🧠 Domain 3: Applications of Foundation Models

Task 3.1: Design Considerations for Applications That Use Foundation Models

Welcome! In this part of your journey, we'll explore how to **design applications that use foundation models** effectively. Think of this as choosing the right tool for the job—except the tools here are massive, pre-trained models that can chat, summarize, generate images, write code, and more.

Let's break it down together.

📌 1. Choosing the Right Foundation Model

Before you build, you have to **pick a model**—and not all models are created equal. Here's what to consider:

Selection Criteria	Why It Matters
Cost	Bigger models usually cost more to run. Consider your budget and usage frequency.
Modality	Need to work with images, text, or audio? Choose a model trained on the right types of data (aka "modality").
Latency	Real-time chat needs fast responses. Choose low-latency models for interactive applications.
Multilingual Support	Serving global users? Select a model trained in multiple languages.
Model Size & Complexity	Larger models (e.g., Claude 3 Opus) are powerful but expensive and resource-heavy. Smaller models are cheaper but may underperform on complex tasks.
Customization Options	Do you need to fine-tune the model to your domain (finance, healthcare, etc.)? Look for customizable options.
Input/Output Length	Long documents? Choose a model with a large context window (e.g., 8K or 100K+ tokens). This impacts how much it can "remember" at once.

> **Pro Tip:** Use Amazon Bedrock to access a variety of models from different providers (Anthropic, Meta, Mistral, etc.) through one unified API—making comparison easier.

🎛️ 2. Inference Parameters: Controlling the Output

Once you've chosen a model, you still need to **tune how it behaves.** That's where inference parameters come in.

Common Parameters

Parameter	What It Does
Temperature	Controls randomness. Low = more deterministic, repeatable. High = more creative or diverse. Typical range: 0.1 to 1.0.
Max Output Length	Caps the number of tokens in the response. Prevents overly long replies (and cost overruns).
Top-k / Top-p Sampling	Limits how many likely next tokens are considered. Useful for balancing creativity and coherence.

> Think of these settings like turning knobs on a music mixer—you can adjust tone, tempo, and rhythm to get the right “sound” for your application.

🔍 3. What is Retrieval-Augmented Generation (RAG)?

Ever wish your GenAI model could **consult a knowledge base** before answering?

That’s exactly what **RAG (Retrieval-Augmented Generation)** does.

🧠 How RAG Works:

1. **User input** (like a question) comes in.
2. The system **retrieves relevant documents** from a knowledge base (e.g., product manuals).
3. These documents are **fed into the model** alongside the input.
4. The model **generates an answer** based on both the question *and* the retrieved content.

🏢 Business Applications:

- **Customer Support:** Give accurate, on-brand answers based on company documentation.
- **Internal Tools:** Help employees find policies, guidelines, or training material.
- **Search Augmentation:** Improve accuracy of chatbots or virtual assistants.

💡 RAG + AWS:

Use **Amazon Bedrock** + **Amazon OpenSearch Service** to build scalable, serverless RAG pipelines. You store your docs as **embeddings** in a **vector database**, and Bedrock uses them to ground the model’s responses.

📦 4. Where Do Embeddings Go?

Embeddings are **numerical representations of text**—like turning words into math the model can understand and compare. These are essential for similarity search (like finding the most relevant document in RAG).

AWS Services That Can Store Embeddings:

Service	Details
Amazon OpenSearch Service	Scalable, managed search engine with vector search support. Ideal for RAG.

Service	Details
Amazon Aurora	Managed relational DB. With pgvector extension (PostgreSQL), you can store and query vectors.
Amazon RDS for PostgreSQL	Similar to Aurora; use the pgvector extension for embedding storage.
Amazon Neptune	Graph database with semantic search features. Useful for complex knowledge graphs.
Amazon DocumentDB	MongoDB-compatible document store. Store embeddings alongside metadata.

> Choose your storage based on the data structure you need—OpenSearch is best for search-heavy use cases, while RDS is good for structured relational data.

💰 5. Understanding Customization Trade-Offs

You can customize foundation models in different ways—each with its own **cost, complexity, and use case**.

Approach	What It Is	Cost	Flexibility
Pre-training	Training a model from scratch.	Very expensive (millions of dollars)	Full control over everything
Fine-tuning	Training a pre-trained model on your data.	Moderate to high	High
In-context Learning	Providing examples in the prompt ("few-shot learning").	Low	Limited to current session
RAG	Augmenting model with retrieved data at inference time.	Moderate	Very flexible and safe

> TL;DR: RAG and in-context learning are faster and cheaper; fine-tuning gives more control; pre-training is rare outside big tech or research labs.

🤖 6. What Are Agents in Generative AI?

Imagine a model that doesn't just respond—it **takes action**. That's what agents do.

🛠️ Agents for Amazon Bedrock:

Agents can:

- **Break down multi-step goals** ("Book a flight and a hotel")
- **Call external APIs** ("Look up the latest exchange rate")
- **Use tools or plugins** (e.g., databases, knowledge sources)
- **Maintain memory/state** across interactions

This turns your GenAI model into a **workflow engine**, not just a text generator.

> Example: A travel bot that checks weather, books travel, and updates your calendar—autonomously.

✅ QUICK RECAP

Concept	Key Takeaway
Model Selection	Choose based on cost, modality, latency, and customization needs.
Inference Parameters	Control model behavior with temperature, max length, and sampling.
RAG	Combine search + generation for more accurate, grounded answers.
Embedding Storage	Use services like OpenSearch or RDS to store vectorized data.
Customization Approaches	Balance cost vs. control: use fine-tuning or RAG as needed.
Agents	Extend GenAI to perform multi-step tasks with reasoning and API calls.

📚 Suggested Resources

- Amazon Bedrock Documentation
- RAG on AWS – Blog
- pgvector GitHub
- AWS re:Invent 2023 sessions on GenAI (especially ones by Bedrock and Q teams)