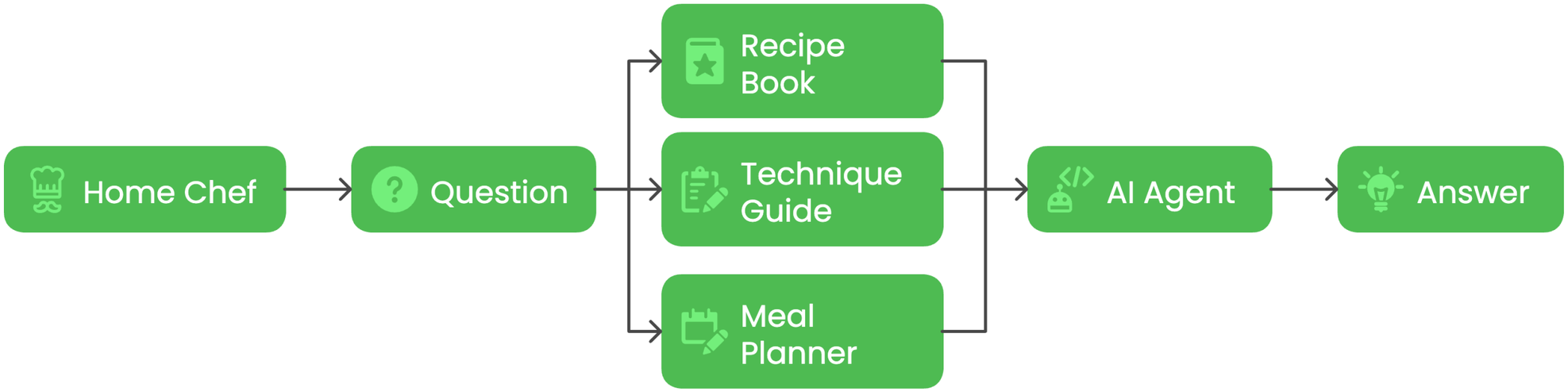
RAG Refresher

AI AGENTS WITH HUGGING FACE SMOLAGENTS

Adel Nehme
VP of AI Curriculum, DataCamp

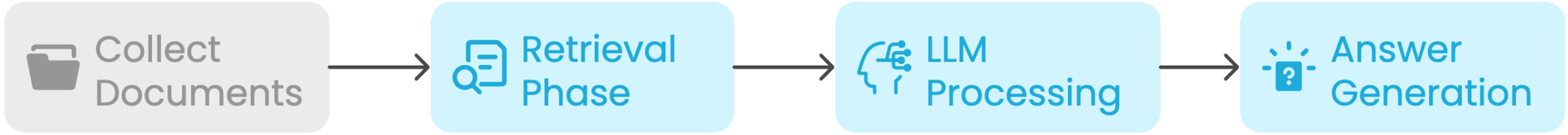
datacamp

Example: A Smart Cooking Assistant



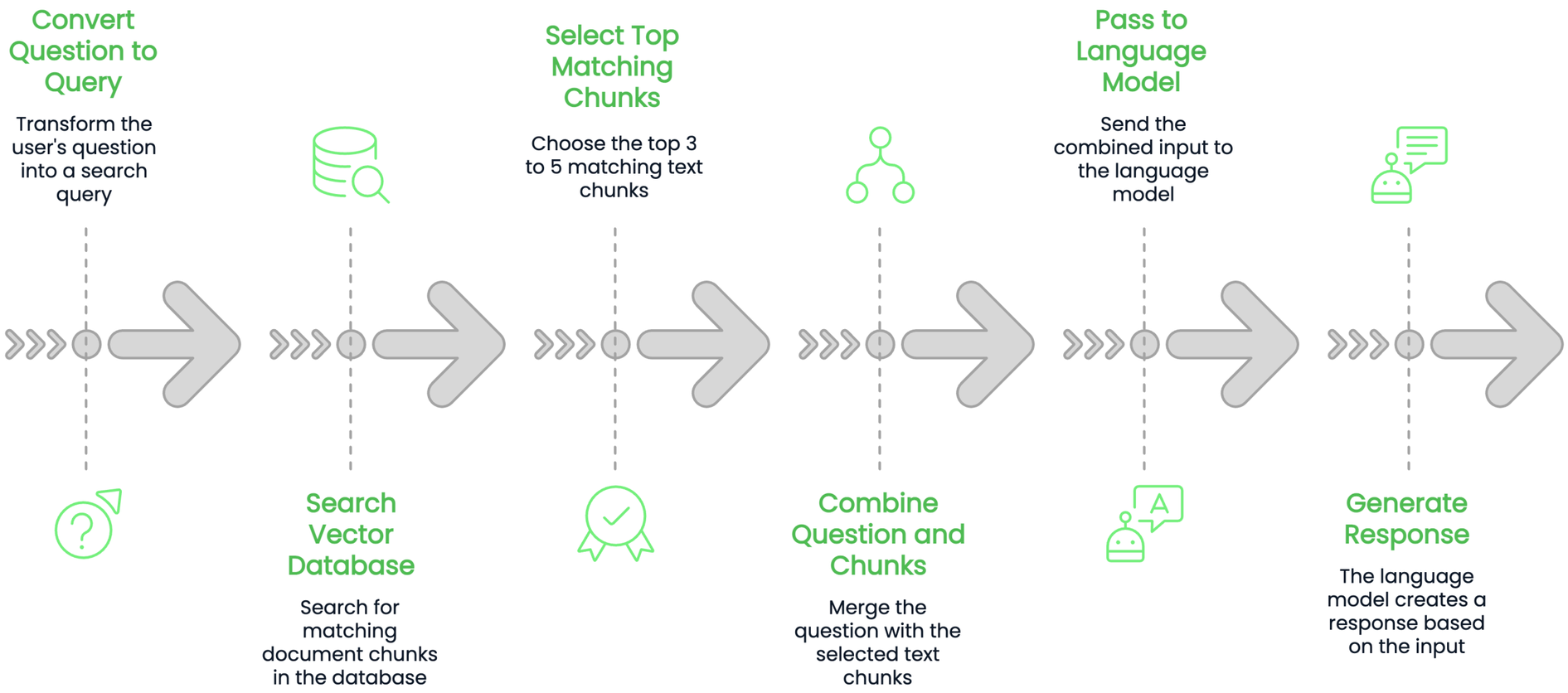
- Needs access to culinary knowledge
- Info is scattered across recipes, guides, and planning resources
- Must search and extract only the most relevant details

What is Retrieval Augmented Generation (RAG)?



RAG = Combine information retrieval with LLM generation

The RAG Workflow



Loading and Splitting Documents

```
from langchain_community.document_loaders import PyPDFDirectoryLoader
from langchain_text_splitters import RecursiveCharacterTextSplitter

# Load documentation from directory
loader = PyPDFDirectoryLoader("cooking_docs", mode="single")
documents = loader.load()

# Split into chunks
splitter = RecursiveCharacterTextSplitter(
    chunk_size=1000,
    chunk_overlap=200
)
chunks = splitter.split_documents(documents)
```

Creating a Vector Store

```
from langchain_huggingface import HuggingFaceEndpointEmbeddings
from langchain_community.vectorstores import FAISS

# Create embeddings and vector store
embedder = HuggingFaceEndpointEmbeddings(
    model="BAAI/bge-base-en-v1.5",
    task="feature-extraction",
)
vector_store = FAISS.from_documents(chunks, embedder)
```

Querying the Vector Store

```
query = "How do I cook salmon with herbs?"

# Similarity search
relevant_docs = vector_store.similarity_search(query, k=3)

# Create a context string
context = "\n\n".join(doc.page_content for doc in relevant_docs)
```

Query: How do I cook salmon with herbs?

Top 2 retrieved chunks (semantic matches):

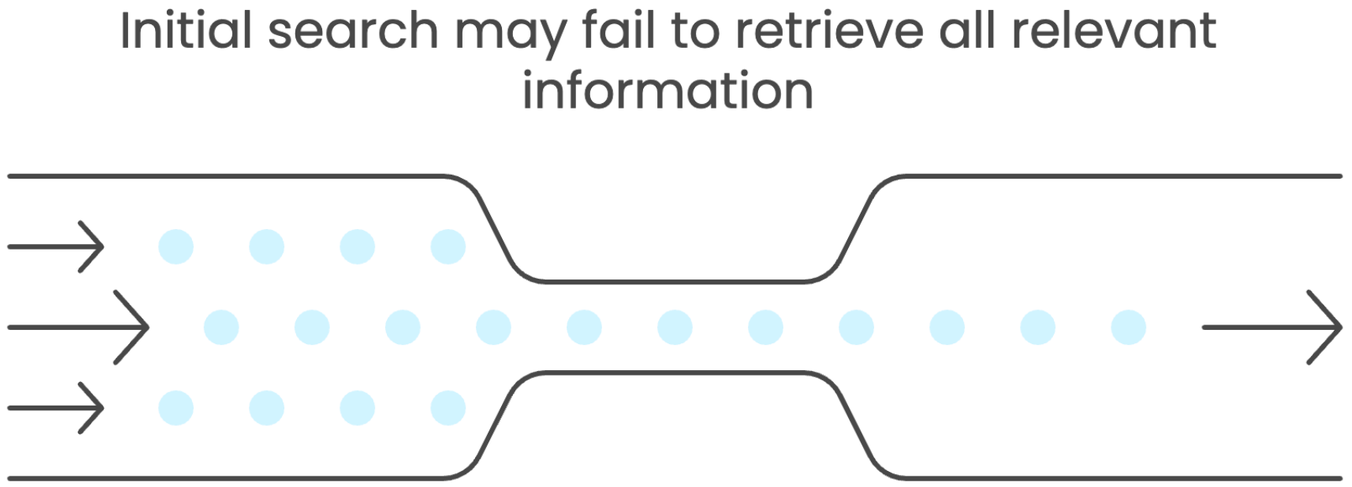
[1] *Salmon preparation basics* (p. 2) Pat the salmon dry to ensure browning. Season generously with salt, pepper, and fresh dill or parsley. Let the fillets rest 10 minutes so the salt penetrates. For even cooking, bring to room temperature...

[2] *Baking salmon in the oven* (p. 5) Preheat oven to 200°C (392°F). Place fillets on a parchment-lined tray, top with lemon slices and herb butter (dill/parsley). Bake 10 to 12 minutes until just opaque and flaky; rest 2 minutes before serving...

Traditional RAG Pipeline Limitations

> *How do I plan a week of meals under $50 while meeting all nutritional requirements?*

Answer is spread across docs (budget, nutrition, techniques, recipes).



Let's practice!

AI AGENTS WITH HUGGING FACE SMOLAGENTS

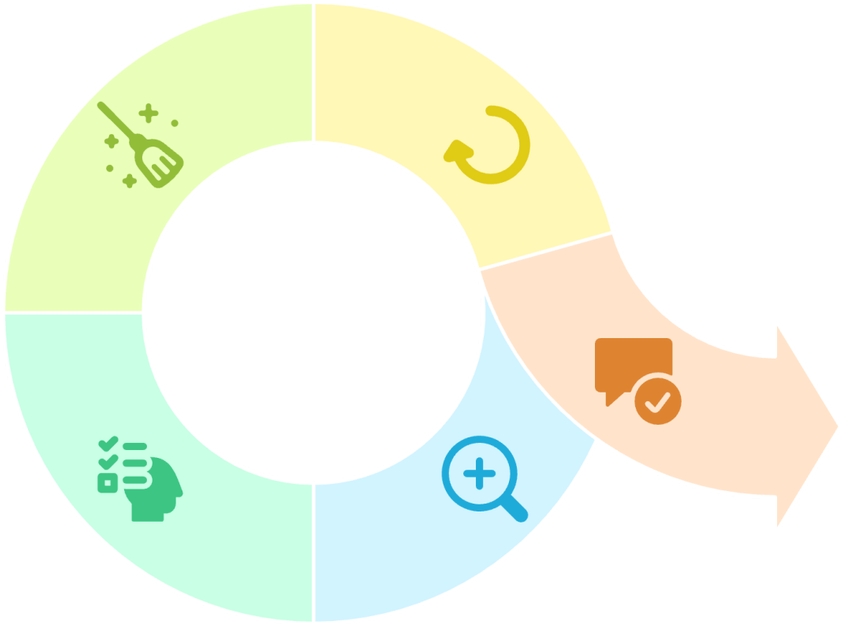
Introduction to Agentic RAG

AI AGENTS WITH HUGGING FACE SMOLAGENTS

Adel Nehme
VP of AI Curriculum, DataCamp

datacamp

Agentic RAG: Iterative Retrieval + Reasoning



1	2	3	4	5
Retrieve	**Evaluate**	**Refine**	**Repeat**	**Answer**
Initial search for information	Assess relevance and gaps	Adjust query for better results	Continue until sufficient evidence	Provide comprehensive response

Stateless vs. Stateful Tools in smolagents

Function Tools (`@tool` decorator)

- Stateless: no memory between calls
- Can't remember your vector store between calls

Class-Based Tools (`Tool` subclass)

- Stateful: keep references across calls
- Store complex objects

The Anatomy of a Class-Based Tool

```python
from smolagents import Tool

class ToolName(Tool):
    name = "tool_name"
    description = "Clear description for the agent"
    inputs = {
        "parameter_name": {"type": "parameter_type", "description": "Parameter purpose"}
    }
    output_type = "string"
    def __init__(self, custom_parameters):
        super().__init__()
        self.custom_attribute = custom_parameters
    def forward(self, parameter_name):
        # Agent calls this method
        return "processed output"
```

Build a Recipe Search Tool

```
class RecipeSearchTool(Tool):
    name = "recipe_search"
    description = "Search cooking documentation for recipes, techniques, and meal planning informat
    inputs = {
        "query": {"type": "string", "description": "Natural language cooking query"}
    }
    output_type = "string"
    def __init__(self, vectorstore, k=6):
        super().__init__()
        self.vectorstore = vectorstore
        self.k = k
    def forward(self, query):
        docs = self.vectorstore.similarity_search(query, k=self.k)
        return "\n\n".join(doc.page_content for doc in docs) or "Nothing found."
```

Cooking Assistant Agent: Putting It All Together

```
# Initialize the retrieval tool
recipe_search = RecipeSearchTool(vector_store)

agent = CodeAgent(
    tools=[recipe_search],
    model=model,
    instructions="Search thoroughly to provide complete recipe answers.
      If initial results seem incomplete, try different search terms.",
    verbosity_level=1,
    max_steps=8
)
```

Agent Run Example

> *Query: How do I cook salmon with herbs using professional techniques?*

[Step 1] Search "salmon herbs cooking techniques"...

[Step 2] Search "professional salmon preparation"...

[Final Answer]

"Pat fillets dry, season with herbs, and:

- *Pan-sear skin-side down, basting with butter, garlic, thyme*
- *Or bake at 200°C with lemon & herb butter for 10 to 12 min"*

Let's practice!

AI AGENTS WITH HUGGING FACE SMOLAGENTS

Working With Multi-Step Agents

AI AGENTS WITH HUGGING FACE SMOLAGENTS

Adel Nehme
VP of AI Curriculum, DataCamp

datacamp

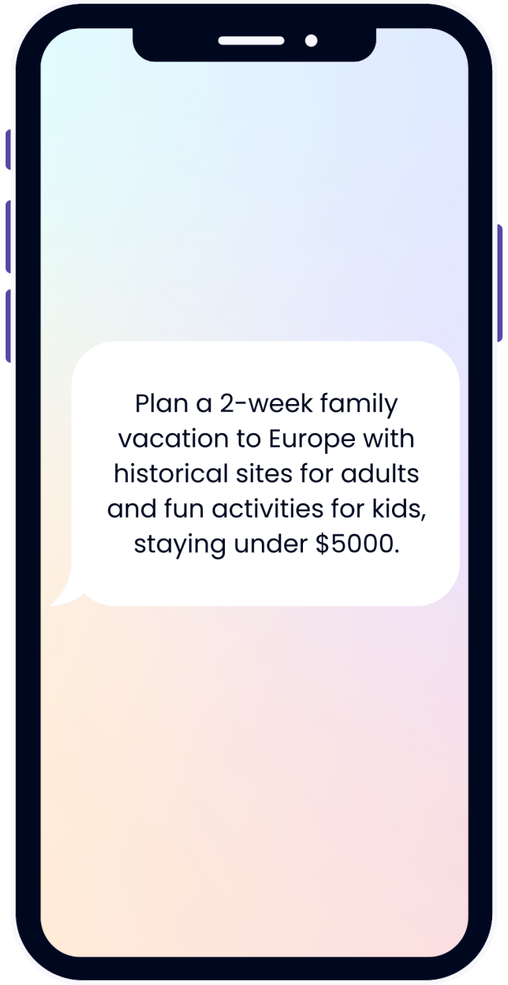
What is a multi-step agent?

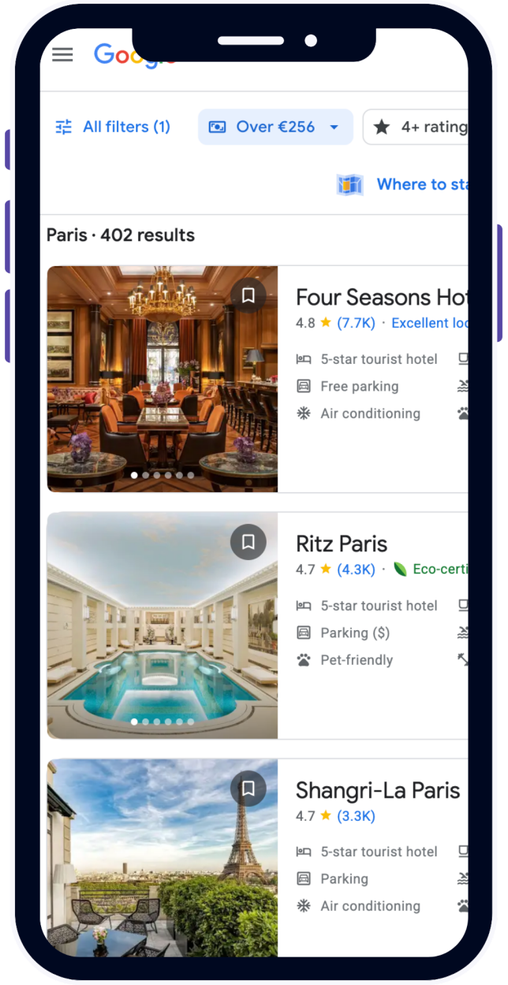
It's a system like the agentic RAG you built, where tasks are completed in multiple steps.

What challenges arise when deploying these systems?

Many, but let's start with the first one and how smolagents can help.

Example: Travel Assistant





Planning Intervals: Helping Agents Rethink

```
agent = CodeAgent(
    tools=[document_search_tool],
    model=model,
    planning_interval=3,
    max_steps=12
)
```

- `planning_interval=3` : Agent pauses after every 3 steps.

Agent Run with Planning Intervals

> *"Plan a 2-week family vacation to Europe with historical sites for adults and fun activities for kids, staying under $5000."*

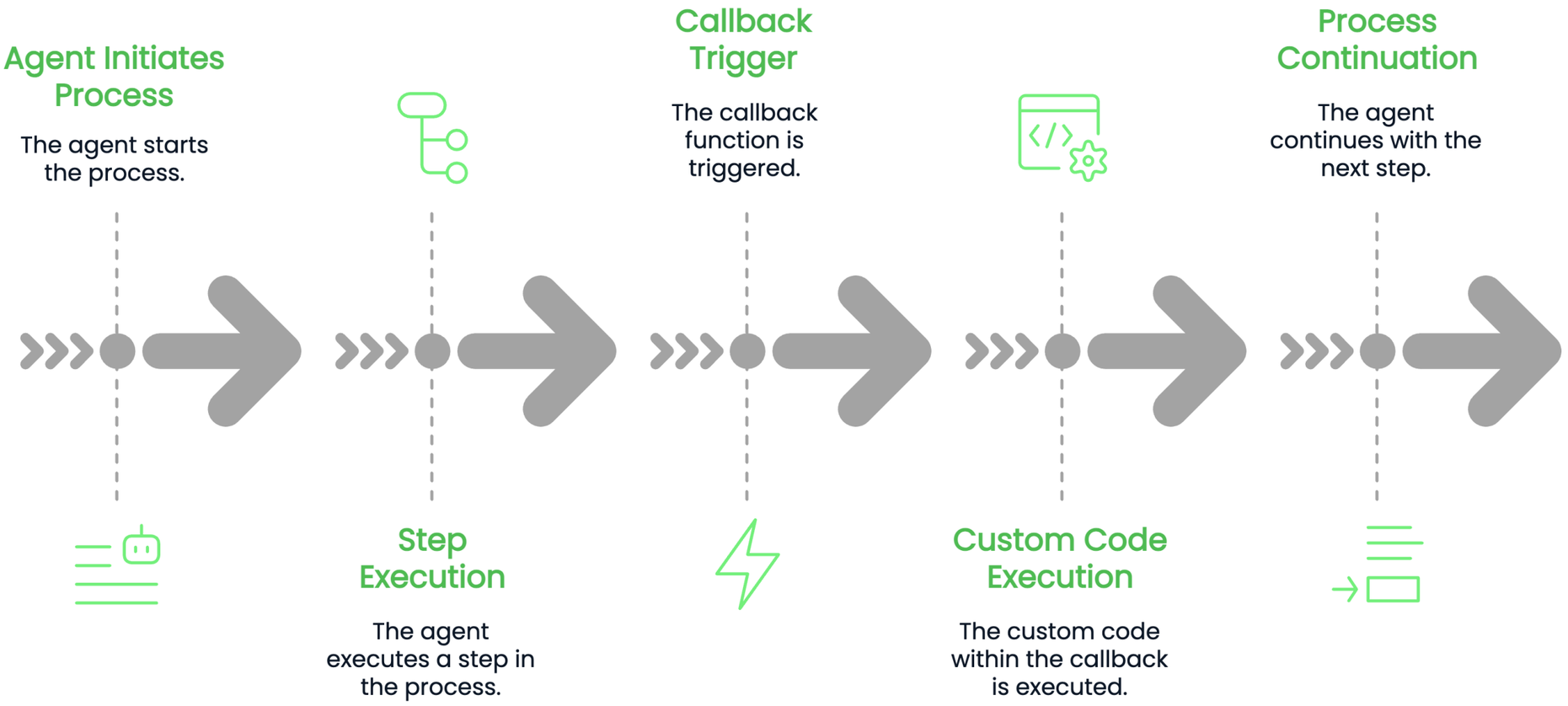
[Step 1] Search "Paris hotels" -> Found luxury hotels (~$4000 total)

[Step 2] Search "Paris attractions for families" -> Found Eiffel Tower, Louvre, theme park tickets

[Step 3] Pause and rethink (planning interval) Hotels are too expensive -> budget blown. I should look for cheaper options + kid-friendly activities.

[Step 4] Search "affordable family hotels in Europe" -> Found mid-range options in multiple cities

Callbacks: Hooks Into the Agent's Process



Basic Callback Function

```
def callback_function(agent_step, agent):
    # Do something with the agent step or the agent itself
    pass
```

- `agent_step` : details about the step (plan, step number, etc.)
- `agent` : the full agent object (state + methods)

Planning Step Callbacks

```python
def planning_callback(agent_step, agent):
    print("AGENT PLANNING")
    print("=" * 50)
    print(agent_step.plan[:300])
    if len(agent_step.plan) > 300:
        print("\n... (plan truncated)")
    print("=" * 50)
```

```
AGENT PLANNING
==================================================
Search affordable hotels + kid activities
Then create balanced itinerary...
... (plan truncated)
==================================================
```

Action Step Callbacks

```python
def action_callback(agent_step, agent):
    step_num = agent_step.step_number
    print(f"Step {step_num}: Taking action")

    if agent_step.is_final_answer:
        total_tokens = agent_step.token_usage.total_tokens
        print(f"Total tokens used: {total_tokens}")
```

```
Step 2: Taking action!
Step 3: Taking action!
Step 4: Taking action!
Total tokens used: 4,218
```

Adding Callbacks to Agents

```
from smolagents import ActionStep, PlanningStep


agent = CodeAgent(
    tools=[document_search_tool],
    model=model,
    step_callbacks={PlanningStep: planning_callback, ActionStep: action_callback}
)
```

What Can You Do with Callbacks?

- Log searches to understand what users look for most
- Add human-approval checkpoints
- Send progress updates to dashboards or apps
- Adjust agent behavior mid-run based on performance
- And more...

Let's practice!

AI AGENTS WITH HUGGING FACE SMOLAGENTS