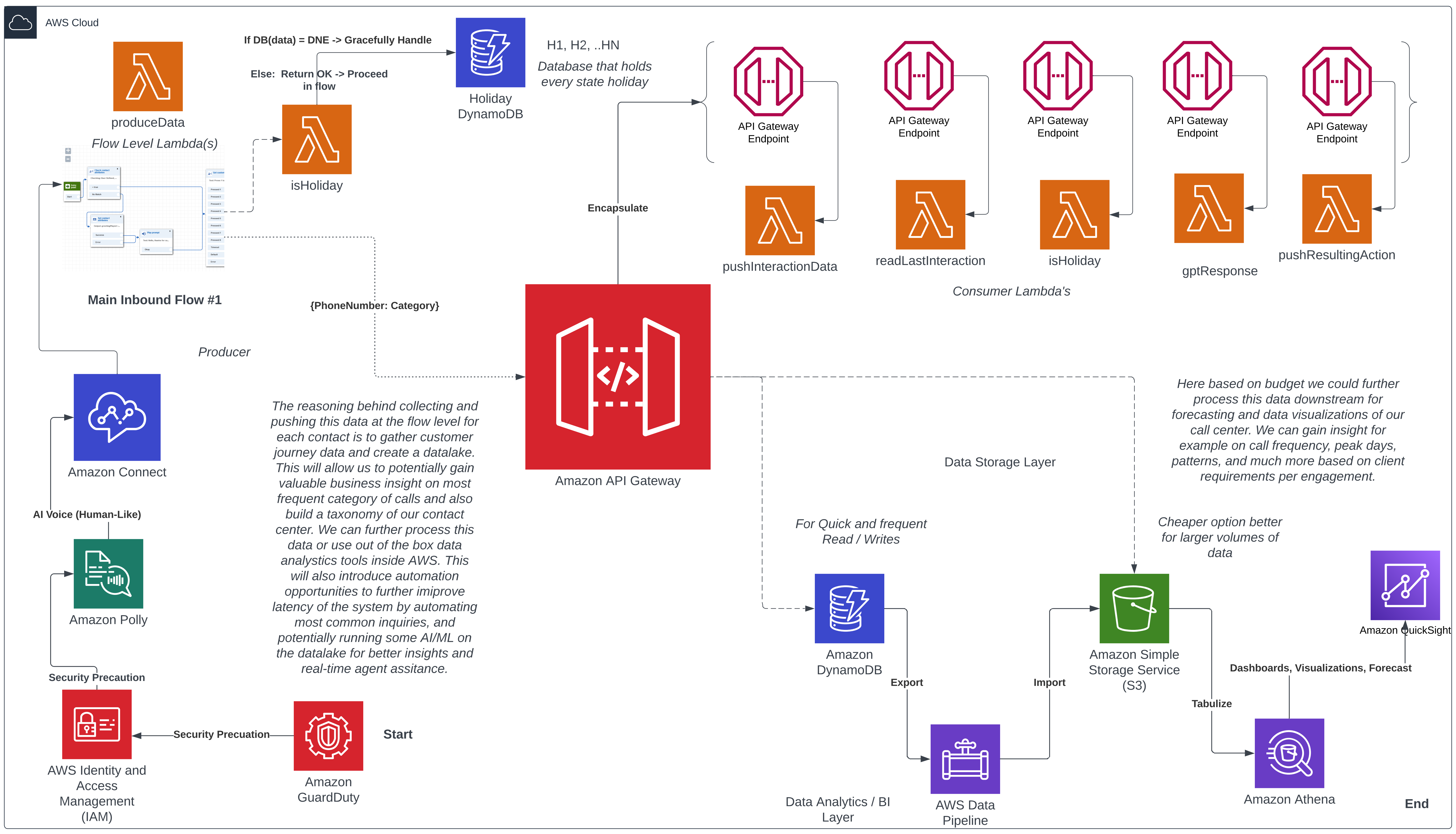
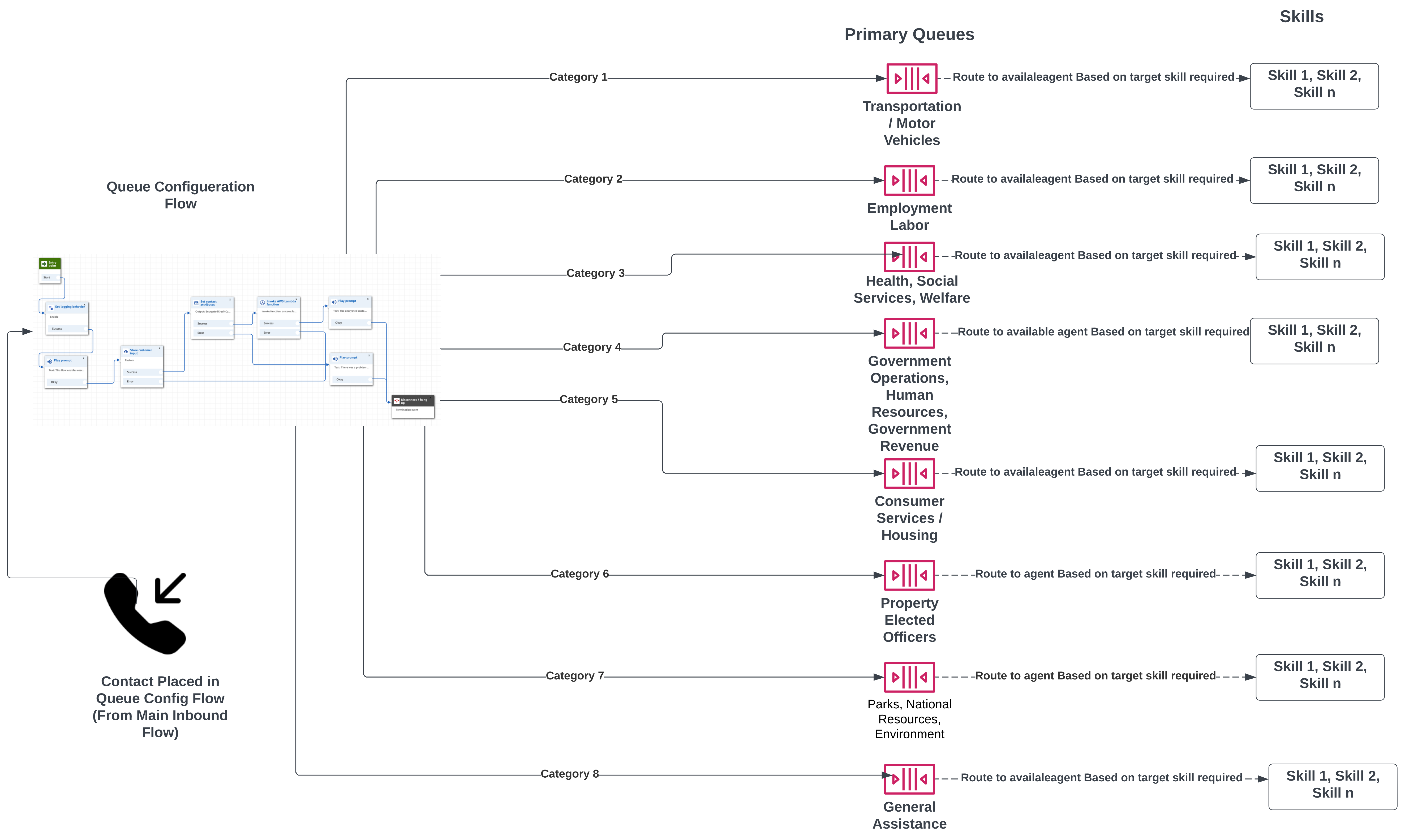
Client-Server (Server-less) Micro-service
AWS Architecture Design



The reasoning behind collecting and pushing this data at the flow level for each contact is to gather customer journey data and create a datalake. This will allow us to potentially gain valuable business insight on most frequent category of calls and also build a taxonomy of our contact center. We can further process this data or use out of the box data analytics tools inside AWS. This will also introduce automation opportunities to further imiprove latency of the system by automating most common inquiries, and potentially running some AI/ML on the datalake for better insights and real-time agent assitance.

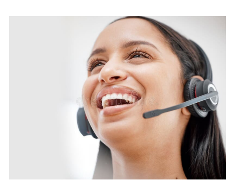
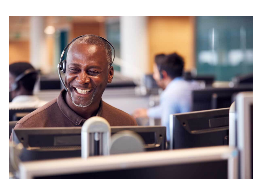
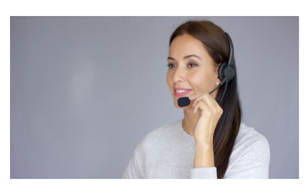
Here based on budget we could further process this data downstream for forecasting and data visualizations of our call center. We can gain insight for example on call frequency, peak days, patterns, and much more based on client requirements per engagement.

Agent Routing Connect Instance Level


Skills
Primary Queues
Queue Configueration Flow
Category 1
Route to availaleagent Based on target skill required
Transportation / Motor Vehicles
Skill 1, Skill 2, Skill n
Category 2
Route to availaleagent Based on target skill required
Employment Labor
Skill 1, Skill 2, Skill n
Category 3
Route to availaleagent Based on target skill required
Health, Social Services, Welfare
Skill 1, Skill 2, Skill n
Category 4
Route to available agent Based on target skill required
Government Operations, Human Resources, Government Revenue
Skill 1, Skill 2, Skill n
Category 5
Route to availaleagent Based on target skill required
Consumer Services / Housing
Skill 1, Skill 2, Skill n
Category 6
Route to agent Based on target skill required
Property Elected Officers
Skill 1, Skill 2, Skill n
Category 7
Route to agent Based on target skill required
Parks, National Resources, Environment
Skill 1, Skill 2, Skill n
Category 8
Route to availaleagent Based on target skill required
General Assistance
Skill 1, Skill 2, Skill n
Contact Placed in Queue Config Flow (From Main Inbound Flow)

I'd further define skills here based on the given set of human agents we could work with, and their skills. Then, we can implement some sort of logic to figure out a transfer logic so agents in one queue are busy. Some agents should be able to handle multiple queues, and associated skills will also help further define the subset of actions. Requirements gathering phase here can vary significantly hence why I chose to dive less into this and more into back-end processes / features in the limited time for this assignment. Routing strategy however is something that's VERY important in an actual client setting, and should be carefuly planned, thought out, and implemented/re-evaluteed frequently

Agent Hierarchy (Agent Level)

Let's assume we have 100 agents for simplicity. The most optimal way to design the hierachy / skillset of the agents from my perspective would depend on the existing capacity of our agents from a skillset perspective. It would be be ideal to analyze what most people call about, hence why I'd set up the data-lake creation API. From there, we can gain insight on what queues/skills are our bottlenecks (i.e what are category and task is associated with most calls). We should upskill most of our agents to address poential bottlenecks in queue routing, ultimtaely lowering queue wait times and improving "latency" of the average contact from call being made to ACW (after call work) being completed. To further ehance the system, training can be designed and carried out to upskill our agents and ultimately enable them to handle more queues and kinds of tasks. This also introduces automation capabilities as a next stage to cut some costs on most repetitive actions that can be emulated by a script encapsulated in a server-less Lambda function. The beauty of the cloud is as calls scale, the system and compute resources will also scale to meet the neccessary workload demand.