## **Path 1: RAG Application Builder - Detailed Journey**

### **Step 1: Environment Access**

**What the AI Engineer Does:**

* Logs into **SageMaker Domain Studio**
* Opens either:
  + **Jupyter Hub** (notebook interface), OR
  + **VSCode extension** integrated with SageMaker Domain Studio

**What They Get:**

* Pre-configured environment with SageMaker instance
* Connectivity to enterprise artifactory for downloading approved libraries
* Access to Bedrock models via application inference profiles tied to their SageMaker role

### **Step 2: S3 Storage Setup**

**What the AI Engineer Does:**

* Runs **S3 provisioning utility script** for S3 provisioning (Amandeep is working on this)

**What the Script Does (Two Options):**

1. **Provisions a NEW S3 bucket** for their application, OR
2. **Creates a folder within an existing Test S3 Bucket**

**Special Case:**

* If the engineer wants to use an **existing Line-of-Business (LOB) bucket**, they submit a request
* Your team works with them to get access to that specific LOB bucket

### **Step 3: Upload Documents for Testing**

**What the AI Engineer Does:**

* Uploads documents/data to the S3 bucket or folder (critical step for RAG testing)
* This is their test dataset they'll use to build the RAG application

**Why This Matters:**

* Cannot skip this step - RAG needs source documents to index and retrieve from

### **Step 4: OpenSearch Provisioning**

**What the AI Engineer Does:**

* Runs **OpenSearch serverless provisioning utility** for OpenSearch serverless provisioning (Amandeep is recommending to use a preferred config settings which cannot be provisioned and is not straightforward)

**What the Script Does:**

* Provisions OpenSearch serverless instance for vector storage and retrieval

### **Step 5: Build RAG Application**

**What the AI Engineer Does:**

* Writes RAG script using their preferred framework
  + **LangChain** is an example, but NOT limited to it
  + Can use any approved library from enterprise artifactory
  + Examples: LlamaIndex, Langchain etc.,
* Connects to:
  + S3 bucket/folder (document source)
  + OpenSearch (vector store)
  + Bedrock models (via inference profiles)

**What They're Building:**

* Document ingestion pipeline
* Embedding generation
* Vector storage in OpenSearch
* Retrieval and generation logic

### **Step 6: Local Testing Within SageMaker**

**What the AI Engineer Does:**

* Runs the RAG application **locally within SageMaker instance**
* Tests retrieval quality, response accuracy, performance
* Iterates on the code until satisfied and its all local to SageMaker instance

**What This Achieves:**

* Proves the RAG application works in Builder with No CI/CD
* No deployment needed for basic testing and validation

### **Step 7: Deployment Scenarios (Stretch Goal)**

**When This Is Needed:**

* Engineer needs to **schedule** the RAG application (e.g., daily document ingestion)
* Needs to expose it as a **service** (API endpoint, batch processing)
* Wants it running **outside** their SageMaker session

**What the AI Engineer Does:**

* Runs **ECS utility script** for ECS deployment

**What the Script Does:**

* Packages their code into a Docker container
* Pushes container to ECR (Elastic Container Registry)
* Deploys the container as an ECS service
* Provides endpoint or schedule configuration

**Result:**

* RAG application now runs as a production service
* Can be scheduled, scaled, monitored
* No longer dependent on their SageMaker session being active

## **Our Team's Deliverables for Path 1**

**Core (Must Have by October 31st):**

* S3 provisioning utility script (new bucket OR folder in Test bucket)
* OpenSearch serverless provisioning utility script
* Bedrock access configuration guide
* Document upload instructions and examples