knowit

A beginners guide to Agentic AI in AWS

Martin Aamodt

knowit

Welcome

- Martin Aamodt
  - Principal Cloud Architect @ Knowit

knowit

It doesn't take much to get started

- Be wary of the demo ghost
- Nothing here is production ready, and that's deliberate
- The goal is to get you started
- We all use agents. Few of us have built our own
- Agents reach problems that are hard to solve deterministically
  - no fixed set of if-statements covers every way a network path can break
- Getting started takes about 40 lines of Python
- The blog post behind this talk is a year old, and that part hasn't changed

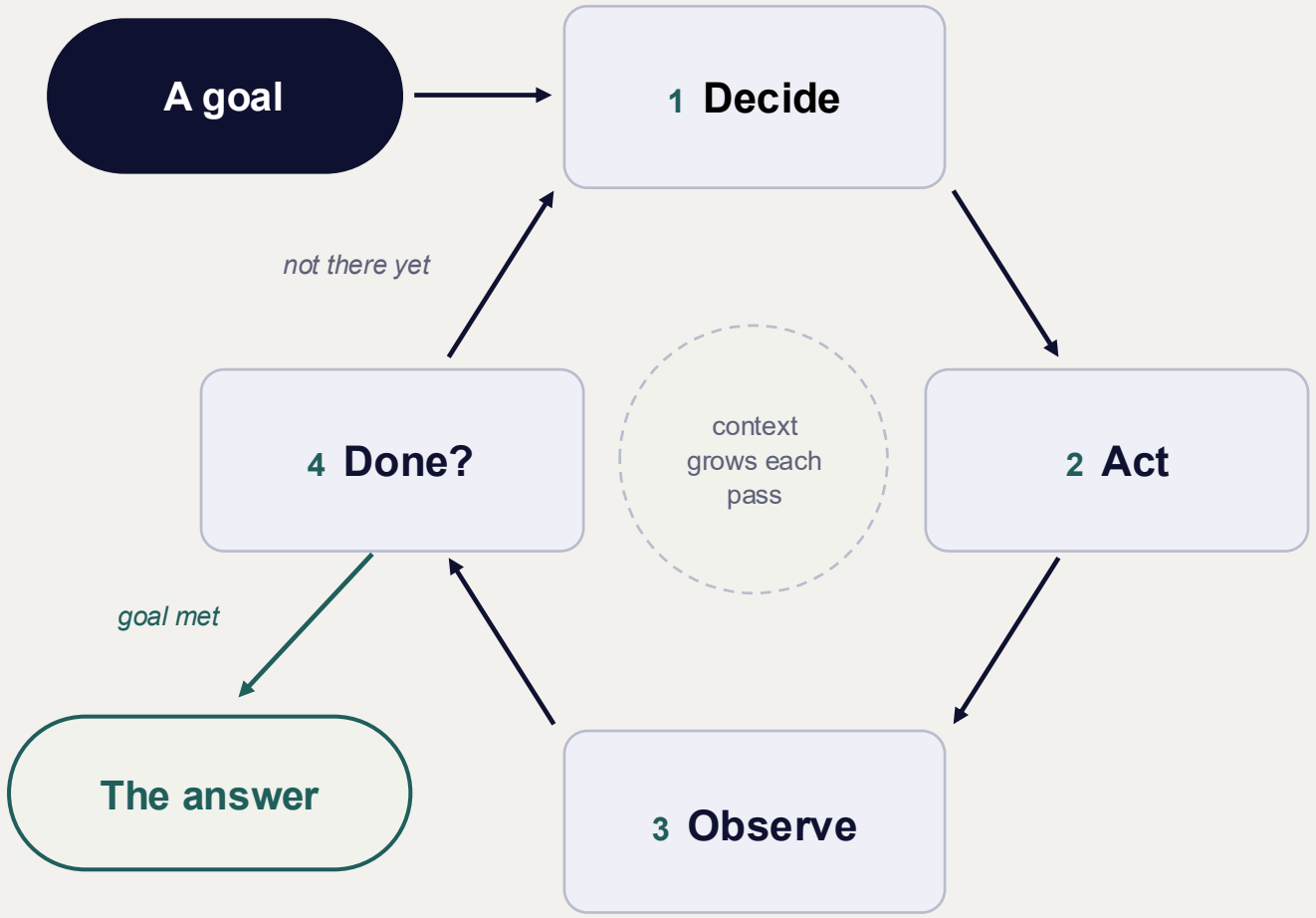
knowit

What is an agent?

- A model, a handful of tools you wrote, and a loop around them
- The tools are ordinary Python functions sitting in your code
- The model chooses which tool to call, and when
- It keeps calling tools until it can answer the question
- We'll be using AWS Strands for the loop, and building our own tools

knowit

The Agentic Loop



The model decides what happens next and when to stop.

knowit

Where this started

- 2 AM. ECS tasks can't connect to Aurora
- So you start the usual dance:
  - security groups, both ends
  - route tables
  - transit gateway, its attachments and routes
  - the VPN
- Every one is a different console page, and by the end you've lost the big picture

knowit

My first agent

- First time ever with AI agents or Strands
- 3 to 4 hours, to a working agent
- 30+ tools: VPC, route tables, security groups, NACLs, endpoints
- It traced the path and named the blocking component

knowit

Agentic AI on AWS today

- Amazon Q in the console: ask why a resource is misbehaving, grounded in your account.
- Operational agents: investigate incidents and propose a root cause
  - DevOps, FinOps, Security Agent, Kiro
- Bedrock AgentCore: runtime, memory, gateway and identity when you take an agent to production
- Strands Agents: the open source SDK for building your own

knowit

So why build your own

- The console agent knows AWS. Your agent knows AWS and you
- Things no console assistant has read:
  - your tagging policy and its exceptions
  - which regions you are allowed to deploy in
  - your runbook, and which steps are out of date
  - which services on the bill anyone still cares about
- Each of those is a file, an API call, or a paragraph of prose

knowit

Strands -> Getting started

```
$ pip install strands-agents strands-agents-tools


from strands import Agent


agent = Agent()
agent("Explain a NAT gateway to my manager")
```

- Python 3.10 or newer
- AWS credentials you already have
- Bedrock model access, granted per region and per model
- No account to create, no server to run

knowit

A tool is a function with a docstring

```
from strands import Agent, tool

@tool
def services_in_use() -> dict:
    """Which AWS services this account spent
    money on in the last 30 days."""
    return cost_explorer_by_service()

agent = Agent(tools=[services_in_use])
```

- Strands reads the name, the type hints and the docstring
- That becomes the schema the model sees
- It is the only thing telling the model when to call this
- This is the tool from tonight's demo

knowit

The system prompt

```
# one paragraph, in your own words
SYSTEM = """..."""

agent = Agent(
    system_prompt=SYSTEM,
    tools=[services_in_use],
)
```

- The third thing you hand Agent, alongside the model and the tools
- Plain prose: who is asking, what they care about, what to ignore
- Our regions, our constraints, how long an answer should be
- This is where the judgement lives
- Editing it costs one sentence and no redeploy

knowit

Demo: AWS announcements that affect us

- AWS ships around 40 announcements a week, none of them written for you
- I read all new announcements every morning through an rss feed
- Four steps, two tools, one edit to the prompt

knowit

The four steps

- 1. Agent() with no tools. Confident, useless answer
- 2. One tool: the AWS What's New feed for the last 7 days
- 3. Second tool: what this account actually spends money on
- 4. One paragraph of system prompt about who we are.

knowit

Swapping the model is one line

- Same tools, same prompt, different model
- Claude Sonnet and Amazon Nova Lite give different answers
- Fetching and formatting is easy work for a small model
- Deciding what matters to us is judgement
- The same line can point at OpenAI, Gemini or a local Ollama model

knowit

Tools gather data, AI provides reasoning

- A tool that filters, ranks and applies your rules in Python is logic you maintain forever
- Return raw data and the judgement lives in the prompt
- All 30 of my network tools were dumb: describe, list, get
- Changing my mind cost one sentence

knowit

Gotchas

- Ask the same question twice, get two different answers
- Its IAM role is your blast radius. Read-only first
- Anything your tools fetch is untrusted input
- You can't test this by eye. Once it matters, you need evals
- Read-only credentials and a laptop are enough to start

knowit

Start building

- pip install strands-agents and get an empty Agent to answer anything
- Take the most boring script you own and put @tool above it
- Write one paragraph about what your team cares about
- Point it at a question only your team has

knowit

Thank you!