AWS Certified AI Practitioner – Study Guide and Summary

Here is a study guide summarizing the information in our Mastering AWS Certified AI Practitioner AIF-C01 – Hands On! Course, organized by AWS services.

While this guide will help you review for the exam, it is not a replacement for the course itself! Reviewing the course videos, slides, and activities will be essential in passing this challenging exam. You need more than the trivia below; you need to know how these tools fit together to build applications, and that's covered more in the course.

Good luck on the exam!

-Frank Kane

Amazon SageMaker

- **SageMaker** is a machine learning service that provides tools to build, train, and deploy machine learning models at scale.
- **SageMaker** handles the entire machine learning workflow.
- **SageMaker Training & Deployment** includes the client app, model deployment/hosting, model training, S3 model artifacts, S3 training data, inference code image, training code image, and SageMaker ECR.
- **SageMaker Notebooks** can direct the process and are spun up from the console on EC2 instances.
- **SageMaker Notebooks** provide S3 data access, use Scikit_learn, Spark, and TensorFlow, and have a wide variety of built-in models, as well as the ability to spin up training instances and deploy trained models to make predictions.
- Data for **SageMaker** usually comes from S3.

- **SageMaker** ingests data from Athena, EMR, Redshift, and Amazon Keyspaces DB.
- Apache Spark integrates with **SageMaker**.
- Scikit_learn, NumPy, and pandas can all be used in a **SageMaker** notebook.
- **SageMaker Processing** jobs copy data from S3, spin up a processing container (either SageMaker built-in or user-provided), and then output processed data to S3.
- To train on **SageMaker**, create a training job with the URL of the S3 bucket that has the training data, ML compute resources, URL of the S3 bucket for output, and the ECR path to the training code.
- **SageMaker** training options include built-in training algorithms, Spark MLLib, custom Python Tensorflow/MXNet code, PyTorch, Scikit-Learn, RLEstimator, XGBoost, Hugging Face, Chainer, your own Docker image, or an algorithm purchased from the AWS Marketplace.
- To deploy a trained **SageMaker** model, save your trained model to S3 and deploy either with a persistent endpoint for making individual predictions on demand or with SageMaker Batch Transform to get predictions for an entire dataset.
- Some cool **SageMaker** options include Inference Pipelines for more complex processing, SageMaker Neo for deploying to edge devices, Elastic Inference for accelerating deep learning models, automatic scaling to increase the number of endpoints as needed, and Shadow Testing, which evaluates new models against the currently deployed model to catch errors.
- **SageMaker Studio** is a visual IDE for machine learning that integrates many features.
- You can create and share Jupyter notebooks with **SageMaker Studio**.
- **SageMaker Studio** lets you switch between hardware configurations without having to manage the infrastructure.
- **SageMaker Experiments** organizes, captures, compares, and searches your ML jobs.
- **SageMaker Debugger** saves the internal model state periodically.
- **SageMaker Debugger** saves gradients and tensors over time as a model is trained.
- **SageMaker Debugger** lets you define rules for detecting unwanted conditions while training.
- **SageMaker Debugger** runs a debug job for each rule you configure.
- **SageMaker Debugger** logs and fires a CloudWatch event when the rule is hit.
- **SageMaker Debugger** includes SageMaker Studio Debugger dashboards and auto-generated training reports.

- **SageMaker Debugger**'s built-in rules let you monitor system bottlenecks, profile model framework operations, and debug model parameters.
- **SageMaker Debugger** supports TensorFlow, PyTorch, MXNet, XGBoost, and the SageMaker generic estimator.
- **SageMaker Debugger**'s API's, available on GitHub, allow you to construct hooks and rules for CreateTrainingJob and DescribeTrainingJob API's.
- The **SageMaker Debugger** SMDebug client library lets you register hooks for accessing training data.
- **SageMaker Debugger** includes newer features: SageMaker Debugger Insights Dashboard and Debugger ProfilerRule (ProfilerReport).
- The **SageMaker Debugger** ProfilerReport includes hardware system metrics (CPUBottlenck, GPUMemoryIncrease, etc.) and framework metrics (MaxInitializationTime, OverallFrameworkMetrics, StepOutlier).
- **SageMaker Debugger** includes built-in actions to receive notifications or stop training: StopTraining(), Email(), or SMS() in response to Debugger Rules, which sends notifications through SNS.
- **SageMaker Debugger** profiles system resource usage and training.
- **SageMaker Autopilot** automates algorithm selection, data preprocessing, model tuning, and all infrastructure.
- **SageMaker Autopilot** takes care of all the trial and error for you.
- **SageMaker Autopilot** is also called AutoML.
- The **SageMaker Autopilot** workflow loads data from S3 for training, selects your target column for prediction, automatically creates the model, creates a model notebook that is available for visibility and control, and creates a model leaderboard that is a ranked list of recommended models from which you can choose.
- With **SageMaker Autopilot** you can deploy and monitor the model, refining it in a notebook if needed.
- **SageMaker Autopilot** can include human guidance, with or without code, in SageMaker Studio or AWS SDK's.
- **SageMaker Autopilot** supports binary classification, multiclass classification, and regression problems.
- **SageMaker Autopilot** algorithm types include Linear Learner, XGBoost, Deep Learning (MLP's), and Ensemble mode.
- **SageMaker Autopilot** requires tabular CSV or Parquet data.
- **SageMaker Autopilot** has three training modes: HPO, Ensembling, and Auto.

- HPO (Hyperparameter optimization) in **SageMaker Autopilot** selects the most relevant algorithms for your dataset, Linear Learner, XGBoost, or Deep Learning.
- HPO in **SageMaker Autopilot** selects the best range of hyperparameters to tune your models and runs up to 100 trials to find optimal hyperparameters in the range.
- HPO in **SageMaker Autopilot** uses Bayesian optimization if the dataset is less than 100MB and multi-fidelity optimization if it is greater than 100MB.
- HPO in **SageMaker Autopilot** implements early stopping if a trial is performing poorly.
- The ensembling training mode in **SageMaker Autopilot** trains several base models using the AutoGluon library, including more tree-based and neural network algorithms.
- The ensembling training mode in **SageMaker Autopilot** runs 10 trials with different model and parameter settings and then models are combined with a stacking ensemble method.
- The Auto training mode in **SageMaker Autopilot** uses HPO if the dataset is greater than 100MB and ensembling if the dataset is less than 100MB.
- **SageMaker Autopilot** needs to be able to read the size of your dataset, or it will default to HPO.
- If the S3 bucket is hidden inside a VPC, the S3DataType is ManifestFile, or the S3Uri contains more than 1000 items, then **SageMaker Autopilot** will default to HPO.
- **SageMaker Autopilot** Explainability provides transparency on how models arrive at predictions.
- **SageMaker Autopilot** Explainability integrates with SageMaker Clarify and uses feature attribution, which uses SHAP Baselines/Shapley Values, which uses research from cooperative game theory, to assign each feature an importance value for a given prediction.
- **SageMaker Model Monitor** sends alerts on quality deviations on your deployed models through CloudWatch.
- **SageMaker Model Monitor** visualizes data drift.
- **SageMaker Model Monitor** detects anomalies and outliers as well as new features.
- **SageMaker Model Monitor** requires no code.
- **SageMaker Model Monitor** integrates with SageMaker Clarify to detect potential bias, such as imbalances across different groups, ages, or income brackets.
- You can use **SageMaker Model Monitor** and **SageMaker Clarify** together to monitor for bias and get alerts for new potential bias through CloudWatch.
- **SageMaker Clarify** can help explain model behavior and help you understand which features contribute the most to your predictions.
- **SageMaker Model Monitor** stores data in S3, where it is secured.

- **SageMaker Model Monitor** monitoring jobs are scheduled through a Monitoring Schedule.
- **SageMaker Model Monitor** sends metrics to CloudWatch.
- **SageMaker Model Monitor** CloudWatch notifications can trigger alarms.
- **SageMaker Model Monitor** alarms prompt you to take corrective action, such as retraining the model or auditing the data.
- **SageMaker Model Monitor** integrates with Tensorboard, QuickSight, and Tableau or you can use visualizations within SageMaker Studio.
- **SageMaker Model Monitor** monitors for drift in data quality, drift in model quality (accuracy, etc.), bias drift, and feature attribution drift.
- **SageMaker Model Monitor** monitors drift in data quality relative to a baseline that you create.
- **SageMaker Model Monitor** determines data "quality" based on the statistical properties of the features.
- **SageMaker Model Monitor** monitors drift in model quality with a model quality baseline.
- **SageMaker Model Monitor** monitoring of drift in model quality can integrate with Ground Truth labels.
- **SageMaker Model Monitor**'s feature attribution drift monitoring is based on the Normalized Discounted Cumulative Gain (NDCG) score, which compares feature ranking of training data and live data.
- **SageMaker** Deployment Safeguards include Deployment Guardrails and Shadow Tests.
- **SageMaker** Deployment Guardrails control shifting traffic to new models (Blue/Green Deployments) for asynchronous or real-time inference endpoints.
- You can shift traffic to new models in **SageMaker** all at once, shifting everything, monitoring, and then terminating the blue fleet.
- You can shift traffic to new models in **SageMaker** using canary deployments, shifting a small portion of traffic and then monitoring.
- You can shift traffic to new models in **SageMaker** using linear deployments, shifting traffic in linearly spaced steps.
- **SageMaker** Deployment Guardrails include auto-rollbacks.
- **SageMaker** Shadow Tests compare the performance of a shadow variant and the production variant.
- You can use **SageMaker** Shadow Tests to monitor in the SageMaker console and decide when to promote the shadow variant.

- **SageMaker JumpStart** has one-click models and algorithms from model zoos, with over 150 open source models in NLP, object detection, image classification, etc.
- **SageMaker Data Wrangler** imports, transforms, analyzes, and exports data within SageMaker Studio.
- **SageMaker Feature Store** finds, discovers, and shares features in Studio, operating in online (low latency) or offline (for training or batch inference) modes, with features organized into Feature Groups.
- **SageMaker Edge Manager** is a software agent for edge devices.
- **SageMaker Edge Manager** uses models optimized with SageMaker Neo.
- **SageMaker Edge Manager** collects and samples data for monitoring, labeling, and retraining.
- **SageMaker** includes asynchronous Inference endpoints.

Amazon SageMaker Feature Store

- **Amazon SageMaker Feature Store** is a purpose-built repository that makes it easy to store, discover, and share machine learning features.
- **Amazon SageMaker Feature Store** organizes your data with a record identifier, feature name, event time, Feature Group, and Feature Store.
- **Amazon SageMaker Feature Store** data ingestion happens either through streaming or in batches.
- **Amazon SageMaker Feature Store** has an online store, offline store, model, Amazon Kinesis, Amazon Managed Streaming for Apache Kafka, and Spark, Data Wrangler, and other streaming access through the PutRecord/GetRecord API's.
- You can access **Amazon SageMaker Feature Store** data in batches through the offline S3 store and use anything that uses S3, like Athena or Data Wrangler.
- **Amazon SageMaker Feature Store** automatically creates a Glue Data Catalog for you.
- **Amazon SageMaker Feature Store** is encrypted at rest and in transit and works with KMS customer master keys.
- **Amazon SageMaker Feature Store** uses fine-grained access control with IAM and may also be secured with AWS PrivateLink.

Amazon SageMaker ML Lineage Tracking

- **SageMaker ML Lineage Tracking** creates and stores your ML workflow (MLOps), keeps a running history of your models, and provides tracking for auditing and compliance.
- **SageMaker ML Lineage Tracking** automatically or manually creates tracking entities.

- **SageMaker ML Lineage Tracking** integrates with AWS Resource Access Manager for cross-account lineage.
- **SageMaker ML Lineage Tracking** entities include trial component, trial, experiment, context, action, artifact, and association.
- A trial component is a processing, training, or transform job.
- A trial is a model composed of trial components.
- An experiment is a group of trials for a given use case.
- The context is a logical grouping of entities.
- An action is a workflow step or model deployment.
- An artifact is an object or data, such as an S3 bucket or an image in ECR.
- An association connects entities and has an optional AssociationType, including ContributedTo, AssociatedWith, DerivedFrom, Produced, or SameAs.
- You can query lineage entities with the LineageQuery API from Python, which is part of the Amazon SageMaker SDK for Python.
- You can use **SageMaker ML Lineage Tracking** to find all models, endpoints, etc. that use a given artifact.
- **SageMaker ML Lineage Tracking** can produce a visualization, although it requires an external Visualizer helper class.

Amazon SageMaker Data Wrangler

- **SageMaker Data Wrangler** has a visual interface in SageMaker Studio that lets you prepare data for machine learning.
- **SageMaker Data Wrangler** imports, visualizes, and transforms data.
- **SageMaker Data Wrangler** has over 300 transformations to choose from or you can integrate your own custom transformations with pandas, PySpark, or PySpark SQL.
- **SageMaker Data Wrangler** uses "Quick Model" to train your model with your data and measure the results.
- **SageMaker Data Wrangler** imports data from Amazon Simple Storage Service (Amazon S3), Amazon Athena, Amazon Redshift, AWS Lake Formation, Amazon SageMaker Feature Store, Amazon SageMaker Processing, Amazon SageMaker Pipelines, a notebook, or JDBC (Databricks, SaaS).
- **SageMaker Data Wrangler** exports data flow.

Amazon SageMaker Canvas

- **SageMaker Canvas** provides no-code machine learning for business analysts.

- **SageMaker Canvas** lets you upload CSV data, select a column to predict, build the model, and then make predictions.
- **SageMaker Canvas** can also join datasets, and performs classification or regression.
- **SageMaker Canvas** automatically cleans data, handling missing values, outliers, and duplicates.
- **SageMaker Canvas** shares models and datasets with SageMaker Studio.

Amazon Bedrock

- **Amazon Bedrock** is an API for generative AI Foundation Models.
- **Amazon Bedrock** invokes chat, text, or image models that are pre-built, fine-tuned, or your own models.
- **Amazon Bedrock** bills you for third-party models through AWS, using their pricing.
- **Amazon Bedrock** supports RAG (Retrieval-Augmented Generation).
- **Amazon Bedrock** supports LLM agents and is serverless.
- **Amazon Bedrock** can integrate with SageMaker Canvas.
- The **Amazon Bedrock API** has endpoints for bedrock, bedrock-runtime, and bedrock-agent.
- The bedrock endpoint manages, deploys, and trains models.
- The bedrock-runtime endpoint performs inference against models, executes prompts, and generates embeddings.
- The bedrock-runtime endpoint uses Converse, ConverseStream, InvokeModel, and InvokeModelWithReponseStream.
- The bedrock-agent endpoint manages, deploys, and trains LLM agents and knowledge bases.
- The bedrock-agent-runtime performs inference against agents and knowledge bases.
- The bedrock-agent-runtime uses InvokeAgent, Retrieve, and RetrieveAndGenerate.
- You must request access before using any base model in **Bedrock**.
- **Bedrock** will approve requests for Amazon (Titan) models immediately.
- Third-party models in **Bedrock** might require you to submit additional information and you will be billed at the third party's rates through AWS.
- Approval for third-party models only takes a few minutes.
- **Bedrock** provides "playground" environments for chat, text, and images.
- You must have model access before using **Bedrock** playgrounds.

- **Bedrock** playgrounds are useful for evaluating your custom or imported models.
- Fine-tuning adapts an existing large language model to your specific use case with additional training using your own data.
- Fine-tuning eliminates the need to build up a large conversation to get the desired result, saving on tokens in the long run.
- You can fine-tune a fine-tuned model.
- Applications for fine-tuning include chatbots with a certain personality, style, or objective; training with data that is more recent than the data the LLM has; training with proprietary data; and specific applications, such as classification and evaluating truth.
- **Bedrock**'s fine-tuning feature is called "Custom Models."
- You can fine-tune Titan, Cohere, and Meta models in **Bedrock**.
- To fine-tune text models, provide labeled training pairs of prompts and completions, such as questions and answers.
- Upload training data into S3 to fine-tune models.
- For image models, provide pairs of image S3 paths to image descriptions (prompts), which are used for text-to-image or image-to-embedding models.
- Use a VPC and PrivateLink for sensitive training data.
- Fine-tuning in **Bedrock** can be expensive.
- You can use the resulting custom model like any other model.
- You can import Custom Models from SageMaker or from a model in S3 in Hugging Face weights format.
- "Continued Pre-Training" is like fine-tuning but with unlabeled data.
- "Continued Pre-Training" familiarizes the model with your business documents or any other data, which means you are including extra data in the model itself so that you do not need to include it in the prompts.
- Retrieval Augmented Generation (RAG) is like an open-book exam for LLMs, allowing you to query an external database instead of relying on the LLM.
- With RAG, you can incorporate the answers from a database query into the prompt for the LLM to work with or you can use tools and functions to incorporate the search into the LLM.
- RAG is a faster and cheaper way to incorporate new or proprietary information into "GenAI" than fine-tuning.
- Updating information with RAG is just a matter of updating the database.
- RAG leverages "semantic search" with vector stores.

- RAG prevents "hallucinations" when you ask the model about something it wasn't trained on.
- Technically, you aren't "training" a model with RAG.
- One example of RAG is "RAG with a Graph database" in the OpenAI Cookbook.
- Most examples of RAG use a Vector database.
- Elasticsearch/OpenSearch can function as a vector DB.
- An embedding is a large vector associated with your data, which is a point in multi-dimensional space.
- Embeddings are computed such that items that are similar to each other are close together in that space.
- You can use embedding base models, like Titan, to compute embeddings en masse.
- You store embeddings in a vector database.
- A vector database stores your data and their computed embedding vectors.
- To retrieve information from a vector database, compute an embedding vector for the item you want to search for, query the vector database for the top items that are closest to that vector, and then you get back the top-N most similar things (K-Nearest Neighbor) in a "vector search."
- Examples of vector databases in AWS include Amazon OpenSearch Service (provisioned), Amazon OpenSearch Serverless, Amazon MemoryDB, the pgvector extension in Amazon Relational Database Service (Amazon RDS) for PostgreSQL, the pgvector extension in Amazon Aurora PostgreSQL-Compatible Edition, and Amazon Kendra.
- You can upload your own documents or structured data through S3, maybe with a JSON schema, into **Bedrock** "Knowledge Bases," or you can use other sources, such as a web crawler, Confluence, Salesforce, or SharePoint.
- **Bedrock** Knowledge Bases require an embedding model for which you must have obtained model access.
- Currently, you can use Cohere or Amazon Titan for **Bedrock** Knowledge Bases and you can control the vector dimension.
- You can use a serverless OpenSearch instance by default, MemoryDB, Aurora, MongoDB Atlas, Pinecone, or Redis Enterprise Cloud for **Bedrock** Knowledge Bases.
- You can control the "chunking" of data (how many tokens are represented by each vector) in **Bedrock** Knowledge Bases.
- You can use Knowledge Bases to "chat with your document" (automatic RAG) or incorporate it into an agent.

- You can also integrate a Knowledge Base into an application directly using "Agentic RAG."
- **Amazon Bedrock Guardrails** filters content for prompts and responses for text foundation models.
- **Amazon Bedrock Guardrails** includes word filtering, topic filtering, profanities, PII removal (or masking), and Contextual Grounding Check, which helps prevent hallucination, measures "grounding" (how similar the response is to the contextual data received), and relevance (of response to the query).
- You can incorporate **Amazon Bedrock Guardrails** into agents and knowledge bases.
- You can configure the **Amazon Bedrock Guardrails** "blocked message" response.
- LLM agents give tools to your LLM.
- An LLM agent has a memory, an ability to plan how to answer a request, and tools it can use in the process.
- In practice, the "memory" is just the chat history and external data stores and the "planning module" is guidance given to the LLM on how to break down a question into sub-questions that the tools might be able to help with.
- Prompts associated with each tool guide the agent on how to use its tools.
- In a more practical approach to LLM agents, "tools" are functions provided to a tools API, which in **Bedrock**, can be a Lambda function.
- Prompts guide the LLM on how to use the tools.
- Tools may access outside information, retrievers, other Python modules, services, etc.
- To get an agent to know which tools to use, start with a foundational model to work with.
- "Action Groups" define tools in **Bedrock**.
- A prompt tells the foundational model when to use a tool.
- You must define the parameters that your tool expects.
- You need to define a tool's name, description, type, and whether it is required or not.
- The description is used by the LLM to extract the required information from the user prompt.
- You can allow the foundational model to go back and ask the user for missing information that it needs for a tool.
- You can use OpenAI-style schemas or visually with a table in the Bedrock UI to allow the foundational model to go back and ask the user for missing information that it needs for a tool.
- You can associate knowledge bases with agents.
- A prompt tells the LLM when to use an associated knowledge base.

- This is called "agentic RAG" because RAG is another tool.
- The "Code Interpreter" allows the agent to write its own code to answer questions or produce charts.
- You deploy a **Bedrock** agent by creating an "alias" for the agent, which creates a deployed snapshot.
- **Bedrock** agents can use on-demand throughput (ODT), which allows the agent to run at quotas set at the account level.
- **Bedrock** agents can use provisioned throughput (PT), which allows you to purchase an increased rate and number of tokens for your agent.
- Your application can use the InvokeAgent request with your alias ID and an Agents for Amazon Bedrock Runtime Endpoint.
- **Bedrock** includes more features, such as imported models, model evaluation, provisioned throughput, watermark detection, and Bedrock Studio.
- Watermark detection in **Bedrock** detects if an image was generated by Titan.
- **Bedrock Model Invocation Logging** lets you collect full request data, response data, and metadata.
- You can output **Bedrock Model Invocation Logging** data to CloudWatch or S3 within the same account and region.
- You can use **Bedrock Model Invocation Logging** to log Converse, ConverseStream, InvokeModel, and InvokeModelWithResponseStream.
- **Bedrock** pricing varies by the foundation model and has on-demand, batch, and provisioned throughput pricing.
- On-demand pricing for **Bedrock** is pay-as-you-go with no time-based term commitments.
- On-demand pricing for **Bedrock** charges by all input tokens processed or generated, by image generated, and supports cross-region inference for some models to increase throughput.
- Batch pricing for **Bedrock** requires you to provide a set of prompts as input and then you get the output responses in S3.
- Batch pricing is 50% lower than on-demand pricing.
- Provisioned throughput for **Bedrock** provisions resources to meet your requirements in exchange for a time-based term commitment.
- Provisioned throughput for **Bedrock** charges by the hour with a 1- to 6-month commitment and is required for custom models.

- **Bedrock** model customization charges for tokens processed in fine-tuning, training tokens multiplied by the number of epochs, and storage costs.
- **Bedrock** security uses TLS 1.2 encryption, SSL in-transit, and encrypts model customization jobs and artifacts at rest.
- **Bedrock** security uses KMS or AWS-owned keys at rest and S3 training, validation, and output data may be encrypted with SSE-KMS.
- **Bedrock** network security uses VPC's to contain Bedrock.
- You can set up Amazon S3 VPC Endpoints to allow your model access to specific S3 buckets.
- You can use IAM and bucket policies to restrict access to S3 files.
- You can use AWS PrivateLink to establish a private connection to your data.

Amazon CodeWhisperer / Amazon Q Developer

- **Amazon CodeWhisperer** / **Amazon Q Developer** is an "AI coding companion" that supports Java, JavaScript, Python, TypeScript, and C# in Visual Studio, JetBrains, JupyterLab, SageMaker, and Lambda.
- **Amazon CodeWhisperer** / **Amazon Q Developer** provides real-time code suggestions.
- **Amazon CodeWhisperer** / **Amazon Q Developer** suggests blocks of code into your IDE after you write a comment describing what you want.
- **Amazon CodeWhisperer** / **Amazon Q Developer** is based on LLMs trained on billions of lines of code from Amazon and open source code.
- **Amazon CodeWhisperer** / **Amazon Q Developer** security scans analyze code for vulnerabilities in Java, JavaScript, and Python.
- **Amazon CodeWhisperer** / **Amazon Q Developer**'s reference tracker flags suggestions that are similar to open source code and provides annotations for proper attribution.
- **Amazon CodeWhisperer** / **Amazon Q Developer** avoids bias by filtering out code suggestions that might be biased or unfair.
- **Amazon CodeWhisperer** / **Amazon Q Developer** integrates with AWS services to suggest code for interfacing with AWS APIs, including EC2, Lambda, and S3.
- **Amazon CodeWhisperer** / **Amazon Q Developer** transmits all content with TLS, encrypting data in transit and at rest, although Amazon can mine your data for individual plans.
- You can customize **Amazon CodeWhisperer** / **Amazon Q Developer** by providing your organization's code as a reference (using RAG).

Amazon Q Business

- **Amazon Q Business** is an AI assistant for your enterprise information that uses RAG.

- **Amazon Q Business** has a web interface or API.
- You can embed **Amazon Q Business** into Slack or MS Teams.
- **Amazon Q Business** answers questions, generates content, creates summaries, and completes tasks.
- **Amazon Q Business** has over 40 built-in connectors, both cloud and on-prem, such as Salesforce, Oracle, S3, Jira, ServiceNow, and Zendesk.
- **Amazon Q Business** uses IAM Identity Center and SAML 2.0 to control user access and includes guardrails.
- **Amazon Q Business** is fully managed and you can't pick the underlying foundational model.
- **Amazon Q Business** doesn't use your data to train Q.
- **Amazon Q Business** encrypts your data, conversations, and feedback with KMS.
- **Amazon Q Business** saves chat history for one month and supports PDF, CSV, DOCX, HTML, JSON, and PPT data formats.
- You can use the Amazon Q SDK to create custom connectors for **Amazon Q Business**, allowing you to incorporate other data.
- You can build your own generative AI apps with **Amazon Q Apps** based on your organization's data.
- You can use **Amazon Q Apps Creator** to build apps by entering, "build an app that..."
- You can use "cards" to control the app's layout for input and output in **Amazon Q Apps**.
- Input cards can be for files or documents to upload.
- You publish **Amazon Q Apps** in your "Amazon Q Apps library."
- You can customize **Amazon Q Apps** to create a copy of the app that you can modify.
- **Amazon Q Business** pricing includes "Lite" for $3/user/month for just chat and "Pro" for $20/user/month for Amazon Q Apps, Amazon Q in QuickSight, and custom plugins.
- **Amazon Q Business** also has index pricing, which charges by the "unit" with 100 hours of usage included per month or 20,000 documents or 200MB.
- Index pricing costs $0.264/hour/unit.

Amazon QuickSight

- **Amazon QuickSight** is a fast, easy, cloud-powered business analytics service that is serverless.
- **Amazon QuickSight** allows all employees in an organization to build visualizations, perform ad hoc analysis, and quickly get business insights from data on any device at any time.

- **Amazon QuickSight** data sources include Redshift, Aurora/RDS, Athena, EC2-hosted databases, files (S3 or on-premises, including Excel, CSV, TSV, or common or extended log format), AWS IoT Analytics, and data preparation that allows limited ETL.
- Use cases for **Amazon QuickSight** include interactive ad hoc exploration/visualization of data, dashboards and KPIs, and analyzing/visualizing data from logs in S3, on-premises databases, AWS (RDS, Redshift, Athena, S3), SaaS applications, such