MODULE 7 · STUDY NOTES

Introducing Generative AI

What generative AI is, how foundation models and prompts make it work, the AWS services that deliver it, and Amazon Q Developer for coding

Generative AI is the branch of AI that does not just analyze data — it **creates** it: conversations, stories, images, videos, music, and code. This module explains what makes that possible and how to put it to work on AWS. At the center sits the **foundation model** — a single model pretrained on enormous amounts of data that, unlike a traditional ML model built for one narrow task, can be adapted to many. You will see how FMs relate to LLMs and to the deep-learning ideas from earlier modules, the three FM types (text-to-text, text-to-embeddings, and multimodal), and how a **prompt** — steered by prompt engineering — drives a model through inference to an output. From there the module surveys the AWS generative AI stack across three layers — compute (AWS Trainium and AWS Inferentia), foundation models as a service (Amazon Bedrock and Amazon SageMaker JumpStart), and applications — and closes on **Amazon Q Developer**, the coding assistant that speeds ML and application development across the whole software development lifecycle.

LEARNING OBJECTIVES

- Define generative artificial intelligence (AI)
- Differentiate between traditional machine learning (ML) and generative AI
- Use a foundation model (FM) in a Jupyter notebook to generate an image from text
- Identify the benefits of Amazon Q Developer
- Explain how Amazon Q Developer works with Amazon SageMaker to speed up development of an ML application

7.1 What is generative AI?

Generative AI is a type of artificial intelligence that can create new content and ideas — including conversations, stories, images, videos, and music. The generators behind it are powered by pretrained machine learning models called **foundation models (FMs)**; because these models are capable of producing the content, you don't have to. And the output is not final: AI-generated content can be **edited**, so you can make whatever modifications meet your needs.

WHAT SETS GENERATIVE AI APART

- It **creates** new content and ideas rather than only analyzing existing data.
- It is powered by large pretrained models called **foundation models (FMs)**.
- Its output can be **edited and refined** after it is generated.
- It applies to a **broad range of purposes** — customer-facing, employee-facing, operational, and creative alike.

Generative AI applies across a broad range of use cases. The common thread is that a single general-purpose capability — generating relevant new content on request — can be pointed at very different business problems. The module groups the possibilities into four themes.

TABLE 7.1 Generative AI use cases

USE CASE	HOW GENERATIVE AI HELPS
Enhance customer experiences	Chatbots and virtual assistants streamline self-service and cut operational costs; summarize and analyze customer conversations; personalize experiences
Boost employee productivity	Accelerate application development, automate report generation, and improve content search through a conversational interface
Optimize processes	Intelligent document processing that extracts and summarizes data through Q&A; improve supply-chain logistics; generate synthetic data to train ML models
Enhance creativity and content	Produce marketing and sales content matched to a prospect's profile; generate multiple design prototypes to speed the ideation phase

7.2 How does generative AI work?

Generative AI is a subset of machine learning. It is powered by foundation models, and those models themselves make use of **deep-learning techniques such as neural networks** — so the nested picture from earlier modules still holds: AI contains ML, ML contains deep learning, and generative AI sits inside as the part built on FMs.

At the core of every AI generator is a foundation model. FMs are a class of powerful ML models set apart by being **pretrained on vast amounts of data** so they can perform a **wide range of downstream tasks** — text generation, data summarization, information extraction, question-and-answer responses, and chatbot interactions, plus generating images, music, videos, or code. A **traditional ML model**, by contrast, is trained to perform one **specific task** from a single dataset.

TRADITIONAL ML MODELS	FOUNDATION MODELS
siloed — one model per task	*general-purpose — pretrain once, adapt many*
▪ Built for a specific task: sentiment analysis, image classification, forecasting trends ▪ Each task needs its own labeled data ▪ Gather data, train a model, deploy it — then repeat for the next task ▪ Narrow, single-purpose output	▪ Pretrained on vast amounts of data ▪ One pretrained model adapted to several downstream tasks ▪ Customizable to domain-specific functions using a small fraction of data and compute ▪ Large and general-purpose by design

The difference is the whole workflow. For a traditional model, each task means gathering **labeled data**, training a model, and deploying it — then starting over for the next task. With foundation models you **pretrain once** on data and **adapt** the same model to several tasks. You can also customize an FM to perform **domain-specific functions** that differentiate a business, and doing so uses only a **small fraction** of the data and compute that training a model from scratch would require.

TABLE 7.2 What tasks foundation models can do

CAPABILITY AREA	EXAMPLE TASKS	EXAMPLE FM
Language processing	Text generation, Q&A, sentiment analysis, text summarization, search	Amazon Titan
Visual comprehension	Image generation, image classification	Stable Diffusion
Code and media	Code generation, audio generation, video generation	—

A subset of FMs called **large language models (LLMs)** are trained on **trillions of words** across many natural-language tasks. They can understand, learn, and generate text that is **nearly indistinguishable from human writing**, and — like FMs generally — one pretrained LLM adapts to tasks such as text generation, summarization, or sentiment analysis. LLMs can also be **customized with domain-specific knowledge**, such as a particular business's internal documents.

KEY TERMS

Text-to-text FM
Built for natural-language processing: takes a user's input text and extends it with new generated text — summarize, extract information, answer questions, create content (for example, sentence auto-completion).

Text-to-embeddings FM
An LLM type that compares pieces of text — such as what a user types into a search bar against indexed data — and connects them. Amazon's product search uses this to match a user's ask with catalog data for more relevant results.

Multimodal FM
Understands and generates different formats, such as text and images. Stable Diffusion is an example.

Generative AI architectures use **prompt engineering** to initiate an action from a foundation model. Prompt engineering is the process of **designing and refining the prompts** — the input stimuli — so a language model generates a specific type of output. It involves selecting appropriate keywords, providing context, and shaping the input to encourage the desired response, and it is a vital technique for actively shaping an FM's behavior and output.

> WORKED EXAMPLE – PROMPT, INFERENCE, OUTPUT
>
> A **prompt** is the text you input into the model; the model generates a response, and that act of generating the response is **inference**, whose result is shown as the **output**. Prompt: *Where is Japan located?* The model's inference on that prompt returns the output: *Japan is located in the northwest Pacific Ocean.* If the output is not what you want, you alter the prompt or provide examples of the task within the prompt — that refinement is prompt engineering.

TABLE 7.3 Structure of a prompt

PART	WHAT IT PROVIDES
Instructions	The task for the LLM to perform — a task description or instruction for how the model should act
Context	External information that guides the model
Input data	The input for which you want a response
Output indicator	The output type or format you want returned

DEMONSTRATION – EXPLORE AN FM AND USE PROMPT ENGINEERING

The recorded demo *Explore an FM and Use Prompt Engineering in a Notebook* explores foundation models in **Amazon SageMaker Studio** and shows how to use generative AI to **generate images from text prompts** — modifying the prompt engineering to change the result. This is the module objective of using an FM in a Jupyter notebook to generate an image from text.

COMMON PITFALL

Not every FM is an LLM, and not every FM writes prose. **LLMs are the text subset** of FMs; **text-to-embeddings** models compare and connect text (they power search, not content generation); and **multimodal** models handle formats such as images. On scenario questions, match the FM type to the job.

7.3 AWS generative AI offerings

AWS provides several generative AI offerings, arranged as a stack you can enter at any level: the **compute** that trains and serves models, **foundation models delivered as a service**, and ready-built **applications**. The five services below cover those layers.

TABLE 7.4 AWS generative AI offerings

OFFERING	LAYER	WHAT IT DOES
Amazon Q Developer	Applications	A generative AI coding companion that integrates with your IDE
Amazon Bedrock	FMs as a service	Fully managed service making FMs from AWS and leading AI startups available through an API; helps build apps that deliver up-to-date answers from proprietary knowledge sources
Amazon SageMaker JumpStart	FMs / ML	Prebuilt solutions for common use cases, deployable in a few steps; fully customizable, with CloudFormation templates and reference architectures
AWS Trainium	Compute	Second-generation ML accelerator purpose-built for deep-learning training of 100B+ parameter models
AWS Inferentia	Compute	Custom ML chip for high-performance inference predictions; set up an EC2 instance and use the AWS Neuron SDK to invoke Inferentia or Trainium

BENEFITS OF USING AWS FOR GENERATIVE AI

- **Flexibility** — a wide selection of FMs built by Amazon and top AI startups, including AI21 Labs, Anthropic, and Stability AI.
- **Secure customization** — with Amazon Bedrock, fine-tune a model for a task with **as few as 20 examples**; no customer data trains the underlying models, and all data is encrypted and stays within your VPC.
- **Cost-effective infrastructure** — best price-performance from AWS-designed ML chips and NVIDIA GPUs; scale cost-effectively to FMs with hundreds of billions of parameters.
- **Builder friendliness** — bring the model to your data using familiar controls and services such as Amazon SageMaker and Amazon S3, instead of sending your data out to the model.
- **Built-in AWS services** — services such as Amazon Q Developer add generative AI to boost productivity, and you can deploy solutions like call summarization and Q&A that combine AWS AI services with leading FMs.

COMMON PITFALL

AWS Trainium and AWS Inferentia are not interchangeable. Trainium is the training accelerator, purpose-built for deep-learning training of 100B+ parameter models; Inferentia is the chip for high-performance inference predictions. Both are used on an EC2 instance and invoked through the **AWS Neuron SDK**.

7.4 Amazon Q Developer

Amazon Q Developer is a generative AI-powered **coding assistant** designed for developers and IT professionals — including those without extensive experience. It is trained on years of high-quality AWS examples and documentation, and it can also be trained on your company's own code and systems. It generates code and helps you **understand, build, extend, and operate** AWS applications, and it is **secure and private by design**.

WHAT AMAZON Q DEVELOPER CAN DO

- Chat about code, provide **in-line code completions**, and generate new code.
- **Scan code for security vulnerabilities** and suggest remediations.
- Make code upgrades and improvements — language updates, debugging, and optimizations.
- Take a natural-language description of an application and generate and suggest the code for it — a **no-code** path well suited to simple web apps, automating business processes, and building prototypes or proofs of concept.

TABLE 7.5 Amazon Q Developer across the software development lifecycle

SDLC PHASE	WHAT IT PROVIDES
Plan	Ask questions and get referenceable, contextual guidance; explain code with conversational coding
Create	In-line code recommendations for multiple languages; implement features through comments or code prompts; chat in the IDE
Test and secure	Generate unit tests; scan project code for security vulnerabilities and get remediation suggestions
Operate	Troubleshoot errors; troubleshoot network connectivity with VPC Reachability Analyzer
Maintain and modernize	Modernize code with the Amazon Q Developer Agent for code transformation, upgrading projects to more current language versions

DEMONSTRATION – A LINEAR REGRESSION MODEL WITH AMAZON Q DEVELOPER

The recorded demo *Creating a Linear Regression Model Using Amazon Q Developer* shows how Amazon Q Developer accelerates coding tasks in ML applications by suggesting code and comments directly in **Jupyter notebooks**, increasing developer productivity — an illustration of how Amazon Q Developer works with Amazon SageMaker to speed up development of an ML application.

COMMON PITFALL

Amazon Q Developer is more than autocomplete. Beyond in-line suggestions it **generates unit tests, scans for security vulnerabilities, troubleshoots errors, and modernizes code** (via the Amazon Q Developer Agent for code transformation) across the entire SDLC — from **Plan** through **Maintain and modernize**.

Module summary

- Generative AI is a type of AI that creates new content and ideas — conversations, stories, images, videos, music, and code — powered by pretrained foundation models, and its output can be edited.
- Generative AI is a subset of ML; foundation models use deep-learning techniques such as neural networks, and one FM is pretrained once and adapted to many tasks.
- A traditional ML model is siloed to one task with its own labeled data; customizing a foundation model to a domain uses only a small fraction of the data and compute of training from scratch.
- LLMs are the text subset of FMs, trained on trillions of words; FMs come as text-to-text (extend text), text-to-embeddings (compare and connect text for search), and multimodal (text and images).
- A prompt (instructions, context, input data, output indicator) drives a model through inference to an output; prompt engineering refines the prompt to steer the result.
- AWS delivers generative AI in three layers — applications, foundation models as a service, and compute — via Amazon Q Developer, Amazon Bedrock, Amazon SageMaker JumpStart, AWS Trainium, and AWS Inferentia.
- Benefits of AWS for generative AI: flexibility, secure customization (fine-tune with as few as 20 examples; data encrypted in your VPC and never used to train shared FMs), cost-effective ML-purpose infrastructure, builder friendliness, and built-in services.

- Amazon Q Developer is a generative AI coding assistant that integrates with the IDE, scans code for security vulnerabilities, and supports developers across the full SDLC.

Review questions

1. **What is generative AI, and what powers it?**

   Answer: A type of AI that creates new content and ideas — conversations, stories, images, videos, music, and code — powered by pretrained ML models called foundation models (FMs); the generated content can be edited.

2. **How does a foundation model differ from a traditional ML model in how it is built and used?**

   Answer: A traditional ML model is siloed to a specific task and needs its own labeled data, training, and deployment for each task; a foundation model is pretrained once on vast data and adapted to many downstream tasks, and can be customized to a domain with only a small fraction of the data and compute of training from scratch.

3. **What is an LLM, and how does it relate to foundation models?**

   Answer: A large language model is a subset of FMs trained on trillions of words across natural-language tasks; it can understand, learn, and generate near-human text, adapt to many tasks, and be customized with domain-specific knowledge.

4. **Name the three foundation-model types and what each does.**

   Answer: Text-to-text (extends input text with new generated text — summaries, answers, content), text-to-embeddings (compares text inputs such as a search query with indexed data and connects them), and multimodal (understands and generates multiple formats such as text and images, e.g. Stable Diffusion).

5. **Trace what happens from prompt to output, and define prompt engineering.**

   Answer: You input a prompt (instructions, context, input data, output indicator); the model performs inference to generate a response, shown as the output. Prompt engineering is designing and refining the prompt — keywords, context, examples — to get a specific type of output.

6. **What are the three layers of AWS generative AI offerings, and name a service in each?**

   Answer: Compute (AWS Trainium, AWS Inferentia), foundation models as a service (Amazon Bedrock, Amazon SageMaker JumpStart), and applications (Amazon Q Developer).

7. **Why is customizing a model on Amazon Bedrock considered secure?**

   Answer: You can fine-tune with as few as 20 examples; no customer data is used to train the underlying models, and all data is encrypted and stays within your VPC.

8. **How does Amazon Q Developer support the software development lifecycle?**

   Answer: Across Plan (contextual guidance, explain code), Create (in-line recommendations, build from prompts, chat in the IDE), Test and secure (generate unit tests, scan for vulnerabilities), Operate (troubleshoot errors, VPC Reachability Analyzer), and Maintain and modernize (code transformation to upgrade language versions).