MODULE 8 · STUDY NOTES

Implementing Generative AI

The generative AI application lifecycle, AI agents in Amazon Bedrock, and matching the AWS service to the use case

This module is where the course's pieces click together. Prompt engineering, RAG, fine-tuning, model evaluation, and responsible AI stop being isolated techniques and take their places in a single ordered workflow — the generative AI application lifecycle — that carries an idea from business problem to monitored production system. Along the way it introduces AI agents (the orchestration layer that lets an LLM call tools and data sources), and it closes with the practical question every organization faces first: given our skills, data, and appetite for control, do we build on Amazon Q Business, Amazon Bedrock, or Amazon SageMaker AI?

LEARNING OBJECTIVES

- Discuss the workflow to develop a generative AI application
- Review key considerations when selecting a foundation model (FM)
- List metrics to evaluate performance of foundation models
- Map AWS generative AI services and a generative AI workflow for a given use case

8.1 The generative AI application lifecycle

The lifecycle outlines specific steps for applying the techniques from earlier modules — prompt engineering, Retrieval Augmented Generation (RAG), fine-tuning, model evaluation, responsible AI — to a real-world implementation. This module views it through two lenses: an implementation of **Amazon Q Business** and an application built on **Amazon Bedrock**. The lifecycle is **iterative**: insights from deployment and real-world usage drive refinements in earlier stages — sometimes another round of performance work, sometimes all the way back to redefining the use case.

TABLE 8.1 The six stages of the lifecycle

STAGE	WHAT HAPPENS
1 · Define a use case	Identify specific business problems or opportunities where generative AI improves things for customers — employees, partners, patients, other stakeholders. Technical teams and business stakeholders align on goals, resources, and skill sets
2 · Select a foundation model	Choose an appropriate pre-trained model against the selection criteria from the Working with Foundation Models module
3 · Improve performance	Enhance the selected model for the use case: prompt engineering, augmentation such as RAG, or fine-tuning with domain-specific data
4 · Evaluate results	Rigorously assess performance with relevant metrics and test datasets; iterate between stages 3 and 4 until a minimally viable project is ready for a live environment
5 · Deploy the application	Integrate the optimized model into production systems; implement infrastructure and monitoring tools; establish processes for maintenance and updates
6 · Monitor, audit, get feedback	Ongoing proactive and reactive monitoring across infrastructure, model performance, and compliance, so issues are detected and answered quickly

Defining the use case. AWS teams practice **working backwards**: start by defining the customer experience, then iteratively work backwards until it is clear what to build and the value it adds — customers at the heart of the business, customer-centric innovation, and analytics plus cloud computing applied to common business problems. Concretely, articulate the specific problem, gather all relevant requirements — necessary **inputs, outputs, constraints, and performance expectations** — and confirm that generative AI is actually the right solution. Surface potential **ethical implications and responsible-AI risks early**, along with data privacy and security requirements.

> ORGANIZATIONAL READINESS – IS THE ORG A GOOD FIT?
>
> - Evaluate **technical infrastructure and capabilities**.
> - Assess **data availability and quality** — what data does the project need, and is high-quality data actually available?
> - Consider **regulatory and compliance** requirements.
> - Determine **resource availability** — budget and skills — including the compute, storage, and cloud resources required.
> - Gauge **cultural readiness** for AI adoption.
> - Define **success metrics** every stakeholder agrees on — productivity gain, time savings, customer-satisfaction rating, quality improvement, cost savings — so goals, scope, and impact are shared.

Selecting the foundation model. The criteria are the ones from Working with Foundation Models — model capabilities, size and complexity, inference speed, costs, information about its training data, ethical considerations, implementation considerations — but production sharpens them. How many requests per time frame (**throughput**)? What overall volume? What **context window** does the use case need — remembering the window counts tokens for *both* the input prompt and the output? A chat application can live with a smaller window than an app that writes reports from documents supplied in the prompt.

On cost, look past the sticker: some models charge upfront **license fees**, some a **flat fee per API call**, others **per input and output token** — and under token pricing, understanding and managing input and output sizes *is* cost

management. On ethics: with a pre-trained model you **don't control the training data**, and depending on the provider there may be little clarity about the datasets used — so budget real effort for mitigating bias and fairness risks. Trade-offs are inherent, prioritized by your organizational readiness assessment.

Improving performance. Everything learned so far applies here: **adjust inference parameters**, **perform prompt engineering**, **implement RAG**, or **customize the model through fine-tuning or continued pre-training**. These levers shape speed as well as relevance — well-structured prompts that yield concise responses, and inference parameters that cap output size, complete each request faster and lift throughput. RAG carries **less upfront cost and effort**, but a **fine-tuned model in the same domain can probably respond faster**. Finding the right trade-off is the work.

TABLE 8.2 Speed, throughput, cost, and availability levers

LEVER	WHAT IT BUYS YOU
Concise prompts + output limits	Each request completes more quickly — better throughput at no infrastructure cost
SageMaker AI inference options	For self-deployed models: options designed for real-time, and batch options for offline inference with no immediate-response requirement
Bedrock Provisioned Throughput	Guaranteed throughput for large, consistent inference workloads
Bedrock batch inference	Submit multiple prompts, generate responses asynchronously — priced lower than standard on-demand
AI agents	Use LLMs more efficiently; fold additional data and workflow steps into the application (next section)
Managed services	Offload undifferentiated infrastructure work; weigh control kept vs. lifting offloaded. Availability tolerance varies — a real-time medical help line tolerates far less downtime than a plant-care chat app

EVALUATION BEST PRACTICES

- Align evaluation criteria with the **specific business objectives and use case**.
- Use a **combination of evaluation methods** for a comprehensive view.
- Consider **both quantitative metrics and qualitative assessments**.
- Evaluate **continuously throughout the lifecycle** — not as a single step.
- **Adapt** evaluation strategies as the model and its applications evolve.

Test across multiple datasets. A common misconception is that models are inherently good or bad; in reality performance is a function of **both the model and the specific dataset** it is tested on. A model might show excellent performance on dataset A over time, even better results on dataset B, and progressively worse performance on dataset C. And datasets are not static — they evolve as real-world data distributions shift, so a model that performs well initially can degrade later: **model drift**. Using multiple diverse test datasets and tracking performance over time reveals strengths, weaknesses, and where to improve — and keeps models effective as they meet new production data.

TABLE 8.3 A combination of evaluation techniques

METHOD	EXAMPLES	CHARACTER
Human evaluation	Reviewers assess outputs	Time-consuming and potentially expensive, but invaluable for qualitative aspects: relevance, coherence, appropriateness
Popular benchmarks	GLUE, SuperGLUE, SQuAD	Standardized datasets covering a range of natural-language understanding and generation tasks; consistent comparison points across models
Automated metrics	Perplexity, BLEU	Efficient, scalable quantitative measures; quick insight into performance, though they miss nuances of language quality

Both **Amazon SageMaker Clarify** and **Amazon Bedrock model evaluation jobs** support automated evaluation jobs and jobs that incorporate human evaluation.

> **COMMON PITFALL**
>
> "The model passed evaluation" is never a permanent state. Performance belongs to the **model–dataset pair**, and production data keeps moving. The exam expects: multiple diverse test datasets, quantitative *and* qualitative methods, and continuous re-evaluation to catch drift.

Deploying the application moves it out of the test environment into production, where intended users generate real outputs. Scope depends heavily on how you deploy: **with AWS managed services, you don't provision infrastructure, manage security plumbing, or implement monitoring — AWS handles those**. Otherwise: configure compute, storage, and networking for expected traffic and data volumes, and integrate the FM as the "backend" or as a new component of an existing application; carry your **defense-in-depth** security into production — access control properly scoped, data protections active; implement robust monitoring of **technical metrics** (latency, error rates), **business metrics** (user engagement, task completion), and **responsible-AI issues** such as bias detection — Amazon CloudWatch automates much of this; and **prepare staff and implement change management**. Continuous optimization — regular model updates, performance tuning, adapting to changing needs and data — keeps the application effective over time.

> **MONITOR, AUDIT, AND GET FEEDBACK – KEY METRICS**
>
> - **Compute resources, network performance, storage usage** — spot bottlenecks and scale up or down.
> - **System logs** — surface application errors, security issues, and operational problems.
> - **Model performance metrics** — accuracy and latency; degradation signals potential model drift requiring retraining.
> - **Bias and fairness metrics** — watch for unfair treatment across demographic groups.
> - **Compliance adherence** — regulations, policies, and ethical principles.

TABLE 8.4 Monitoring tools – and the actions they drive

TOOL	ROLE
Amazon CloudWatch	Set up automated alerts on the metrics you define; logs support response analysis
SageMaker Model Monitor	For models you deploy yourself: detects **data and model drift** by analyzing inputs and predictions over time; real-time monitoring for endpoints or scheduled monitoring for batch jobs; monitors input-data quality; alerts when quality thresholds are breached
A/B testing	Compare model versions to validate updates before full deployment
Amazon Augmented AI (A2I)	Human-review workflows for ML predictions — automatically selects random samples or predictions with **low confidence scores**, manages reviewers at scale, and feeds human feedback into training data for retraining

Acting on monitoring data blends automated and human-driven responses: analyze system logs, input data, and predictions to find the **root cause** of degradation; **retrain with new data** when drift is detected; **adjust compute and storage allocation** to observed usage; **trigger human review** for low-confidence predictions needing expert judgment; use deployment strategies like **canary releases** when validating model updates; and **continuously refine thresholds and metrics** as the system evolves. The goal: high model quality, reliable performance, and compliance over time — with the combination of Model Monitor's automated vigilance and A2I's targeted human insight also supporting responsible-AI accountability in high-stakes domains.

8.2 Using AI agents in Amazon Bedrock

AI agents are autonomous or semi-autonomous software entities designed to perform tasks or solve problems on behalf of users. In generative AI they are usually built with an **LLM as the core**, combined with components that interact with **external tools, databases, or APIs**; they process natural-language inputs, make decisions, and take actions toward specific goals. Think of a really good front-desk person: they take your request, know what information to fetch and who to involve, and hand the specialist a complete file. Example tools for creating agents: **LangChain, AutoGPT, Microsoft Semantic Kernel, and Amazon Bedrock Agents**.

WHAT AMAZON BEDROCK AGENTS DO

- **Orchestrate** interactions between the foundation model, data sources, software applications, and user conversations.
- **Extend FMs** to understand user requests and break the required tasks into smaller steps.
- **Collect additional information** from a user through natural conversation.
- **Take actions** by automatically making API calls to your company systems.
- **Augment performance and accuracy** by querying data sources and invoking knowledge bases.
- Fully managed: **no capacity provisioning, no infrastructure management, no custom code** — accelerating development of, e.g., an insurance-claims helper or a travel-reservation agent.

WORKED EXAMPLE – A TRAVEL SITE WITH A BEDROCK AGENT

(1) The user asks for a priced weekend itinerary for a chosen destination and style of weekend. **(2)** The agent recognizes it needs two things before involving the LLM: trip options with pricing from the **trip catalog database**, and totals from a **calculator app**. **(3)** It sends a search request to the trip catalog, which returns itinerary options with prices. **(4)** It passes that pricing data to the calculator app, which computes the total cost of each relevant option. **(5)** The agent bundles the catalog results, the calculated costs, and the original user request into a single prompt for the LLM. **(6)** The LLM generates a response that incorporates the retrieved data and returns it to the user. The agent improved the model's performance for the travel app — the LLM answered once, with everything it needed.

COMMON PITFALL

For the certification, you are tested on **why** an agent is used and **where it sits in the application flow** — not on building one (that requires function and API skills beyond this course). Key phrase to recognize: agents *orchestrate* the FM, data sources, applications, and conversations.

8.3 Matching the AWS service to the use case

AWS's generative AI services sit on a spectrum: **technical skills and control rise from Amazon Q, through Amazon Bedrock, to Amazon SageMaker AI — and complexity and cost rise with them**. If you need to **build your own model**, choose SageMaker AI, which retains full control across the entire ML lifecycle. To **use or customize a pre-trained model**, use **Amazon SageMaker JumpStart or Amazon Bedrock**. For a **no-code option**, choose **Amazon Q Business**. The techniques ladder up the same axis — prompt engineering and RAG at the accessible end, deeper customization in the middle, build-your-own at the top. For organizations just starting out, the **organizational assessment is a big part of service choice**: Q Business offers an entry point with a lower bar of technical skill and in-house resources, while mature ML development organizations may prefer SageMaker AI's full control.

TABLE 8.5 The three services, side by side

SERVICE	SWEET SPOT	PRICING & NOTES
Amazon Q Business	Launch quickly on internal documents. Implements RAG with your own data source — but you don't select, evaluate, or manage which models are used	Powered by Amazon Bedrock; lowest technical bar
Amazon Bedrock	Incorporate AI capabilities without significant ML expertise or resources — e.g., a customer-support organization enhancing its chat function	On-demand per token · batch inference · Provisioned Throughput
Amazon SageMaker AI	Unique or specialized needs requiring customized models and fine-grained control over architecture, training process, and deployment — e.g., a healthcare company predicting patient outcomes from specific biomarkers	Pay-as-you-go for compute, storage, and services across the ML lifecycle — flexible, but variable costs need careful management

CASE 1 – HR POLICY ASSISTANT → AMAZON Q BUSINESS

A large multinational wants an AI assistant so employees can quickly access and understand HR policies; goals: reduce HR workload, provide accurate responses. Organizational assessment: **limited development expertise, a very small operations staff, HR systems already on AWS with access controlled by IAM, speed-to-market priority** (a pilot to gauge response — they are new to generative AI), and **high-quality data** in policy documents, handbooks, and FAQs. The fit is **Amazon Q Business**: the ops team created a Q Business application and uploaded the documents; existing **IAM** grants each user the appropriate location-specific subset of HR documents; **guardrails** set global and topic-level controls; employees **rate responses**, and feedback improves accuracy; the knowledge base is regularly updated with policy changes. *Evaluation:* randomly selected HR queries, targeting a **95%-or-higher correct response rate**. *Success metrics:* employee feedback on usefulness and ease of use, reduction in HR ticket volume and time on routine inquiries, and analysis of usage frequency and question types. *Rollout:* pilot in one department, then company-wide; embedded in the intranet and HR portal; brief employee training sessions; **CloudWatch and CloudTrail** monitor performance and use.

CASE 2 – CUSTOMER-SERVICE CHAT → AMAZON BEDROCK

An e-commerce company wants an AI chatbot handling order tracking, returns, and product information — 24/7 support with less load on human representatives; goals: response accuracy, customer satisfaction, reduced agent workload. Organizational assessment: an **experienced software development team** eager to try generative AI, **no ML models built in-house** (one data scientist on analytics), and leadership **concerned about risk to the customer experience**. The fit is **Amazon Bedrock**: developers handle the front end and API calls without building, deploying, or managing models. They **compared several LLMs and chose a smaller model** that balanced natural-language understanding with faster performance than larger models; set up an **Amazon Bedrock Knowledge Base** with product catalogs and troubleshooting guides; and **fine-tuned** on a labeled dataset of past queries and answers. *Evaluation:* human evaluation combined with automated accuracy metrics. *Success metrics:* KPIs — response accuracy, customer-satisfaction scores, successful query-resolution rate — targeting a **25% improvement over historical values**. *Rollout:* start with a small percentage of customer traffic and increase gradually to manage risk; integrate into the existing customer-service platform and website; customers rate each interaction. *Monitoring:* **CloudTrail** audits the API calls made to the model; **CloudWatch logs** analyze responses; **Bedrock runtime metrics in CloudWatch** track performance.

TABLE 8.6 Cost considerations for the customized Bedrock model

COST COMPONENT	HOW IT'S CHARGED
Customized model training	Based on the total number of tokens processed by the model during training
Customized model storage	Charged per month, per model
Provisioned Throughput	**Required to run a customized model.** You purchase model units — each provides a maximum number of input/output tokens processed per minute — charged hourly; available with no time commitment, or at a discount for longer commitments
Standard on-demand	For comparison: pay-as-you-go per input token processed and output token generated (base models); batch inference costs less than on-demand

COMMON PITFALL

Service-choice questions hinge on the **organizational assessment**: short time frame + internal documents + thin technical bench → **Amazon Q Business**; software developers who don't want to build, deploy, and manage models → **Amazon Bedrock**; mature ML org needing full lifecycle control → **SageMaker AI**. Don't let a big company name pull you toward the heaviest service.

SAMPLE EXAM QUESTION, WORKED

A media company wants an AI system that generates personalized news articles from user preferences and browsing history. Which workflow is MOST effective? Key phrases: **generate personalized news articles** and **user preferences and browsing history**. Answer: **Amazon Bedrock with a fine-tuned FM** — Bedrock allows customization of pre-trained models, ideal for personalized content from user data. Why not the others: **SageMaker AI** (building a custom language model) could work but demands significant time and resources that pre-trained models make unnecessary; **Amazon Q Business** is suited to business intelligence and data analysis, not personalized content generation; **Comprehend + Lambda** fit text *analysis* tasks, not article generation.

Module summary

- The lifecycle: define a use case → select an FM → improve performance → evaluate results → deploy → monitor, audit, and get feedback — iterative, with production insights feeding any earlier stage.
- Define use cases by working backwards from the customer experience; gather inputs, outputs, constraints, and performance expectations; confirm generative AI actually fits; surface responsible-AI risks early.
- Assess organizational readiness — infrastructure, data quality, compliance, budget and skills, culture — and set agreed success metrics (productivity, time savings, satisfaction, quality, cost).
- Production model selection adds throughput, request volume, and context window (input + output tokens) to the familiar criteria — and cost structure matters: license fees, flat per-call fees, or per-token pricing.
- Improve performance with inference parameters, prompt engineering, RAG, and fine-tuning/continued pre-training; manage speed and cost with concise prompts, SageMaker inference options, Bedrock batch inference, and Provisioned Throughput.
- Evaluate with a combination — human evaluation, benchmarks (GLUE, SuperGLUE, SQuAD), automated metrics (perplexity, BLEU) — across multiple datasets, continuously: performance is a model-and-dataset property, and drift is real.
- Monitor infrastructure, logs, model performance, bias/fairness, and compliance; CloudWatch alerts, SageMaker Model Monitor, A/B testing, and A2I human reviews turn observations into retraining, reallocation, and refined thresholds.
- Agents orchestrate FMs, data sources, apps, and conversations (Bedrock Agents: no infrastructure, no custom code); service choice follows the org assessment — Q Business for no-code speed, Bedrock for developers without model ops, SageMaker AI for full control.

Review questions

1. Your piloted chatbot answers accurately but slows badly at peak traffic. Which lifecycle stage are you revisiting, and what are three levers to pull?

Answer: Improve performance. Levers: tighter prompts and inference parameters that cap output size (faster per-request completion), Bedrock Provisioned Throughput for guaranteed throughput on a consistent workload, batch inference for requests that don't need real-time answers — and possibly a smaller, faster model.

2. Why can a model that evaluated well at launch degrade months later, and what practice catches it?

Answer: Datasets aren't static — real-world data distributions shift, producing model drift. Continuous evaluation across multiple diverse test datasets, plus production monitoring (e.g., SageMaker Model Monitor), catches the degradation and triggers retraining.

3. Distinguish human evaluation, benchmarks, and automated metrics — and name AWS features supporting them.

Answer: Human evaluation judges qualitative aspects (relevance, coherence, appropriateness) but is slow and costly; benchmarks (GLUE, SuperGLUE, SQuAD) are standardized datasets for consistent cross-model comparison; automated metrics (perplexity, BLEU) are fast, scalable quantitative scores that miss nuance. SageMaker Clarify and Bedrock model evaluation jobs support automated and human-evaluation jobs.

4. When would you favor RAG over fine-tuning for improving performance, and when the reverse?

Answer: RAG when upfront cost and effort must stay low or knowledge changes often; fine-tuning when domain response speed matters — a fine-tuned model in the same domain probably responds faster than RAG, at higher upfront investment.

5. Trace the six steps of the Bedrock agent travel-site example.

Answer: User requests a priced weekend itinerary → agent recognizes it needs catalog data and cost totals → queries the trip catalog database for options with prices → sends pricing to the calculator app for totals → bundles catalog results, costs, and the original request into one LLM prompt → the LLM returns a response built on the retrieved data.

6. The HR Policy Assistant team chose Amazon Q Business. Which facts from their organizational assessment made it the right call?

Answer: Limited development expertise and a very small ops staff (no-code fits), priority on speed to market for a pilot, high-quality documents ready to upload (policies, handbooks, FAQs), and existing AWS infrastructure with IAM — which Q Business uses to scope each user's document access.

7. List every cost component of running a customized model on Amazon Bedrock.

Answer: Training (total tokens processed during training), model storage (per month per model), and Provisioned Throughput — required for customized models, billed hourly per model unit (max tokens/minute), cheaper with a longer time commitment.

8. Name the five families of metrics to monitor in production, and the two services that pair automated drift detection with human review.

Answer: Compute/network/storage utilization, system logs, model performance (accuracy, latency), bias and fairness, and compliance adherence. SageMaker Model Monitor detects data/model drift automatically; Amazon A2I routes random samples or low-confidence predictions to human reviewers, whose feedback re-enters training data.