Lesson 11 · AWS Bedrock Agents

PDF: this lesson · **Install (Linux · macOS · Windows):** guide · PDF

> **Part of local-ai-lab** - a hands-on course for building local AI.

> **Course home:** https://nikolareljin.github.io/local-ai-lab/ **Source:** https://github.com/nikolareljin/local-ai-lab

> **Lessons:** 1 · RAG → 2 · MCP → 3 · Hybrid retrieval → 4 · RAG safety → 5 · RAG evaluation → 6 · Repo assistant → 7 · LangChain → 8 · LangGraph → 9 · Ollama tools → 10 · Semantic Kernel → **11 · Bedrock Agents (you are here)** → 12 · Google ADK → ... → 15 · Docs from changes

> **Status: planned.** Outline below; full published slideshow lesson + step-by-step coming later. This is a **managed cloud** lesson - you build and drive it from a **local** dev environment (AWS CLI/SDK) against Amazon Bedrock. Star the repo to follow along.

The idea

You built RAG and agents by hand. **Amazon Bedrock Agents** is the **managed** version: a hosted agent that combines a **knowledge base** (managed RAG over your documents) with **action groups** (tools backed by Lambda). This lesson shows how the primitives you wrote map onto a cloud agent platform - and where the trade-offs are (less control, less ops).

What you'll learn

- **Bedrock model access** - enabling foundation models (Claude, Titan, Llama) in your account
- **Knowledge Bases** - managed ingestion + embeddings + vector store over your `documents/` (this is Lesson 1's pipeline, as a service)
- **Action groups** - defining tools with an OpenAPI schema, implemented by a Lambda (the cloud equivalent of Lesson 5's `search_docs`)
- **Agent orchestration** - how Bedrock plans, retrieves, and calls actions; reading the trace
- **Driving it locally** - invoking the agent from your machine with `boto3` / AWS CLI

The design we'll build

```
import boto3

agent = boto3.client("bedrock-agent-runtime")   # run locally, talks to AWS

resp = agent.invoke_agent(
   agentId="XXXX", agentAliasId="YYYY",
   sessionId="demo-1",
   inputText="How do I reset the device? Cite the manual.",
)
# stream the completion chunks; the agent transparently used the Knowledge Base
# (managed RAG) and any action groups, then grounded its answer in your docs.
```

> **The mapping:** Knowledge Base = Lesson 1 retrieval (managed). Action group + Lambda = Lesson 9 tool calling (managed). Agent = Lesson 8's orchestration (managed). Same ideas, someone else runs the servers - you trade transparency and cost for less ops.

Builds on

Concept	From	Cloud equivalent
extract → chunk → embed → retrieve	Lesson 1	Bedrock Knowledge Base
a tool the agent can call	Lesson 9	Action group + Lambda
orchestration / control flow	Lesson 8	Bedrock Agent runtime

Prerequisites

An AWS account with Bedrock model access, the AWS CLI configured locally, and `boto3`. Lessons 1 and 9 give the mental model.

Next lesson

Lesson 12 · Google AI Development Kit →
