Foundation Model (FM) → Trained on a large volume of data

Large ML Data (Labelled Data) → Training → FM

Unlabelled Data → Training → FM

- Usually needed Data
  - Text, Pics, Code, Audio, Video, Sensor, Spec, Graph, etc.

FM → Text generation, Text summarization, Code generation, Video/Image generation, Video summarization, Chat/Q&A/Conversation

Amazon Bedrock – Fully Managed Service

FMs | LLMs → Amazon offers foundation

TRANSFORMER Architecture (LLM)

↳ Deep Learning Model Architecture

↳ Seq-to-seq tasks

- ↳ Language translation (Eng → French)
- ↳ Text Summarization
- ↳ Question Answering
- ↳ Speech to text

Text

2017 → Attention is All you need

Self-attention mechanism

Core Components of a TRANSFORMER

1. Input "Embedding" → Converts tokens (words) into vectors

I	love	cats
[0.1, 0.3]	[0.5, 0.7]	[0.9, 0.2]

token → numerical representation (vector)

2. Positional Encoding
   - Inject position info
3. Encoder → Processes input seq & generates a contextual representation
   - ↳ self-attention
4. Decoder → Generates the output sequence
5. Self-attention – understand context
6. Feed-forward network – refine network.

Step by step workflow

Input = "I love cats"

Task = Translate to French

Step 1: Tokenization & Embedding (vectors)

1. "I", "love", "cats"
2. Convert tokens into vectors (embeddings)

I → [0.2, 0.8, 0.1]

love → [0.9, 0.1, 0.5]

cats → [0.5, 0.7, 0.3]

3. Add positional Encoding

Step 2: Encoder

1. Compute Q, K, V for each word

love Q (query) → [0.7, 0.1]

K (key) → [0.3, 0.4]

V (value) → [0.2, 0.6]

2. Compute attention scores

"love" → "I" / "cats"

Step 3: Feed-Forward network (Encoder)

↳ refine contextual representation

Step 4: Encoder output

Step 5: Decoder = Input → Encoder output

output Projection

Aime	0.85
vous	0.45
les	0.2

"Inference Parameters" for Amazon Bedrock models

↳ How to control output diversity, coherency, length of model response

Invoke FM(s) in Amazon Bedrock → fine-tune the model behavior ⇒ Inference Parameters

1. Temperature → Controls randomness
   - (0 – 1)
   - 0.1 – 0.5 — 0.6 — 0.7 – 1
   - Low = more focused
   - High = more creative / varied
2. Top P → Limits tokens to top probable ones
3. Top K → limits the vocab size

vocabulary for seq

4. max_tokens → maximum output length = 1000 tokens
5. Stop sequences → Define termination string.
   - ↳ ("###")

Top K = How many tokens the model considers at each step

How it works?

If top K = 0.7 → tokens that together make up 70% probability

Eg

- cat – 0.50
- dog – 0.30
- car – 0.10
- hat – 0.05
- bat – 0.05

If top K = 0.8: cat/dog

If top K = 0.55: cat/dog / car/hat

Low = (0.3 – 0.6) → very focused

Med – (0.7 – 0.9) → Balanced

High – (0.95 – 1.0) → very creative

for chatbots/summarization → (0.7 – 0.9)

for creative writing/idea generation → (0.95 – 1.0)

If top P, top K → creative

Top K

cat	0.50
dog	0.30
car	0.10
hat	0.05
bat	0.05

Top P, Top K

If top K = 1 ⇒ cat

top K = 3 ⇒ cat/dog / car

Low = focused

High = creative / diff tokens

Med = balanced

Top P, Top K

0.9

Customer Support / FAQ (very deterministic, very focused) → Temp – Low — 0.2 – 0.3; Top P – Balanced — 0.7 – 0.8; Top K – Balanced — 50

Code Generation (very focused) → ——— ↓ ———

Content Summarizer

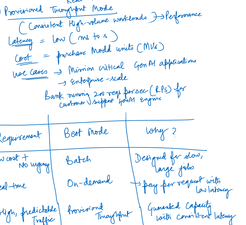
Bedrock – THREE INFERENCE ?

Inference ?

↳ Refers to the process of generating output from an input provided to the model

Application – Cost, Latency & workload pattern

1. Batch Inference (Offline)
   - REST – Non-real time / large-scale processing
   - COST → Most Cost-effective
     - ↳ 50% of Ondemand cost per request
   - Latency → High (minutes to hours)
   - Use Cases = ML/BI reports
     - Offline processing (summarize millions of documents overnight)
     - Sentiment analysis on 3 million internal customer reviews
2. On-Demand Inference (Real-time, interactive apps)
   - Latency = low (milliseconds to sec)
   - Cost → Higher per request
     - pay as you go
   - Use Cases → Chatbots, Q&A applications, Real-time recommendations
3. Provisioned Throughput Mode
   - (Consistent high-volume workloads → Performance)
   - Latency = Low (not as a)
   - Cost = purchase model units (MUs)
   - Use Cases → Mission critical GenAI applications
     - → enterprise-scale

Bank running 24 very for sec (RPS) for customer Support chatbot engine

Requirement	Best Mode	Why?
1. Low cost + No urgency	Batch	Designed for slow, large jobs
2. Real-time	On-demand	pay for request with low latency
3. High, predictable traffic	Provisioned Throughput	Guaranteed capacity with consistent latency