. Next, enter a name in **User name** and check the boxes for **Programmatic access** and **AWS Management Console access**. For the password fields, you can leave the default options. Hit the **Next: Permissions** button to proceed:

Figure 1.3 – Creating a user name and password for an IAM user 


Figure 1.3 – Creating a user name and password for an IAM user

1. On the next page, choose **Add user to group** under **Set permissions**. In a new account, you do not have any groups. You should click on **Create group**.
2. In the pop-up dialog, enter Administrator in **Group name**, select AdministratorAccess in the policy list, and hit the **Create group** button:

Figure 1.4 – Creating an IAM group with AdministratorAccess


Figure 1.4 – Creating an IAM group with AdministratorAccess

1. The dialog will close. Make sure the new administrator is selected and hit **Next: Tags**. You can optionally add key-value pair tags to the IAM user. Hit **Next: Review** to review the configuration. Hit **Create user** when everything is correct.

You will see the following information. Please note down the sign-in URL for easy console access, **Access key ID** and **Secret access key** for programmatic access, and the one-time password. You can also download the credential as a CSV file by clicking the **Download .csv** button:

Figure 1.5 – A new IAM user is created


Figure 1.5 – A new IAM user is created

1. After the IAM user creation, you can sign in to your AWS account with the sign-in URL and your IAM user. When you first sign in, you will need to provide the automatically generated password and then set up a new one. Now, you should note in the top-right corner that you are logged in as your newly created IAM user instead of the root user:



Figure 1.6 – Confirm your newly created credentials

If you are new to AWS, don't worry about the cost of trying out AWS. AWS offers a free tier for more than 100 services based on the consumption of the service and/or within a 12-month period. The services we are going to use throughout the book, such as an S3 bucket and Amazon SageMaker, have a free tier for you to learn the skills without breaking the bank. The following table is a summary of the free tier for the services that are going to be covered in this book:

Figure 1.7 – Notable free trial offers from AWS
