Recap on Day 1.

? Generative AI = Branch of ML, Subset of DL
= Create a new content - Conversation, image, video, text, code.

Use Cases : Text generation, Text summarization, Conversational (Chatbot, Q&A, VA, Agents), Code generation, Image generation.

Generative AI is powered FMs = very large models which are pretrained on Internet scale of data (Raw Data - MultiModal Data - Files (XML, JSON, JSONL, TXT, PDFs, etc), Image, Repo, Videos, etc.

Foundation Model ?
↳ Raw Data (unlabeled Data)
↳ TRANSFORMER Architecture
↳ Deep learning architecture (2017)
↳ Self-attention Mechanism.

TRANSFORMER - Components
① Input Embedding
↳ Tokenizations & Embedding
Input ⇒ tokens (words) → vectors (numerical representation)
② Positional Information
③ Encoder
↳ Encoder (Self attention)
Contextual representation
output of Encoder = Input for Decoder
↳ Feed Fwd Network (neural network)
④ Decoder (Self attention + Feed Fwd)
↳ output
⑤ output layer

★ Amazon Bedrock (fully managed service)

Generative AI Services (AWS)
① Build, deploy, manage - Your OWN ML Models
↳ Amazon SageMaker
② Customized parameter of FMs / LLMs
↳ Amazon SageMaker Jumpstart
③ Create / Build a Generative AI Applications using prebuild FMs / LLMs - fine-tune the data to FMs / LLMs
↳ Amazon Bedrock.

Amazon Bedrock.
① Model Catalog → Providers wise FMs / LLMs
② Playground - Chat playground
- Image playground
③ Build → Knowledge bases
- Agents
- Guardrails.
EXTRA

Day 2 - Agenda.
① Inference Parameters
② Amazon Bedrock - Application Components ✓
→ lab time.
③ Prompt Engineering
④ Amazon Bedrock - Foundation Models ✓
→ lab time
→ Demo - Agent, Guardrail.
⑤ Langchain ✓
↳ lab time.

Inference Parameters - How to control the model response.
① Temperature = Controls Randomness in output
Low → more deterministic & focused
HIGH → more creative & varied
② Top P (Probability) = limits tokens to top probability mass
0.9 = 90%
③ Top K = limits the vocabulary size per step.
④ MaxTokens = Set maximum output length

Top p = How many tokens the model considers for each step based on the cumulative probability.

How it works ?
If top p = 0.9 ⇒ Model considers tokens that together make up 90% probability.
⇒ Rare tokens (low probability) outside the 90% - IGNORED

Ex
cat = 0.50
dog = 0.30
car = 0.10
hat = 0.05
sun = 0.05

If topP = 0.8 ✓
cat / dog.
If top p = 0.95
cat / dog / car / hat

Effect on output
Top p value
Low (0.3 - 0.6) = very deterministic, focused
Med (0.7 - 0.9) = Balanced.
High (0.95 - 1.0) = very creative, but may lose accuracy
✓ Chatbots / Summarization → use 0.7 - 0.9 for coherent but varied answers.
✓ creative writing / Idea generation → use 0.95 to 1.0

(Temp & Top p.)

Top K ⇒ Claude family*.
↳ Controls how many tokens the model consider at each generation step.

How it works.
→ cat = 0.50
dog = 0.30
car = 0.10
hat = 0.05
sun = 0.05

If top k = 1 = cat
top k = 2 = cat / dog
top K = 5 = cat / dog / car / hat / sun

Effect on output
Low (1-5) → less creative output
Med (20-50) → Balanced
High (100-200) → Highly diverse

Use cases
↳ summarization / coding = Low
↳ Story telling / brainstorming = High.

Top K vs top P
Top k = fixed # of tokens - top 50 tokens.
Top P = probability mass cutoff - (90%)

Temperature = 0.7
Top k = 50 ✓
Top p = 0.9 ✓

Use Case	Temp	Top P	Top K
Customer Support / FAQ.	0.2 to 0.3	0.7 - 0.8	40 - 50
(very deterministic, coherent, aviod hallucinations)			
Content Summarization	0.2 to 0.4	0.8 - 0.9	50 - 100
(Concise, focused with minimum randomness)			
Code Generation	Low	Balance	Balance
(Generate Syntactically correct & mostly deterministic Code)			
Creative Storytelling	High (0.9 - 1.0)	High	High
(Highly creative, diverse, but less predictable.)			

+1
+2

Provider	FM	What they can do ?
Amazon	✓ Amazon Titan	— General purpose Models (Text generation, summarization, embedding)
AI21 labs	Jamba	→ Multi Lingual LLMs for Text generation in Spanish, French, German, Italian, Dutch.
Anthropic	✓ Claude	→ Intelligence, speed, cost
Cohere	✓ Command & Embed	→ Text generation Models for Business application & embedding model for search, clustering, classification in more than 100 languages
Meta	Llama	→ chat interfaces, VA, language Translations
Mistral AI	Mistral / Mixtral	→ Text Generation, Code Generation, RAG, Agents.
Stability AI	Stable Diffusion	→ Text to Image Models for generation of Image, (High-details)

Amazon Bedrock Methods (APIs)
① List Foundation Models.
Amazon Bedrock Runtime - related APIs
① Invoke Model ✓
② InvokeModelWithResponseStream
③ Converse →

Diffusion - Deep Learning Architecture
↳ Two-step process
① Forward diffussion (U-Net Model)
② Reverse diffusion (denoising)

image vector

Design effective Prompts
① Be clear & concise = Straightforward.
② Include context (if needed)
③ Consider the output in the prompt.
④ Start prompts with an interrogation
⑤ Provide an example (Few-shot prompt)

Stop Seq (Inference parameter)
↳ Specific string. = tell the model when to stop
↳ Even if the maxTokens limit is not reached.

Why ? ① Control the output format
② Ensure structured dialogue
③ Prevent unnessary token usage (Save cost)

How ? {JSON}
{
"input Text": "Generate a JSON respons with a name & age",
}
temperature : 0.3
maxToken : 200
StopSequence : ["}"]

Real world use cases.
① Structured JSON output = ["}"]
② Code generation = ["```"]
③ Chatbots & Q&A = ["\nUser:"] → Stop at next user prompt

LANGCHAIN (Integrate AWS)
langchain-aws
langchain-community
> Packages.
① Models.
② Prompt Templates
③ Indices
④ Memory
⑤ Chains
⑥ Agents