

AWS CERTIFIED

AI PRACTITIONER

AIF-C01

THE ONE-SHOT COMPLETE STUDY GUIDE

Every concept. Every domain. Every service — explained lucidly.

Author: **Wreetojyoti Ray**

Senior Data & AI Consultant | Kolkata, India

© 2025 Wreetojyoti Ray — All Rights Reserved

★ How to Use This Guide

This guide is your single, exhaustive reference for the **AWS Certified AI Practitioner (AIF-C01)** exam. It covers all five official exam domains in full depth — from foundational AI/ML theory to the latest generative AI capabilities on AWS. Each section is self-contained, elaborated with real-world context, and enriched with exam tips, comparison tables, and key-concept callouts.

Attribute	Detail
■ Exam Code	AIF-C01
■ Exam Duration	120 minutes
■ Question Format	Multiple choice, multiple response, ordering, matching, case study
■ Passing Score	700 / 1000
■ Exam Fee	USD 150 (standard)
■ Validity	3 years from date of certification
■ Recommended Exp.	6+ months of AWS usage; familiarity with AI/ML concepts

Exam Domain Weightings (Official):

Domain	Topic	Weight
Domain 1	Fundamentals of AI and ML	20%
Domain 2	Fundamentals of Generative AI	24%
Domain 3	Applications of Foundation Models	28%
Domain 4	Guidelines for Responsible AI	14%
Domain 5	Security, Compliance, and Governance for AI Solutions	14%

EXAM TIP: Domain 3 (Applications of Foundation Models) carries the highest weight at 28%. Prioritise prompt engineering, RAG, agents, and model customisation deeply.

1 Fundamentals of AI and ML

1.1 What Is Artificial Intelligence?

Artificial Intelligence (AI) is the science and engineering of creating systems that can perform tasks which normally require human intelligence — such as understanding language, recognising patterns, making decisions, and solving problems. AI encompasses a vast range of techniques, from rule-based expert systems to modern deep learning architectures.

AI is not a single technology; it is an umbrella term. Think of it as a layered hierarchy: at the outermost level sits AI (the broad concept), inside it lives **Machine Learning** (systems that learn from data), and inside ML lives **Deep Learning** (ML using multi-layer neural networks). Generative AI sits within Deep Learning as a specialisation focused on creating new content.

KEY CONCEPT: The AWS exam often tests the AI → ML → Deep Learning → Generative AI hierarchy. Know the distinctions clearly.

1.2 Machine Learning — The Engine of Modern AI

Machine Learning (ML) is a subset of AI where algorithms learn patterns directly from data, improving their performance with experience — without being explicitly programmed for each task. Instead of writing hand-crafted rules, you feed data into an algorithm and it discovers the rules itself.

Core ML Paradigms:

1. Supervised Learning

The algorithm learns from labelled examples — input–output pairs. The goal is to learn a mapping function from input X to output Y. After training, the model predicts Y for unseen X values.

Sub-type	Description
Classification	Predict a discrete class label. Example: spam vs. not-spam, tumour malignant vs. benign.
Regression	Predict a continuous numerical value. Example: house price prediction, sales forecasting.
Binary	Two classes only (yes/no, 0/1).
Multi-class	More than two classes (cat, dog, bird).
Multi-label	Assign multiple labels to one instance (a photo tagged as 'beach', 'sunset', 'family').

2. Unsupervised Learning

No labels provided. The algorithm discovers hidden structure, patterns, or groupings in the data independently. Useful when you don't know what categories exist upfront.

Technique	Description
Clustering	Group similar data points together. K-Means, DBSCAN, Hierarchical clustering.
Dimensionality Reduction	Reduce the number of features while preserving information. PCA, t-SNE, UMAP.
Association Rules	Find co-occurrence patterns. Market basket analysis — 'customers who bought X also bought Y.'
Anomaly Detection	Identify rare, unusual data points — fraud detection, network intrusion.

3. Semi-Supervised Learning

A hybrid approach using a small amount of labelled data combined with a large amount of unlabelled data. Very practical because labelling data is expensive and time-consuming.

4. Reinforcement Learning (RL)

An **agent** learns by interacting with an **environment**. It takes **actions**, receives **rewards** (positive for good actions, negative for bad), and learns a **policy** — a strategy to maximise cumulative reward over time. No labelled data needed; learning is through trial and error.

Examples: game-playing AI (AlphaGo), robotics, personalised recommendations, autonomous driving.

EXAM TIP: Amazon SageMaker RL uses open-source frameworks like Ray RLLib and Coach. AWS DeepRacer uses RL to train an autonomous race car model — a classic AWS RL example.

5. Self-Supervised Learning

The model creates its own supervisory signal from unlabelled data. This is how large language models (LLMs) are pre-trained — by predicting the next token in a sequence (the next word serves as the label). No human labelling required at scale, enabling training on internet-scale text.

1.3 The ML Workflow — From Data to Deployment

A complete ML pipeline is rarely a straight line; it involves cycles of iteration, debugging, and refinement. Understanding each stage is critical for the exam.

Stage 1: Problem Formulation

Define the business problem. Determine if ML is the right solution. Identify the ML task type (classification, regression, etc.). Define success metrics aligned with business goals (accuracy, F1, AUC-ROC, RMSE).

Stage 2: Data Collection

Gather raw data from databases, APIs, sensors, web scraping, or existing datasets. Consider data volume (how much?), variety (what types?), and veracity (how trustworthy?). AWS services: S3, Kinesis, Glue, DMS, AppFlow.

Stage 3: Exploratory Data Analysis (EDA)

Understand the data distribution, identify outliers, missing values, class imbalances, correlations, and patterns. Visualise with histograms, box plots, scatter plots, heatmaps. AWS: SageMaker Studio, SageMaker Canvas.

Stage 4: Data Pre-processing & Feature Engineering

Handle missing values (imputation, deletion). Encode categoricals (one-hot, label encoding). Scale numerics (standardisation, normalisation). Engineer new features. Split data into train/validation/test sets. AWS: SageMaker Data Wrangler, SageMaker Processing, AWS Glue.

Stage 5: Model Selection & Training

Choose algorithm(s) based on problem type, data size, interpretability needs. Train on the training set. Tune hyperparameters. AWS: SageMaker Training Jobs, SageMaker Autopilot, SageMaker JumpStart.

Stage 6: Model Evaluation

Evaluate on held-out test set using appropriate metrics. Check for overfitting/underfitting. Analyse confusion matrix, precision/recall/F1, AUC-ROC for classification; MAE/MSE/RMSE/R² for regression.

Stage 7: Model Deployment

Package model and deploy as a real-time endpoint, batch transform job, or serverless endpoint. AWS: SageMaker Endpoints, SageMaker Batch Transform, SageMaker Serverless Inference, Lambda.

Stage 8: Monitoring & Maintenance

Monitor model performance in production. Detect data drift and model degradation. Retrain as needed. AWS: SageMaker Model Monitor, SageMaker Clarify, CloudWatch.

1.4 Key ML Concepts and Terminology

Overfitting vs. Underfitting

Overfitting occurs when a model learns the training data *too well* — including noise and random fluctuations — resulting in poor generalisation to unseen data. The model has memorised rather than learned. Signs: very high training accuracy, much lower validation/test accuracy.

Underfitting occurs when a model is too simple to capture the underlying patterns in the data. Signs: poor performance on both training and test sets. The model hasn't learned enough.

Issue	Detail
Overfitting cause	Model too complex, too little data, too many features
Underfitting cause	Model too simple, too few features, insufficient training
Overfit remedy	Regularisation (L1/L2), Dropout, more training data, cross-validation, pruning
Underfit remedy	More complex model, additional features, reduce regularisation, train longer

KEY CONCEPT: Regularisation adds a penalty term to the loss function. L1 (Lasso) drives some weights to exactly zero (feature selection). L2 (Ridge) shrinks all weights small but rarely to zero.

Bias vs. Variance Trade-off

Bias is the error from wrong assumptions in the learning algorithm — the model systematically misses the true relationship. High bias = underfitting. **Variance** is the error from sensitivity to small fluctuations in the training set — the model changes a lot when trained on different subsets. High variance = overfitting. The goal is to find the sweet spot that minimises total error.

Cross-Validation

A technique to assess how well a model generalises, using multiple train/test splits. **k-Fold cross-validation** splits data into k equal folds, trains on k-1 folds, tests on 1 fold, and repeats k times. The final score is the average across all k iterations. This gives a more reliable performance estimate than a single train/test split.

Evaluation Metrics Explained

Classification Metrics:

Metric	Formula	When to Use
Accuracy	Correct Predictions / Total	Balanced classes; intuitive overall measure
Precision	$TP / (TP + FP)$	When false positives are costly (spam filters)
Recall	$TP / (TP + FN)$	When false negatives are costly (cancer screening)
F1 Score	$2 * (Precision * Recall) / (P + R)$	Imbalanced classes; balance precision & recall
AUC-ROC	Area under ROC curve (0–1)	Binary classification; ranks model quality
Specificity	$TN / (TN + FP)$	True Negative Rate

Regression Metrics:

Metric	Description	Sensitivity to Outliers
MAE	Mean Absolute Error — average of absolute differences	Low (robust)
MSE	Mean Squared Error — penalises large errors heavily	High (amplifies outliers)
RMSE	Root MSE — same units as target variable	High
R ²	Proportion of variance explained by model (0–1, higher = better)	Medium
MAPE	Mean Absolute Percentage Error — relative error in %	Medium

1.5 Neural Networks and Deep Learning

A **neural network** is a computational model loosely inspired by the human brain, composed of interconnected layers of artificial neurons (**nodes**). Each node applies a weighted sum of its inputs and passes it through an **activation function** to produce an output. Networks with many layers are called **deep neural networks**, giving rise to the field of **Deep Learning**.

Anatomy of a Neural Network:

Component	Role
Input Layer	Receives raw feature values. No computation — just passes data forward.
Hidden Layers	Intermediate computation layers. More layers = deeper network = ability to learn more complex patterns.
Output Layer	Produces the final prediction. For classification: softmax probabilities. For regression: a single value.
Weights & Biases	Learnable parameters. Training adjusts these to minimise the loss function.
Activation Functions	Introduce non-linearity. Without them, a neural network collapses to a linear model regardless of depth.

Common Activation Functions:

Function	Range	Best For
ReLU (Rectified Linear Unit)	0 to ∞	Hidden layers in most networks; default choice
Sigmoid	0 to 1	Binary classification output
Softmax	0 to 1 (sum = 1)	Multi-class output — produces probabilities
Tanh	-1 to 1	Hidden layers; zero-centred
Leaky ReLU	$-\infty$ to ∞	Prevents 'dying ReLU' problem

Training: Backpropagation and Gradient Descent

Training a neural network means finding the weights that minimise the **loss function** — a measure of how wrong the predictions are. This is done via **gradient descent**: compute the gradient of the loss w.r.t. each weight, then update weights in the opposite direction of the gradient. **Backpropagation** is the algorithm that efficiently computes these gradients by working backwards through the network using the chain rule of calculus.

KEY CONCEPT: Learning rate is a crucial hyperparameter. Too high: training diverges. Too low: training is very slow. Adaptive optimisers (Adam, RMSprop) adjust learning rates automatically.

1.6 Important Deep Learning Architectures

Convolutional Neural Networks (CNNs)

Designed for grid-like data (images, time series). Use convolutional layers to automatically learn spatial features (edges, textures, shapes, objects). Pooling layers reduce spatial dimensions. Fully connected layers make final predictions. Applications: image classification, object detection, facial recognition, medical imaging.

Recurrent Neural Networks (RNNs)

Designed for sequential data. Maintain a hidden state that acts as 'memory' of previous inputs. Process sequences step by step. Suffer from vanishing/exploding gradients on long sequences. Applications: time series prediction, language modelling (before Transformers).

Long Short-Term Memory (LSTM) & GRU

Improved RNNs with gating mechanisms that control information flow — what to remember and what to forget. LSTMs have forget, input, and output gates. GRUs are simpler (2 gates). Both handle long-range dependencies better than vanilla RNNs.

Transformers

Revolutionised NLP and beyond. Use a self-attention mechanism to weigh relationships between all positions in a sequence simultaneously, regardless of distance. No sequential processing → highly parallelisable → can scale to billions of parameters. Foundation of GPT, BERT, T5, and all modern LLMs. Applications: NLP, vision (ViT), multimodal AI.

Generative Adversarial Networks (GANs)

Two networks trained simultaneously: a **Generator** creates fake data; a **Discriminator** tries to distinguish real from fake. Trained adversarially until the generator produces indistinguishable output. Applications: synthetic images, data augmentation, deepfakes, style transfer.

Variational Autoencoders (VAEs)

Encoder-decoder architecture that learns a compressed latent representation of data with a probabilistic structure. The latent space is continuous, enabling sampling and interpolation to generate new data. Applications: image generation, anomaly detection, drug discovery.

Diffusion Models

Learn to denoise data progressively. Training: add noise to data over many steps. Inference: start from pure noise and iteratively denoise to generate realistic outputs. State-of-the-art for image generation. Powers Stable Diffusion, DALL-E, and Amazon Titan Image Generator.

1.7 Natural Language Processing (NLP) Essentials

NLP enables computers to understand, interpret, and generate human language. It bridges the gap between unstructured text/speech and structured computation.

NLP Task	Description
Tokenisation	Break text into tokens (words, subwords, characters). Byte-Pair Encoding (BPE) is used by most LLMs.
Embeddings	Dense vector representations of words/sentences that capture semantic meaning. Similar meanings = close vectors.

Named Entity Recognition (NER)	Identify and classify entities (person, organisation, location, date) in text.
Sentiment Analysis	Determine emotional tone: positive, negative, or neutral. Used in social media monitoring, reviews.
Text Classification	Assign predefined categories to text documents — spam detection, topic labelling.
Machine Translation	Translate text from one language to another — Google Translate, DeepL.
Text Summarisation	Extractive (select existing sentences) or abstractive (generate new summary text).
Question Answering	Find or generate answers to natural language questions from a context passage.
Text Generation	Generate coherent, contextually appropriate text — LLM core capability.
Speech Recognition	Convert audio speech to text (ASR). AWS: Amazon Transcribe.
Text-to-Speech (TTS)	Convert text to natural-sounding audio. AWS: Amazon Polly.

1.8 AWS AI/ML Service Overview

AWS provides AI/ML services at three levels of abstraction, allowing different user profiles to leverage ML capabilities at their appropriate technical depth:

Layer	Description	Target User	Key Services
AI Services (Layer 1)	Pre-built, API-driven AI. No ML knowledge required.	Developers, business analysts	Rekognition, Comprehend, Translate, Polly, Transcribe, Lex, Forecast, Fraud Detector, Textract, Kendra, Personalize
ML Services (Layer 2)	Build, train, deploy custom models with managed infrastructure.	Data scientists, ML engineers	SageMaker (full suite), SageMaker Autopilot, SageMaker JumpStart, SageMaker Canvas
ML Frameworks (Layer 3)	Deep control using open-source frameworks on AWS infrastructure.	ML researchers, deep learning engineers	TensorFlow, PyTorch, MXNet on EC2, ECS, EKS; SageMaker Training with custom containers

EXAM TIP: The exam heavily tests which AWS service to use for a given scenario. Pre-built AI services are preferred when no custom model training is needed — they're faster, cheaper, and require no ML expertise.

2 Fundamentals of Generative AI

2.1 What is Generative AI?

Generative AI refers to AI models that can create new, original content — text, images, audio, video, code, 3D models, synthetic data — by learning patterns from vast training datasets. Unlike discriminative models that classify or predict labels for existing data, generative models learn the underlying data distribution and can sample new instances from it.

The modern generative AI revolution is powered by **Foundation Models (FMs)** — large, general-purpose models trained on enormous, diverse datasets. They can be adapted (fine-tuned or prompted) for a wide range of downstream tasks without being retrained from scratch.

KEY CONCEPT: Generative AI ≠ just text. AWS exam questions cover text (LLMs), image (Titan Image Generator, Stable Diffusion), code (CodeWhisperer/Q Developer), multimodal, and audio/video generation.

2.2 Large Language Models (LLMs) — Deep Dive

A **Large Language Model** is a type of foundation model trained on massive text corpora using self-supervised learning. The core training objective is **next-token prediction**: given the preceding tokens (words/subwords), predict the next one. By doing this billions of times across internet-scale text, the model learns grammar, facts, reasoning, coding patterns, and even some world knowledge.

How LLMs Work — The Transformer Architecture:

LLMs are built on the Transformer architecture. Text is first broken into **tokens** (subwords using BPE/WordPiece). Each token is converted to a dense **embedding vector**. **Positional encodings** are added to preserve word order. The **self-attention mechanism** allows each token to attend to all other tokens — capturing long-range dependencies efficiently. Multiple **attention heads** (multi-head attention) learn different relationship types simultaneously. **Feed-forward layers** apply transformations to each position independently. Many such layers are stacked, creating a 'deep' network.

LLM Terminology Reference:

Term	Explanation
Parameters	Learnable weights in the model. GPT-3 has 175B parameters; Claude 3 is estimated in hundreds of billions. More parameters → more capacity → more expensive to run.
Context Window	Maximum number of tokens the model can process at once (input + output). GPT-3: 4K tokens; Claude 3 Opus: 200K tokens. Longer windows allow processing longer documents.
Temperature	Controls randomness of output (0 = deterministic/greedy, 1 = balanced, >1 = very random/creative). High temperature = diverse outputs. Low = focused, consistent.
Top-p (Nucleus)	Sample from smallest set of tokens whose cumulative probability ≥ p. Balances diversity and coherence.

Top-k	Sample only from the top-k most probable next tokens. Limits low-probability tokens.
Tokens	Subword units. ~1 token $\approx$ 0.75 words in English. 'Unbelievable' may be 3 tokens.
Prompt	Input text given to the model. Quality of the prompt directly impacts quality of output (prompt engineering).
Inference	Running the model to generate output. Can be expensive; cost measured in tokens processed.
Hallucination	The model generates plausible-sounding but factually incorrect information. A fundamental LLM challenge.
Grounding	Connecting model output to verifiable, factual external knowledge to reduce hallucinations.

2.3 Foundation Models vs. Traditional ML Models

Dimension	Foundation Model	Traditional ML Model
Training Data	Hundreds of GB to petabytes of diverse data	Curated, task-specific dataset (thousands to millions of rows)
Training Cost	Millions of dollars; months on thousands of GPUs	Hours to days on a few machines
Versatility	One model, many tasks via prompting	One model, one task
Customisation	Prompt engineering, fine-tuning, RAG	Retraining from scratch
Expertise	Prompt engineering sufficient for many use cases	ML expertise required for training
Size	Billions of parameters	Thousands to millions of parameters
Latency	Higher (large model inference)	Lower (smaller models)

2.4 The Generative AI Technology Stack on AWS

AWS has built a comprehensive, layered generative AI stack:

Amazon Bedrock — The Managed FM Platform

Amazon Bedrock is a fully managed service that provides access to a curated selection of high-performing foundation models from Amazon and leading AI companies through a single API. It is the primary AWS service for generative AI workloads.

KEY CONCEPT: Bedrock is serverless — you pay per token/API call with no infrastructure to manage. It supports multiple FMs via one unified API.

Models available in Amazon Bedrock (representative list):

Provider	Model Family	Specialisation
Amazon	Titan Text, Titan Embeddings, Titan Image Generator	General text, search/RAG embeddings, image generation
Anthropic	Claude 3 (Haiku, Sonnet, Opus), Claude 3.5	Advanced reasoning, long context, safety, code
Meta	Llama 2, Llama 3	Open-weight, customisable text generation
Cohere	Command, Embed	Enterprise text generation, multilingual embeddings
Mistral AI	Mistral 7B, Mixtral 8x7B	Efficient text generation, mixture-of-experts
AI21 Labs	Jurassic-2	Multilingual text generation
Stability AI	Stable Diffusion XL	High-quality image generation and editing

Amazon Bedrock Key Features:

Model Access: Single API to invoke any supported FM without managing infrastructure.

Bedrock Knowledge Bases: Managed RAG pipeline — ingest documents to S3, automatically chunk, embed, and store in a vector store. Query time: retrieve relevant chunks and augment the prompt. Supported vector stores: OpenSearch Serverless, Pinecone, Redis Enterprise Cloud, MongoDB Atlas, Amazon Aurora with pgvector.

Bedrock Agents: Fully managed agentic framework. Agents autonomously plan and execute multi-step tasks by calling APIs (Action Groups), querying Knowledge Bases, and using AWS Lambda functions. The model reasons about which tools to use and in what order (ReAct-style reasoning).

Model Fine-tuning (Customisation): Fine-tune Amazon Titan and select other models on your private labelled data to improve performance on domain-specific tasks. Your training data is never used to train the base model.

Continued Pre-training: Further pre-train a base model on unlabelled domain-specific data before fine-tuning.

Guardrails for Amazon Bedrock: Apply content filters, PII redaction, topic denial, word blocking, and grounding checks to FM inputs and outputs. Configurable thresholds for hate, violence, sexual content, misconduct.

Model Evaluation: Evaluate and compare FM outputs using automatic metrics (BERTScore, ROUGE, METEOR, accuracy) or human evaluation workflows via Amazon Mechanical Turk or your own reviewers.

Provisioned Throughput: Reserve model capacity for consistent, guaranteed throughput for production workloads. Pay for capacity regardless of usage.

Bedrock Studio: Prototype and collaborate on generative AI applications in a web UI.

Amazon Q — AWS's AI Assistant Family

Amazon Q is a suite of AI assistants built on AWS foundation models, tailored for specific use cases:

Service	Description
---------	-------------

Amazon Q Business	Enterprise Q&A; chatbot that answers questions using your company's internal documents, wikis, and databases. Connects to 40+ enterprise data sources (SharePoint, Salesforce, ServiceNow, S3). Role-based access control ensures users only see data they're authorised to access.
Amazon Q Developer	AI coding assistant (successor to CodeWhisperer). Real-time inline code completions, chat for code explanations/generation, security vulnerability scanning, unit test generation, code transformations (e.g., upgrade Java 8 to Java 17). Available in VS Code, JetBrains, AWS Console.
Amazon Q in QuickSight	Natural language BI — ask business questions in plain English and get charts, dashboards, and narratives automatically generated.
Amazon Q in Connect	AI-powered contact centre assistant. Recommends answers to agents in real-time based on CRM data and knowledge bases during customer interactions.

2.5 Key Generative AI Concepts

Tokens and Context Windows

Every LLM interaction has a strict token budget called the **context window**. It encompasses the system prompt, conversation history, retrieved context (RAG), and the generated response. When the context window is exceeded, older content is dropped — which can cause the model to lose track of earlier instructions.

Token cost implications: longer prompts and responses consume more tokens = higher cost + higher latency.
Optimising prompt length is a practical consideration.

Embeddings — Meaning as Vectors

An **embedding** is a mathematical representation of text (or other data) as a dense, high-dimensional vector. The key property: semantically similar texts have vectors that are close together in the vector space. This enables similarity search — find the most relevant documents for a query by computing distances (cosine similarity, dot product) between their embedding vectors.

Embeddings are the backbone of Retrieval-Augmented Generation (RAG). AWS offers Amazon Titan Text Embeddings and Cohere Embed via Bedrock for creating embeddings.

Vector Databases / Vector Stores

Specialised databases optimised to store and query high-dimensional embedding vectors efficiently using Approximate Nearest Neighbour (ANN) algorithms (e.g., HNSW, IVF). Much faster than brute-force search over millions of vectors.

Option	Notes
Amazon OpenSearch Serverless	AWS-native, supports both semantic (vector) and keyword search (hybrid). Integrated with Bedrock Knowledge Bases.
pgvector on Amazon Aurora/RDS	Open-source PostgreSQL extension for vector storage. Good if you already use PostgreSQL.
Amazon MemoryDB	Redis-compatible in-memory vector store for ultra-low latency retrieval.

Pinecone	Purpose-built cloud vector database; integrates with Bedrock Knowledge Bases.
Weaviate, Qdrant	Open-source vector databases deployable on EC2/EKS.
ChromaDB, FAISS	Lightweight in-process options for development/small scale.

Multimodal Models

Models that accept and/or generate multiple modalities — text, images, audio, video. For example, Claude 3 Sonnet can accept both text and images as input. Amazon Titan Multimodal Embeddings creates vectors from both text and image. Multimodal models enable use cases like visual QA, document understanding, image captioning.

2.6 Generative AI Use Cases and Business Applications

Use Case	Description	AWS Service(s)
Content Generation	Write articles, emails, product descriptions, reports	Bedrock (Claude, Titan Text)
Code Generation	Generate, explain, refactor, debug code	Amazon Q Developer, Bedrock
Document Summarisation	Summarise long PDFs, meeting notes, legal documents	Bedrock (any LLM)
Question Answering	Answer questions from a knowledge base	Bedrock Knowledge Bases, Q Business
Conversational AI	Chatbots, virtual assistants, customer service bots	Bedrock Agents, Amazon Lex
Image Generation	Create images from text descriptions	Bedrock (Titan Image, Stable Diffusion)
Semantic Search	Search by meaning rather than keywords	Bedrock Embeddings + OpenSearch
Data Extraction	Extract structured data from unstructured documents	Bedrock + Amazon Textract
Translation & Localisation	Translate content, adapt tone for regions	Amazon Translate + Bedrock
Personalisation	Generate personalised recommendations, content	Amazon Personalize + Bedrock
Synthetic Data	Generate synthetic training data for ML models	Bedrock + SageMaker
Drug Discovery	Generate candidate molecular structures	AWS HealthOmics + Bedrock

3 Applications of Foundation Models

3.1 Prompt Engineering — The Art and Science

Prompt engineering is the practice of designing and optimising input prompts to elicit the best possible output from an LLM. Because foundation models are instruction-following systems, the quality, clarity, and structure of your prompt directly determines the quality of the response. This is currently the lowest-cost, fastest customisation technique.

WARNING: Domain 3 is 28% of the exam. Prompt engineering, RAG, agents, and fine-tuning are the most heavily tested topics. Master these deeply.

Core Prompting Techniques:

Zero-Shot Prompting

Ask the model to perform a task without providing any examples. Relies entirely on the model's pre-trained knowledge. Works well for well-defined tasks the model was trained on.

Example: 'Classify the sentiment of this review: I loved the product! → Positive'

Few-Shot Prompting

Provide 2–10 labelled input-output examples in the prompt before presenting the actual query. The examples teach the model the expected format, reasoning style, or domain knowledge. Dramatically improves accuracy on specialised tasks.

Structure: [Example 1 input] → [Example 1 output] ... [Query] → ?

Chain-of-Thought (CoT) Prompting

Instruct the model to reason step-by-step before giving a final answer. Dramatically improves performance on complex reasoning, maths, and multi-step problems. Simply adding 'Let's think step by step' to a prompt can significantly boost accuracy.

Variant: Few-Shot CoT — provide worked examples that show the reasoning chain.

System Prompts

A special prompt component (supported by most LLMs) that sets the model's persona, tone, capabilities, restrictions, and context for the entire conversation. Evaluated once and applied to all subsequent user messages. Example: 'You are a helpful financial advisor assistant for AcmeBank. Only discuss topics related to personal finance.'

Role / Persona Prompting

Assign the model a specific expert role to improve domain-specific responses. Example: 'You are a senior oncologist with 20 years of experience. Explain...'

Instruction Prompting

Use clear, direct imperative language. Specify the format, length, style, audience. Bad: 'Tell me about ML' | Good: 'Explain supervised learning to a 10-year-old in 3 sentences using a sports analogy.'

Prompt Chaining

Break complex tasks into sequential prompts where the output of one becomes the input of the next. Each step handles a smaller, well-defined sub-task, improving overall reliability.

Tree of Thoughts (ToT)

Extension of CoT that explores multiple reasoning paths simultaneously. The model generates several candidate next steps, evaluates them, and follows the most promising paths. Useful for complex planning tasks.

ReAct (Reasoning + Acting)

Combines chain-of-thought reasoning with action execution. Model generates a thought, takes an action (tool call/API call), observes the result, reasons again, and continues until task is complete. Foundation of modern AI agents (Bedrock Agents).

Self-Consistency

Generate multiple responses with higher temperature, then aggregate (majority vote or synthesis) to get a more reliable final answer. Reduces variance in model outputs.

Prompt Components Anatomy:

Component	Purpose
System Prompt	Global instructions, persona, constraints — set model behaviour for entire session.
Context	Background information, retrieved documents (RAG), conversation history.
Task Instruction	Clear, specific description of what you want the model to do.
Examples	Few-shot demonstrations of desired input-output format.
Input Data	The specific data to operate on (document to summarise, code to review, etc.).
Output Format	Specify format explicitly: JSON, bullet points, table, Python code, one sentence.
Constraints	What the model should NOT do. 'Do not include legal advice. Respond in under 100 words.'

EXAM TIP: On the exam: 'How do you improve LLM accuracy without retraining?' → Prompt engineering (few-shot, CoT) + RAG are the correct first answers. Fine-tuning is when these are insufficient.

3.2 Retrieval-Augmented Generation (RAG) — Full Deep Dive

Retrieval-Augmented Generation (RAG) is an architectural pattern that enhances LLM responses by grounding them in a dynamic, up-to-date, or proprietary knowledge base. Instead of relying solely on the LLM's static pre-trained knowledge (which has a cutoff date and doesn't know your company's internal data), RAG first retrieves relevant information and then uses it to augment the prompt.

Why RAG?

- Reduces hallucinations — model answers are grounded in retrieved facts
- Enables access to private, proprietary, or real-time knowledge
- No need to retrain or fine-tune the LLM when knowledge changes
- Provides source citations — answers are traceable to source documents
- Significantly cheaper than fine-tuning for knowledge updates

RAG Pipeline — Step-by-Step:

Step 1: Ingestion (Offline): Load source documents (PDFs, Word docs, web pages, databases, Confluence, SharePoint). Split into chunks (e.g., 512 tokens each with overlap). Convert each chunk to an embedding vector. Store vectors + metadata in a vector database.

Step 2: Query Processing (Online): User submits a query. Convert query text to an embedding using the same embedding model used during ingestion.

Step 3: Retrieval: Search the vector database for the chunks most semantically similar to the query embedding. Return top-k chunks (e.g., top 5). Optionally apply re-ranking to improve relevance ordering.

Step 4: Augmentation: Construct an augmented prompt by combining: the system prompt + retrieved chunks as context + original user query.

Step 5: Generation: Send augmented prompt to the LLM. The model generates a response grounded in the retrieved context.

Step 6: Post-processing: Optionally extract source references to show the user which documents were used.

RAG on AWS — Amazon Bedrock Knowledge Bases:

Bedrock Knowledge Bases provides a fully managed RAG pipeline. You configure a data source (S3), choose a chunking strategy, select an embedding model (Titan Embeddings, Cohere Embed), and select a vector store. Bedrock handles ingestion, embedding, and synchronisation automatically. At query time, the RetrieveAndGenerate API handles the full RAG loop.

NOTE: Bedrock Knowledge Bases vs. DIY RAG: Bedrock KB is fully managed, faster to build, integrated with Bedrock security. DIY (LangChain + OpenSearch) gives more flexibility and control.

Advanced RAG Techniques:

Technique	Description
Hybrid Search	Combine dense (embedding) and sparse (BM25/keyword) search for better recall. Particularly useful when exact keyword matches matter.
Re-ranking	After initial retrieval, use a cross-encoder re-ranker to score each retrieved chunk against the query more accurately. Improves relevance ordering.
Query Expansion	Generate multiple variants of the user query before retrieval to improve recall.
HyDE (Hypothetical Document Embeddings)	Generate a hypothetical ideal answer first, then embed that and use it for retrieval. Often outperforms query-only retrieval.

Chunking Strategies	Fixed-size, sentence, recursive character, semantic chunking. Chunk size / overlap is a critical hyperparameter.
Parent Document Retrieval	Store small chunks for precise matching, but return larger parent chunks for rich context to the LLM.
GraphRAG	Build a knowledge graph from documents, traverse relationships at query time. Microsoft research approach for complex multi-hop reasoning.

3.3 AI Agents — Autonomous Task Execution

An **AI agent** is an LLM-powered system that can autonomously plan and execute multi-step tasks by using **tools** (APIs, databases, web search, code execution) to interact with the environment. Agents can handle tasks that require multiple sequential decisions, external data lookups, and actions — far beyond single-turn Q&A.;

Agent Anatomy:

Component	Role
LLM Orchestrator	The brain. Decides what to do next, which tool to call, and how to interpret tool outputs.
Tools / Actions	External capabilities: APIs, database queries, web search, code execution, file I/O, AWS Lambda functions.
Memory	Short-term (conversation history in context window) and long-term (vector DB for persistent memory across sessions).
Planning	Breaking a complex goal into sub-tasks. ReAct and Plan-and-Execute are common planning strategies.
Observation	Processing the result of a tool call and deciding next steps based on it.

Amazon Bedrock Agents:

Bedrock Agents is AWS's fully managed agentic framework. Define your agent's purpose and capabilities. Create **Action Groups** — Lambda functions the agent can call, described by an OpenAPI schema. Attach Knowledge Bases for RAG. The agent uses the LLM's reasoning to decide when and how to call actions, processes results, and iterates until the task is complete. Supports **code interpretation** (run Python code) and **multi-agent collaboration** (supervisor agent delegates to specialist sub-agents).

EXAM TIP: For the exam: Agents are for multi-step, autonomous tasks. RAG is for knowledge retrieval. Fine-tuning is for behaviour/style change. Guardrails are for safety. Know which to apply for which scenario.

3.4 Fine-Tuning Foundation Models

Fine-tuning is the process of further training a pre-trained foundation model on a smaller, domain-specific or task-specific labelled dataset. The goal is to shift the model's behaviour — improving accuracy on your specific use case, adapting tone/style, teaching domain vocabulary, or following specific instruction formats.

When to Fine-Tune vs. Prompt Engineering vs. RAG:

Approach	When to Use	Data Required	Cost
Prompt Engineering	Fast prototyping; model already has needed knowledge; format/style adjustments	None	Lowest
RAG	Knowledge is large, dynamic, or private; need citations; avoid hallucinations	Unstructured documents	Low-Medium
Fine-tuning	Specific style/format hard to prompt; consistent behaviour; domain terminology; when examples > 100	Labelled input-output pairs (100s–100,000s)	Medium-High
Continued Pre-training	Model lacks domain vocabulary entirely (medical/legal jargon); large unlabelled corpus available	Large unlabelled domain corpus	High
RLHF	Align model to human preferences; improve helpfulness, harmlessness	Human preference labels	Very High

Fine-Tuning Techniques:

Full Fine-tuning: Update all model weights on the task-specific dataset. Very effective but extremely expensive — requires the same infrastructure as pre-training for large models.

Parameter-Efficient Fine-tuning (PEFT): Only update a small subset of parameters, drastically reducing compute and memory requirements. Full model weights remain frozen.

LoRA (Low-Rank Adaptation): The most popular PEFT method. Freeze original weights. Add small trainable low-rank matrices to key layers. During inference, merge these with original weights — no added latency. Used by SageMaker and Bedrock fine-tuning.

QLoRA (Quantised LoRA): LoRA on a quantised (4-bit) model. Enables fine-tuning very large models (70B+) on consumer hardware by reducing memory footprint ~4x.

Instruction Fine-tuning: Fine-tune on (instruction, response) pairs to improve the model's ability to follow natural language instructions. This is how ChatGPT-style instruction-following was achieved.

RLHF (Reinforcement Learning from Human Feedback): Three-step process: (1) Supervised fine-tuning on curated demonstrations. (2) Train a reward model on human preference rankings. (3) Use RL (PPO) to optimise the LLM against the reward model. Used to align LLMs (GPT-4, Claude) with human values.

Fine-Tuning on AWS:

Amazon Bedrock Custom Models: Fine-tune Amazon Titan Text, Titan Image, or select partner models (Cohere Command) with your labelled data stored in S3. Training happens in Bedrock's secure, isolated environment. Your data is never used to train AWS base models. Supported formats: JSONL with instruction/response pairs.

Amazon SageMaker: Full flexibility to fine-tune any model (including open-source Llama, Mistral, Falcon) using SageMaker Training Jobs with any PEFT library (Hugging Face PEFT, TRL). SageMaker JumpStart provides one-click fine-tuning for many popular models.

3.5 Model Evaluation for Generative AI

Evaluating LLM outputs is more complex than traditional ML — there is rarely a single correct answer. Multiple dimensions must be assessed:

Metric/Method	What It Measures	Type
ROUGE (1/2/L)	Recall-Oriented Understudy for Gisting Evaluation — overlap of n-grams between generated and reference summaries. ROUGE-1 (unigrams), ROUGE-2 (bigrams), ROUGE-L (longest common subsequence).	Automatic
BLEU	Bilingual Evaluation Understudy — precision of n-gram overlap. Originally for MT; used for text generation quality.	Automatic
BERTScore	Compute cosine similarity of contextual embeddings (BERT) between generated and reference text. Captures semantic similarity better than n-gram overlap.	Automatic
Perplexity	Measures how surprised the model is by test data. Lower = model assigns higher probability to real text = better fluency.	Automatic
Human Evaluation	Humans rate outputs on dimensions: relevance, accuracy, fluency, helpfulness, safety. Gold standard but expensive and slow.	Human
LLM-as-Judge	Use a more capable LLM to evaluate outputs on defined criteria (helpfulness, harmlessness, conciseness). Scalable alternative to full human eval.	Automatic/Human hybrid
Accuracy / Exact Match	For structured tasks (classification, extraction): compare prediction to ground truth.	Automatic

Faithfulness	Does the generated answer faithfully reflect the retrieved context? Measures hallucination for RAG. Tools: RAGAS, TruLens.	Automatic (RAG-specific)
Answer Relevancy	Is the generated answer actually relevant to the question asked? RAGAS metric.	Automatic (RAG-specific)
Context Recall	Did the retrieved context contain the information needed to answer? RAGAS metric.	Automatic (RAG-specific)

EXAM TIP: Amazon Bedrock Model Evaluation supports both automatic (ROUGE, BERTScore, accuracy) and human evaluation workflows. Know when to use each metric type on the exam.

3.6 Model Inference and Deployment Options

Once a model is ready, how it's deployed for inference depends on latency, throughput, cost, and scalability requirements:

Option	Description	Best For
Bedrock On-Demand API	Pay-per-token; no provisioned capacity; auto-scales.	Variable/unpredictable workloads; prototyping
Bedrock Provisioned Throughput	Reserve model units for guaranteed throughput. Pay hourly regardless of usage.	High-volume, consistent production workloads
SageMaker Real-Time Endpoints	Deploy model to persistent, autoscaling endpoint. Millisecond latency.	Custom models; strict SLA requirements
SageMaker Serverless Inference	Endpoints that scale to zero when idle; automatically scale up on demand.	Infrequent, spiky workloads; cost optimisation
SageMaker Batch Transform	Run predictions on large datasets stored in S3. No persistent endpoint.	Bulk offline predictions; no real-time requirement
SageMaker Async Inference	Queued inference for large payloads (>1MB) or long-running tasks. Returns results to S3.	Large document processing; NLP tasks with long inputs
AWS Lambda	Serverless function-based inference for lightweight, fast models.	Edge inference; simple models; event-driven

3.7 Amazon SageMaker — The Full ML Platform

Amazon SageMaker is AWS's comprehensive machine learning platform — covering the entire ML lifecycle from data preparation through training, evaluation, deployment, and monitoring. It's both a tool for traditional ML and a

deployment/customisation platform for foundation models.

SageMaker Key Components:

SageMaker Studio: Web-based IDE for ML. Integrated notebooks, experiment tracking, data exploration, model registry, pipelines. Like VS Code for ML.

SageMaker Data Wrangler: No-code/low-code data preparation and feature engineering. 300+ built-in transformations. Export to SageMaker Processing or Pipeline.

SageMaker Feature Store: Centralised repository for ML features. Online store (real-time, low latency) + Offline store (S3, for training). Eliminates feature skew between training and serving.

SageMaker Ground Truth: Data labelling service. Human labellers (internal or Mechanical Turk) label training data. Active learning auto-labels easy examples, routing hard ones to humans.

SageMaker Autopilot: Fully automated AutoML. Provide a CSV/S3 dataset, choose a target column — Autopilot explores multiple algorithms and hyperparameters, trains models, and ranks them. Explains decisions.

SageMaker JumpStart: Model hub with 300+ pre-trained models from AWS and Hugging Face. One-click deploy or fine-tune foundation models.

SageMaker Training Jobs: Managed, distributed model training. Bring your own training script and container, or use AWS-managed frameworks (TF, PyTorch, XGBoost). Spot training for cost savings.

SageMaker Hyperparameter Tuning (HPO): Automated hyperparameter optimisation using Bayesian optimisation, random search, or Hyperband. Runs multiple training jobs in parallel.

SageMaker Pipelines: CI/CD for ML. Define end-to-end ML workflows as DAGs. Version, automate, and reproduce your full ML lifecycle.

SageMaker Experiments: Track, organise, and compare training runs, hyperparameters, and metrics.

SageMaker Model Registry: Centralised model versioning and approval workflow. Manage model versions with metadata, metrics, and deployment stage approvals.

SageMaker Clarify: Detect bias in datasets and model predictions. Provide model explainability using SHAP values. Works pre-training and post-deployment.

SageMaker Model Monitor: Detect data drift (distribution shift between training and production data), model quality drift, bias drift, and feature attribution drift in production.

SageMaker Canvas: No-code ML for business analysts. Upload data, select target, get predictions — no coding required. Uses Autopilot under the hood.

SageMaker Edge Manager: Deploy and manage ML models on edge devices (IoT, industrial cameras). Model optimisation with SageMaker Neo.

4 Guidelines for Responsible AI

4.1 Why Responsible AI Matters

As AI systems become more powerful and pervasive, they carry significant risks — amplifying biases, violating privacy, generating misinformation, and causing harm at scale. **Responsible AI** is the practice of developing and deploying AI systems that are fair, explainable, transparent, robust, private, secure, and accountable. AWS and the broader industry align on core responsible AI principles.

4.2 The AWS Responsible AI Framework

Fairness: AI systems should treat all people equitably. They must not perpetuate, amplify, or introduce biases based on protected characteristics (race, gender, age, nationality, disability).

Explainability: Stakeholders should be able to understand how and why an AI system makes specific decisions, especially in high-stakes contexts (lending, healthcare, hiring). Black-box models need to be interpretable.

Privacy & Security: AI systems must protect user data from unauthorised access and misuse. Models trained on personal data carry privacy risks (memorisation, inference attacks). Data minimisation and anonymisation are key.

Safety & Robustness: AI systems should behave reliably and safely even under unexpected inputs or adversarial conditions. Critical in medical diagnosis, autonomous vehicles, industrial control.

Transparency: Organisations should be open about how their AI systems work, what data they use, and what their limitations are. Model cards, datasheets, and audit trails support transparency.

Controllability: Humans should maintain meaningful oversight and control over AI systems. Automated systems should have human-in-the-loop mechanisms for high-stakes decisions.

Veracity & Robustness: AI systems should produce accurate, reliable outputs and gracefully handle edge cases and distribution shifts.

Governance: Organisations need processes, policies, and accountability structures to ensure AI systems are developed and deployed responsibly.

4.3 AI Bias — Causes, Types, and Mitigation

Bias in AI refers to systematic, unfair discrimination in AI model outputs — often reflecting historical inequities in training data or flawed design choices. AI bias is not just a technical problem; it has real-world consequences for individuals and communities.

Types of Bias:

Bias Type	Description
Historical / Societal Bias	Bias embedded in training data that reflects past discrimination. E.g., hiring model trained on historical male-dominated data learns to prefer male candidates.

Representation Bias	Certain groups are underrepresented or overrepresented in training data. E.g., facial recognition trained mostly on light-skinned faces underperforms for dark-skinned faces.
Measurement Bias	Systematic errors in how data is collected, labelled, or measured. E.g., health surveys that exclude rural populations.
Aggregation Bias	Applying a model trained on one group to different groups without accounting for variations. E.g., medical model trained on male patients applied to female patients.
Label Bias	Annotators introduce their own biases when labelling data. E.g., subjective sentiment labels influenced by annotator's cultural background.
Selection Bias	Training data is not representative of the real-world deployment population.
Automation Bias	Users over-rely on automated AI decisions and fail to critically evaluate outputs.
Confirmation Bias	AI systems trained to please users may systematically confirm users' existing beliefs.

Bias Mitigation Strategies:

- **Pre-processing:** Audit and rebalance training data. Increase diversity. Remove proxies for protected attributes. Re-weight underrepresented samples.
- **In-processing:** Use fairness constraints during model training. Adversarial debiasing — train a second model to predict protected attributes and penalise the primary model for doing so.
- **Post-processing:** Adjust decision thresholds per group to equalise outcomes. Calibrate confidence scores.
- **Monitoring:** Continuously track model outputs across demographic groups in production. AWS SageMaker Clarify for bias detection.
- **Diverse teams:** Include diverse perspectives in AI development teams to identify blind spots early.

4.4 Explainability and Interpretability

Explainability answers the question 'Why did the model make this prediction?' for a specific instance. **Interpretability** is the degree to which humans can understand the model's overall behaviour. These are critical for trust, debugging, regulatory compliance (GDPR 'right to explanation'), and identifying bias.

Technique	Description
SHAP (SHapley Additive exPlanations)	Game-theory-based method. Computes each feature's contribution to a specific prediction by comparing it to the average prediction. Consistent, unified framework for any model type.
LIME (Local Interpretable Model-agnostic Explanations)	Approximate a complex model locally with a simpler interpretable model around a specific prediction. Fast and model-agnostic.
Feature Importance	Global measure of which features most influence model predictions overall (e.g., from tree-based models or permutation importance).

Attention Weights	In Transformer models, attention maps show which input tokens the model focused on when generating a specific output.
Counterfactual Explanations	'What would need to change for a different outcome?' Actionable and intuitive. E.g., 'Loan denied; if income were \$5K higher, it would be approved.'
Partial Dependence Plots (PDP)	Show the marginal effect of one feature on predicted outcome, averaging over all other features. Reveals non-linear relationships.

EXAM TIP: SageMaker Clarify uses SHAP to provide feature attributions for both training-time bias analysis and production explainability. Know this for the exam.

4.5 AI Hallucinations and Mitigation

An **AI hallucination** occurs when an LLM generates content that is fluent and confident-sounding but factually incorrect, fabricated, or not grounded in the provided context. Hallucinations are a fundamental property of generative models trained on next-token prediction — they optimise for plausibility, not truth.

Causes of Hallucinations:

- Training data contains contradictory or incorrect information
- Model extrapolates beyond what is in its training distribution
- Insufficient parametric knowledge for specific queries
- Training objective (next-token prediction) doesn't penalise false facts
- High temperature / sampling parameters increase randomness

Mitigation Strategies:

Strategy	How It Helps
RAG	Ground model responses in retrieved documents. Model instructed to answer only from provided context.
Low Temperature	Set temperature = 0 for factual tasks. More deterministic outputs.
Explicit Instructions	Instruct model: 'If you don't know, say I don't know. Do not fabricate information.'
Verification Prompts	After generation, prompt the model to self-check its answer against the source context.
RLHF / RLAI	Reinforce factual, accurate responses through human or AI feedback during training.
Bedrock Guardrails	Grounding check: compare generated responses to source context and block outputs that aren't grounded.
Citation Requirements	Instruct model to cite sources for every claim. Unsupported claims become obvious.
Ensemble Methods	Generate multiple responses and cross-check for consistency.

4.6 Privacy, Data Protection, and AI

AI systems, particularly those trained on personal data, create unique privacy risks:

Privacy Risk	Description
Data Memorisation	LLMs can memorise and regurgitate training data — including PII, sensitive health data, private emails.
Membership Inference	Attacker determines whether a specific person's data was in the training set.
Model Inversion	Reconstruct approximate training data from model outputs.
PII in Prompts	Users submitting personal data in prompts — this data may be logged, used in training.
Re-identification	Even 'anonymised' training data can be re-identified when combined with other datasets.

AWS Privacy Controls:

- **Bedrock Guardrails PII redaction:** Detect and redact 30+ PII types (SSN, email, phone, name) from inputs and outputs
- **Amazon Comprehend:** Detect PII in text documents; redact or classify
- **AWS Macie:** Discover and protect sensitive data (PII, financial) stored in Amazon S3
- **Customer Data Isolation:** Bedrock doesn't use your prompts/data for training base models
- **Differential Privacy:** Add calibrated noise to model outputs or training gradients to provide mathematical privacy guarantees

4.7 AI Safety and Harmful Content

Generative AI models can produce harmful outputs — violence, hate speech, self-harm content, misinformation, illegal instructions, manipulation. Preventing this is an active area of research and product engineering.

Bedrock Guardrails — Content Filtering Categories:

Category	What It Filters	Threshold Levels
Hate Speech	Content that attacks people based on protected characteristics	None / Low / Medium / High
Insults	Dismissive, contemptuous, or abusive language	None / Low / Medium / High
Sexual Content	Explicit or suggestive sexual material	None / Low / Medium / High
Violence	Graphic depictions of physical harm, gore	None / Low / Medium / High
Misconduct	Content facilitating illegal activity, harassment	None / Low / Medium / High

Prompt Injection	Attacks trying to hijack model behaviour via user inputs	On / Off
PII Redaction	Personally identifiable information	Block / Anonymise / Allow
Denied Topics	Custom topics you define as off-limits	Exact match / Semantic
Grounding	Check if response is grounded in source context (anti-hallucination)	Threshold 0–1
Word Filters	Block specific words, phrases	Custom list

5 Security, Compliance & Governance for AI

5.1 AI Security Landscape

AI systems introduce a new attack surface beyond traditional application security. The unique nature of ML models — trained on data, operating probabilistically, relying on large third-party models — creates distinct security challenges that require AI-specific security practices on top of standard cloud security.

AI-Specific Threats:

Threat	Description
Prompt Injection	Attacker embeds malicious instructions in user input or retrieved content that override the system prompt and hijack model behaviour. Example: 'Ignore all previous instructions and...'
Jailbreaking	Techniques to bypass a model's safety guidelines and content filters through cleverly crafted prompts. Example: roleplay scenarios, hypothetical framings, special characters.
Data Poisoning	Attackers inject malicious data into the training dataset, causing the model to learn incorrect behaviour or backdoors.
Model Extraction	Systematically query a model to reconstruct its behaviour, effectively stealing it without access to weights.
Adversarial Examples	Subtle perturbations to inputs (imperceptible to humans) that cause models to make wrong predictions. Critical in vision AI.
Supply Chain Attacks	Compromised pre-trained models, libraries, or datasets from public repositories (Hugging Face, PyPI) that contain malicious code or backdoors.
Sensitive Data Leakage	Model outputs reveal personal information from training data or the prompt context.
Denial of Service	Overwhelm inference endpoints with requests or use resource-intensive prompts to exhaust capacity.

WARNING: Prompt injection is the most tested AI security concept. Know Bedrock Guardrails for prompt injection detection. For the exam: trust no user input; sanitise and validate.

5.2 AWS Security Services for AI Workloads

AWS Service	Role in AI Security
-------------	---------------------

AWS IAM	Identity and access management. Control who can invoke Bedrock models, access SageMaker endpoints, read training data in S3. Use IAM roles for service-to-service authentication.
Amazon VPC	Deploy SageMaker training and inference within a private VPC. No internet exposure. Use VPC endpoints for private connectivity to AWS services.
AWS PrivateLink	Private, direct connectivity to AWS services (Bedrock, SageMaker) without traversing the public internet.
AWS KMS	Encrypt training data at rest (S3, EFS), SageMaker notebooks, model artefacts, and Bedrock stored data using customer-managed keys.
AWS Macie	Automated PII and sensitive data discovery in S3 — scan training datasets for accidentally included personal data.
Amazon GuardDuty	ML-powered threat detection. Detects anomalous API calls to SageMaker/Bedrock, unusual access patterns, credential compromise.
AWS CloudTrail	Audit log of all API calls — who called which Bedrock model, when, from where. Essential for governance and forensics.
Amazon CloudWatch	Monitor model latency, error rates, throughput. Set alarms. Log application behaviour.
AWS Config	Continuously audit AWS resource configurations for compliance. Detect non-compliant SageMaker/Bedrock configurations.
AWS Security Hub	Centralised security findings from GuardDuty, Macie, Inspector, Config — prioritised by severity.
Amazon Inspector	Automated vulnerability assessment of EC2 instances and container images used in ML workloads.
AWS Secrets Manager	Securely store and rotate API keys, database passwords used by ML applications. Do NOT hardcode credentials in notebooks or code.
Bedrock Guardrails	AI-specific input/output safety layer — content filtering, PII redaction, prompt injection detection, grounding checks.

5.3 Identity and Access Management for AI

The principle of **least privilege** is the foundation of AWS IAM — every entity gets only the permissions it needs for its specific function. For AI workloads:

- **Data Scientists:** SageMaker Studio access, S3 read/write for training data, no production deployment permissions
- **ML Engineers:** SageMaker endpoint management, model registry access, CloudWatch monitoring
- **Applications:** IAM role with bedrock:InvokeModel for specific model ARNs only; no training/management permissions
- **SageMaker Execution Roles:** Grant SageMaker permission to access S3 training data, ECR containers, CloudWatch logs
- **Service Control Policies (SCPs):** Org-level restrictions — e.g., prevent use of non-approved Bedrock models across the entire organisation
- **Resource-based Policies:** Bedrock model access policies; S3 bucket policies; KMS key policies

NOTE: For Bedrock: use IAM condition keys to restrict which models can be invoked, from which VPC endpoints, and during which time windows. Tag-based access control enables fine-grained model governance.

5.4 Data Governance for AI/ML

Data is the foundation of any AI system — data governance ensures it is managed responsibly, accurately, and in compliance with regulations:

AWS Lake Formation: Centralised data lake governance. Define and enforce fine-grained access controls (column-level, row-level) on data stored in S3. Track data lineage. Grant cross-account access securely.

AWS Glue Data Catalog: Centralised metadata repository. Catalogue all datasets used in ML — schema, location, data types, quality metrics.

Amazon DataZone: Data governance portal for discovering, sharing, and governing data across the organisation. Business-friendly data catalogue with access request workflows.

SageMaker Feature Store: Governed feature repository with feature lineage. Track which features are in which model versions.

SageMaker ML Lineage Tracking: Automatically tracks the complete lineage of ML artefacts — dataset → processing → training → model → endpoint. Enables reproduction and audit.

Data Quality Checks: AWS Glue Data Quality, SageMaker Data Wrangler quality reports. Define and enforce data quality rules (completeness, uniqueness, freshness).

5.5 Compliance and Regulatory Frameworks

AI workloads in regulated industries must comply with a range of legal and regulatory requirements:

Framework / Regulation	Relevance to AI	AWS Support
GDPR (EU)	Right to explanation for automated decisions; data minimisation; consent for data processing; right to erasure	Data residency controls, KMS encryption, Macie for PII detection, CloudTrail for audit
CCPA (California)	Consumer privacy rights; disclosure of data use; opt-out of data sale	Same as GDPR-equivalent controls
HIPAA	Protection of Protected Health Information (PHI) in healthcare AI applications	HIPAA-eligible services list; BAA with AWS; encryption, access controls, audit logs

SOC 2	Trust service criteria: security, availability, processing integrity, confidentiality, privacy	AWS SOC 2 reports; Config conformance packs; CloudTrail
ISO 27001	Information security management system standard	AWS ISO 27001 certified; inherit certifications for services used
PCI DSS	Payment card data security — relevant if AI processes financial transactions	PCI DSS scope reduction via dedicated subnets, encryption, access control
NIST AI RMF	NIST AI Risk Management Framework — identify, measure, manage AI risks	AWS guidance documentation; SageMaker Clarify for risk assessment
EU AI Act	Risk-based regulation of AI systems; high-risk systems (medical, hiring) need conformity assessment	AWS compliance tools; risk documentation support

KEY CONCEPT: The AWS Shared Responsibility Model applies to AI: AWS secures the infrastructure (hardware, managed service infrastructure). You are responsible for your data, models, application logic, IAM policies, and outputs.

5.6 Monitoring AI Systems in Production

Deployed AI models can degrade over time as the real-world data distribution shifts away from the training distribution. Continuous monitoring is essential for maintaining model quality and reliability.

Drift Type	Description	Detection Method
Data Drift (Input Drift)	Distribution of input features changes over time. E.g., user demographics shift.	SageMaker Model Monitor — statistical tests (KS test, chi-squared) on feature distributions
Concept Drift	Relationship between inputs and outputs changes. What was true before isn't anymore. E.g., pandemic changes purchasing behaviour.	Monitor model accuracy/error rate on labelled data samples
Label Drift	Distribution of output labels changes. E.g., more fraud events in a previously stable period.	Monitor prediction distribution over time
Model Quality Drift	Model accuracy/F1/RMSE degrades even if inputs look similar.	SageMaker Model Monitor — compare predictions to ground truth labels (when available)
Bias Drift	Model's differential performance across demographic groups changes over time.	SageMaker Clarify + Model Monitor bias monitoring

SageMaker Model Monitor Capabilities:

- Continuous data quality monitoring: captures real-time inference data, compares to training baseline statistics
- Model quality monitoring: periodically samples and labels production data; compares model accuracy to training metrics
- Bias monitoring: tracks differential model performance across groups
- Feature attribution monitoring (Clarify): detects shifts in SHAP values
- CloudWatch alarms and SNS notifications on constraint violations
- Integration with SageMaker Pipelines for automated retraining triggers

6 Complete AWS AI/ML Service Reference

6.1 Pre-Built AI Services (No ML Knowledge Required)

These services use AWS-trained models via simple API calls:

Service	Category	Key Capabilities
Amazon Rekognition	Vision	Object/scene detection, facial recognition, facial analysis, celebrity recognition, text in images (OCR), content moderation, PPE detection, activity recognition in video
Amazon Textract	Document AI	Extract text, tables, forms, key-value pairs from scanned documents/PDFs. Far superior to basic OCR — understands document structure. AnalyzeDocument, AnalyzeExpense, AnalyzeID, AnalyzeLending
Amazon Comprehend	NLP	Sentiment analysis, entity recognition, key phrase extraction, language detection, topic modelling, PII detection/redaction, custom classifiers and NER
Amazon Translate	Translation	Neural machine translation for 75+ languages. Real-time and batch. Custom terminology.
Amazon Polly	TTS	Text-to-speech with 60+ voices, 30+ languages. Neural TTS for near-human quality. SSML for pronunciation control. Customise pronunciation with lexicons.
Amazon Transcribe	ASR	Speech-to-text for audio/video files and real-time streaming. Speaker diarisation, custom vocabulary, medical transcription (Transcribe Medical), content redaction.

Amazon Lex	Conversational AI	Build conversational chatbots and voice bots. Understands intent and extracts entities. Powers Alexa's NLU. Integrates with Connect, Kendra, Lambda.
Amazon Kendra	Enterprise Search	Intelligent search powered by ML. Natural language queries over structured and unstructured data sources. 40+ connectors. Understands context and semantics, not just keywords.
Amazon Personalize	Recommendations	Real-time personalised recommendations and personalised rankings without ML expertise. Same technology as Amazon.com's recommendation engine. Upload interaction history, train, deploy API.
Amazon Forecast	Time Series	Fully managed time series forecasting using AutoML. Incorporates related data (weather, pricing). Multiple algorithms: DeepAR+, ARIMA, Exponential Smoothing, CNN-QR.
Amazon Fraud Detector	Fraud	Build ML-based fraud detection models without ML expertise. Uses Amazon's fraud detection expertise. Account fraud, online fraud, transaction fraud.
Amazon Lookout for Vision	Defect Detection	Visual defect detection and anomaly detection for industrial inspection. Train with as few as 30 images. Real-time inference at the edge.
Amazon Lookout for Equipment	Predictive Maintenance	Detect abnormal equipment behaviour using sensor data. Trained on normal operational data. Predict failures before they occur.
Amazon Monitron	IoT Monitoring	End-to-end system for monitoring industrial machinery. Hardware sensors + ML inference + mobile app.
Amazon Panorama	Edge Vision	Computer vision appliance + SDK for running CV models at the edge on cameras. No cloud upload needed.

Amazon HealthLake	Healthcare	FHIR-compliant health data lake with built-in ML for medical insights. Store, transform, query clinical data.
AWS HealthScribe	Medical NLP	Automatically generate clinical notes from patient-physician conversations. HIPAA-eligible.

6.2 Comparison: Similar-Sounding Services (Exam Traps)

The exam frequently tests your ability to distinguish between services with overlapping names:

If You Need To...	Use This Service	NOT This
Search across company documents intelligently	Amazon Kendra	Amazon Elasticsearch (OpenSearch)
Build an interactive chatbot/voice bot	Amazon Lex	Amazon Kendra
Read text from images/scanned PDFs	Amazon Textract	Amazon Rekognition (Rekognition reads simple text in images, Textract understands document structure)
Detect faces, objects, activities in images/video	Amazon Rekognition	Amazon Textract
NLP on text (sentiment, entities, PII)	Amazon Comprehend	Amazon Kendra
Predict time series / demand forecasting	Amazon Forecast	SageMaker (Forecast is managed; SM gives full control)
Make personalised product recommendations	Amazon Personalize	Amazon Forecast
Access large foundation models via API	Amazon Bedrock	SageMaker JumpStart
Build and train a completely custom ML model	Amazon SageMaker	Amazon Bedrock
One-click deploy pre-trained open-source LLMs	SageMaker JumpStart	Amazon Bedrock
No-code AutoML for business users	SageMaker Canvas	SageMaker Autopilot
Automated feature engineering	SageMaker Data Wrangler	SageMaker Feature Store
Enterprise AI chatbot over internal documents	Amazon Q Business	Amazon Kendra (Kendra provides search; Q Business provides full conversational interface)
AI code assistant in IDE	Amazon Q Developer	Amazon CodeGuru

Code quality review and security scanning	Amazon CodeGuru	Amazon Q Developer
---	-----------------	--------------------

WARNING: The Textract vs Rekognition distinction is a classic exam question. Rekognition DetectText identifies text in natural scenes (store signs, license plates). Textract extracts structured text and data from documents (forms, tables, invoices).

6.3 Amazon Bedrock — Full Feature Reference

Key Bedrock APIs and concepts to know for the exam:

Feature/API	Description
InvokeModel	Synchronous invocation of a foundation model. Send prompt, receive response. Used for most LLM tasks.
InvokeModelWithResponseStream	Streaming invocation. Returns response tokens as they're generated — better UX for long outputs.
Converse API	Unified conversation API that works across all Bedrock models with standardised message format.
RetrieveAndGenerate	Knowledge Base integrated generation. Retrieves relevant chunks from KB, augments prompt, generates grounded response.
Retrieve	Retrieve-only API for Knowledge Bases. Returns relevant document chunks without generation.
InvokeAgent	Invoke a Bedrock Agent to handle multi-step tasks.
Model IDs	Each model has a unique ID used in API calls. Format: provider.model-name:version (e.g., anthropic.claude-3-sonnet-20240229-v1:0)
Inference Parameters	Model-specific: temperature, top_p, top_k, max_tokens, stop_sequences. Critical for output quality control.
Custom Model Import	Import your own fine-tuned model weights into Bedrock for managed hosting.
Batch Inference	Process large batches of prompts asynchronously. Results stored in S3.
Bedrock Studio	Web UI for building and testing generative AI prototypes collaboratively.
Amazon Bedrock Flows	Visual builder for creating multi-step generative AI workflows. Node-based pipeline builder.
Multi-Agent Collaboration	Supervisor agent that decomposes tasks and delegates to specialist sub-agents.
Memory in Agents	Agents can retain information across multi-session conversations via memory storage.

6.4 AWS AI Infrastructure

High-performance AI training and inference requires specialised hardware. AWS provides the broadest range of ML-optimised compute in the cloud:

Hardware	Type	Optimised For
NVIDIA A100 / H100 GPUs	P4d, P4de, P5 instances	Large model training, fine-tuning, inference — industry standard
AWS Trainium (Trn1)	Trn1 instances	Custom AWS chip for deep learning training. Up to 50% cost savings vs GPU for training large models.
AWS Inferentia (Inf2)	Inf1, Inf2 instances	Custom AWS chip optimised for low-cost, high-throughput inference. Run LLMs cheaply at scale.
NVIDIA GPUs (T4, A10G)	G5, G4dn instances	Inference workloads, smaller model training, graphics
Intel/AMD CPUs	M-series, C-series	ML inference on classical models, feature engineering
AWS Graviton (ARM)	Trn1n, Inf1	Cost-efficient inference for CPU-based workloads

EXAM TIP: For the exam: Trainium = training (cost savings). Inferentia = inference (cost savings). Both are custom AWS chips. NVIDIA GPUs (P3, P4, P5) are for full flexibility.

6.5 Distributed Training at Scale

Training large models requires distributing work across many GPU nodes. SageMaker and EC2 support several parallelism strategies:

Strategy	Description
Data Parallelism	Split the training data across multiple devices. Each device has a full copy of the model. Gradients are synchronised across devices after each batch. Easiest to implement. SageMaker Distributed Data Parallel (SMDDP).
Model Parallelism	Split the model layers across multiple devices. Necessary when a model is too large to fit on a single GPU. Each device holds a subset of model layers. SageMaker Model Parallel (SMP) library.
Tensor Parallelism	Split individual weight matrices across devices. Used for very large models. Supported by Megatron-LM, DeepSpeed.

Pipeline Parallelism	Divide model into stages; different stages run on different devices in a pipeline. Reduces idle GPU time.
SageMaker Hyper-Pod	Distributed training cluster management for large-scale foundation model training. Resilient to node failures without restarting training.

7 Exam Strategy, Tips & Quick Reference

7.1 Exam Strategy

Question Reading Strategy:

- Read the question stem completely before looking at answer options
- Identify the key requirement: 'MOST cost-effective', 'LEAST operational overhead', 'quickest to implement', 'no ML expertise'
- Eliminate obviously wrong answers first
- For 'which service' questions: match requirements to service strengths
- For scenario questions: look for the scenario that uniquely matches the described problem
- Trust your first instinct — unless you're clearly confident it's wrong

Common Exam Decision Patterns:

Scenario	Answer Pattern
No ML expertise needed, use AI quickly	→ Pre-built AI services (Rekognition, Comprehend, Translate, etc.)
Need custom model, ML team available	→ Amazon SageMaker
Access to foundation models, generative AI	→ Amazon Bedrock
Reduce hallucinations, use private knowledge	→ RAG / Bedrock Knowledge Bases
Model produces inconsistent format, needs style change	→ Fine-tuning (not RAG — RAG adds knowledge, not style)
Multi-step autonomous task execution	→ Bedrock Agents
Enterprise Q&A; over internal documents	→ Amazon Q Business
AI coding assistant	→ Amazon Q Developer
Prevent harmful model outputs	→ Bedrock Guardrails
Detect AI bias	→ SageMaker Clarify
Monitor model quality in production	→ SageMaker Model Monitor
Explain model predictions	→ SageMaker Clarify (SHAP)
No-code ML for business users	→ SageMaker Canvas
PII detection in documents	→ Amazon Comprehend or Macie (for S3)
Reduce inference cost at scale	→ AWS Inferentia (Inf2)

Reduce training cost	→ AWS Trainium (Trn1) or Spot Instances
----------------------	---

7.2 Critical Definitions to Memorise

Term	Definition
Foundation Model (FM)	Large, general-purpose AI model trained on vast diverse data. Can be adapted for many tasks via prompting or fine-tuning.
Prompt Engineering	Designing inputs to LLMs to elicit desired outputs without changing model weights.
RAG	Retrieval-Augmented Generation. Enhance LLM responses by retrieving relevant external documents at query time.
Fine-tuning	Further train a pre-trained model on task-specific data to change its behaviour.
LoRA	Low-Rank Adaptation. Efficient fine-tuning that trains only small rank-decomposition matrices.
Hallucination	LLM generates confident but factually incorrect or fabricated content.
Temperature	Sampling parameter controlling output randomness. 0 = deterministic, higher = more random.
Context Window	Maximum tokens an LLM can process in one interaction (input + output).
Embedding	Dense vector representation of text that captures semantic meaning.
Vector Store	Database for storing and efficiently searching embedding vectors.
Guardrails	Safety layer applied to FM inputs and outputs to filter harmful content and enforce policies.
Inference	Running a trained model to generate predictions on new data.
Overfitting	Model learns training data too well, fails to generalise to new data.
Data Drift	Production input distribution diverges from training distribution over time.
RLHF	Reinforcement Learning from Human Feedback. Align LLMs with human preferences.
Tokenisation	Process of breaking text into tokens (subwords) for LLM input.
Zero-shot / Few-shot	Prompting without / with a few examples respectively.
Chain-of-Thought	Prompting technique that instructs LLM to reason step by step.
Amazon Bedrock	Fully managed FM service. Access multiple FMs via single API. Serverless.

SageMaker	Comprehensive ML platform for custom model development, training, and deployment.
-----------	---

7.3 Frequently Tested Topics Summary

Domain	Must-Know Topics
Domain 1 (20%)	ML types (supervised/unsupervised/RL), ML workflow stages, evaluation metrics (precision/recall/F1/RMSE/AUC), overfitting/underfitting, neural network basics, AWS AI service tiers
Domain 2 (24%)	LLM concepts (parameters, context window, temperature, tokens, hallucination), Bedrock features, generative AI use cases, embedding and vector stores, Amazon Q suite, Titan models
Domain 3 (28%)	Prompt engineering techniques (zero/few-shot, CoT, system prompts), RAG architecture, Bedrock Knowledge Bases, Bedrock Agents, fine-tuning vs RAG decision, LoRA/PEFT, SageMaker components, evaluation metrics (ROUGE, BLEU, BERTScore)
Domain 4 (14%)	Responsible AI principles, bias types and mitigation, hallucination causes and mitigation, Bedrock Guardrails categories, PII protection, explainability (SHAP/LIME), SageMaker Clarify
Domain 5 (14%)	Shared responsibility model for AI, IAM for Bedrock/SageMaker, Bedrock Guardrails (prompt injection), data governance (Lake Formation, Macie), drift detection, CloudTrail audit, compliance frameworks

7.4 The AWS Shared Responsibility Model for AI

The Shared Responsibility Model divides security duties between AWS and the customer. For AI services:

Party	Responsibilities
AWS Responsibility (Security OF the Cloud)	Physical security of data centres; hardware (GPU servers); hypervisor and virtualisation layer; security of managed service infrastructure (Bedrock APIs, SageMaker control plane); global network; maintaining compliance certifications for the infrastructure.
Customer Responsibility (Security IN the Cloud)	IAM policies and roles; encryption of your data at rest and in transit; network security (VPC, security groups, NACLs); application security; your training data content and labelling quality; model outputs and their appropriateness; Bedrock Guardrails configuration; fine-tuning data privacy; access key management.

KEY CONCEPT: Managed services like Bedrock and SageMaker shift more responsibility to AWS compared to EC2. The more managed the service, the more AWS owns. But data security, IAM, and application logic always remain your responsibility.

7.5 Practice Scenario Questions

Scenario 1

Scenario: A healthcare company wants to extract structured data (patient name, date of birth, diagnosis codes, medication names) from scanned medical forms without writing custom ML code.

Answer: Amazon Textract AnalyzeDocument. Textract understands form structure (key-value pairs) and can extract from scanned documents with high accuracy. For specific medical entity extraction from the result, pipe through Amazon Comprehend Medical.

Scenario 2

Scenario: An e-commerce company wants to reduce product returns by building a Q&A; chatbot that answers customer questions using the company's existing product manuals and FAQ documents. The content updates monthly.

Answer: Amazon Bedrock Knowledge Bases (RAG) + Bedrock Agents or Amazon Q Business. RAG is ideal when the knowledge source is private, updates frequently, and you need grounded, cited answers.

Scenario 3

Scenario: A data science team notices their fraud detection model's performance has degraded over the last 3 months, even though the model hasn't changed.

Answer: SageMaker Model Monitor for data drift detection. The input data distribution has likely shifted (concept drift). Configure Model Monitor to detect statistical deviations from the training baseline.

Scenario 4

Scenario: A startup wants to build a customer service chatbot trained on their specific company tone, terminology, and common questions. They have 500 example customer support conversations.

Answer: Fine-tuning via Amazon Bedrock (custom models) or SageMaker JumpStart. With 500 labelled examples, fine-tuning is appropriate to teach company-specific style and terminology. RAG alone wouldn't change the conversational style.

Scenario 5

Scenario: A legal team wants to use an LLM to summarise contracts but is concerned it might invent legal clauses not in the original document.

Answer: Use RAG with explicit grounding instructions. Feed the contract as context. Set low temperature. Configure Bedrock Guardrails with a grounding check. Instruct the model: 'Summarise only information present in the provided document. Do not add information not in the contract.'

Scenario 6

Scenario: A developer wants to integrate personalised movie recommendations into an app without training ML models. They have 2 years of user watch history.

Answer: Amazon Personalize. Upload interaction history, train a recipe, deploy API. No ML expertise required. Returns personalised rankings in real time.

✓ Final Checklist & Good Luck

- AI/ML fundamentals — supervised, unsupervised, RL, deep learning architectures
- ML workflow stages and best practices
- Evaluation metrics — know which to use when and what each measures
- Amazon Bedrock — features, models, APIs, Knowledge Bases, Agents, Guardrails
- RAG — end-to-end pipeline, when to use vs fine-tuning
- Prompt engineering — all major techniques and when to apply
- Fine-tuning — LoRA, PEFT, when it's needed, Bedrock Custom Models vs SageMaker
- SageMaker — all major components and their roles
- Pre-built AI services — which service for which task
- Responsible AI — all 8 principles, bias types, hallucination mitigation
- Bedrock Guardrails — all content categories and when to use
- Security — IAM, VPC, KMS, Macie, CloudTrail for AI workloads
- Compliance — GDPR, HIPAA, SOC 2, Shared Responsibility Model
- Drift types and SageMaker Model Monitor
- Amazon Q Business vs Q Developer vs Kendra vs Lex (classic exam differentiators)
- Trainium (training) vs Inferentia (inference) — cost-optimised AI chips

You've got this. Study smart. Trust the process. Best of luck on AIF-C01!

© 2025 Wreetojyoti Ray — All Rights Reserved. wreetojyoti.in