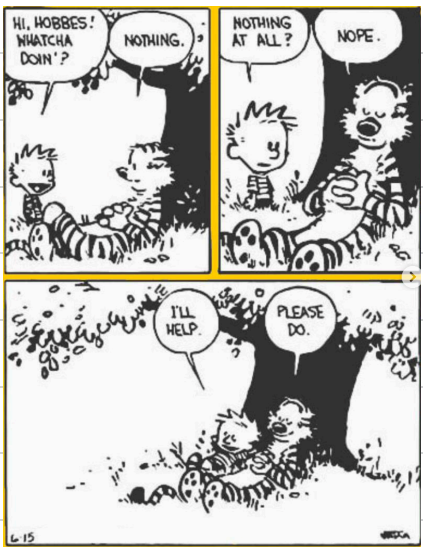
Hands On: MCPs, Multi modality, Observability

We will begin 8:40 PM



List of company issues — Fields
— Documentation.

90% of issues → solvable

6% → Engineers.

Agents ① → Understand issue → what tools can we use

② → Calling the tool

③ → Draft a reply → Solution

④ → Supervise

Q1. Why do we need multiple agents? !?
Q2. Do all agents need LLM? X
Q3. Do Agents that use LLM, should have similar intelligent LLM? X

You hire someone who has to do all these things:

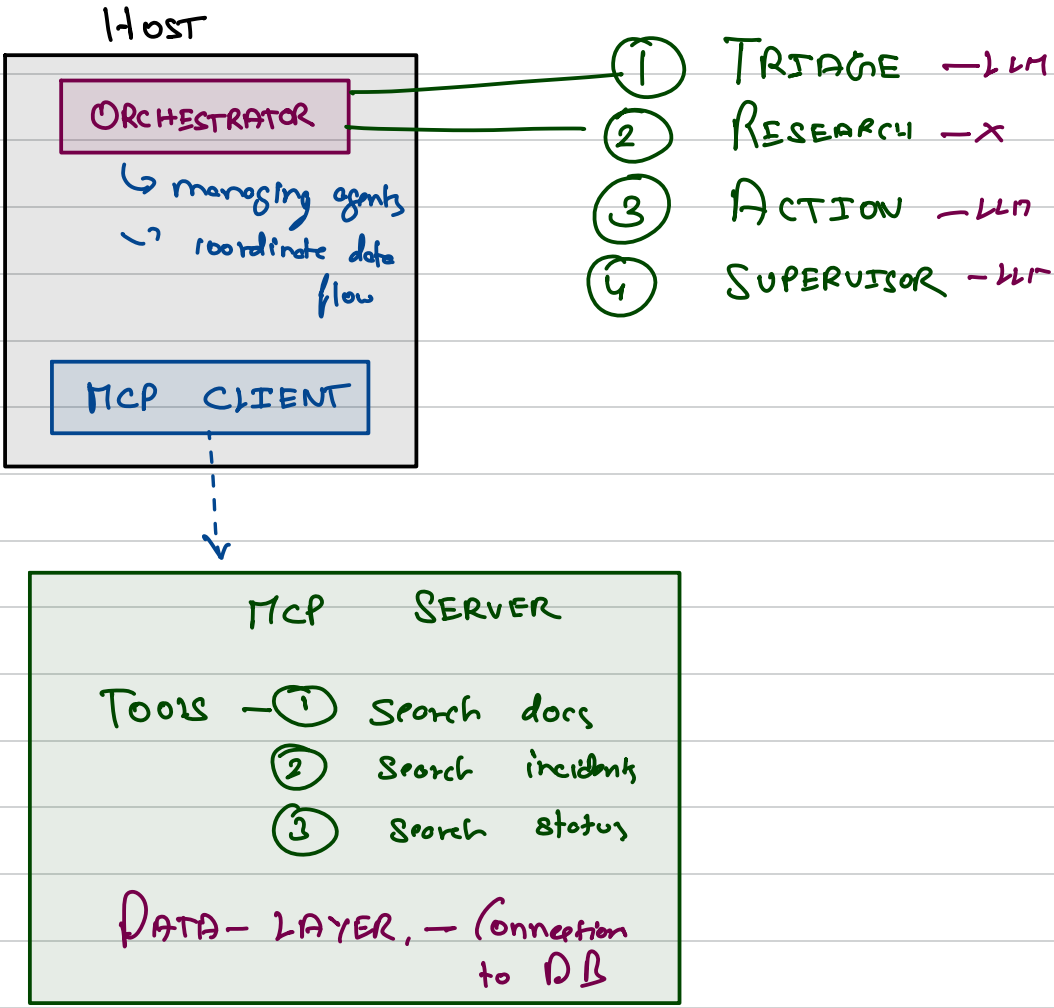
1. Understa issue
2. Look up document
3. check logs
4. draft reply
5. write notes (internal note)
6. check if fix makes sense?
7. Decide if we should take this to prod??

① Ticket → Incident_ID
title.
Status
summary
tags.
Severity ?
— Suggest Solution

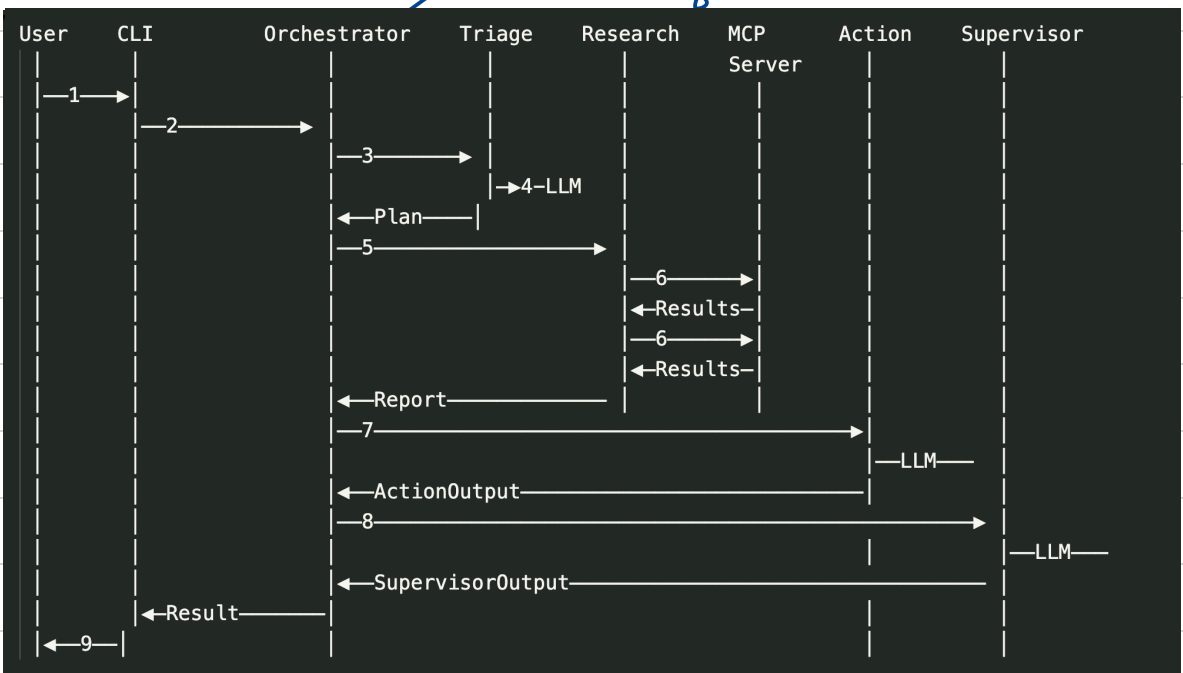
① Support Docs : runbooks / *.md

② Incident Database : Present / ongoing Incidents

③ Service Status : service-name : status
Healthy / Degrad
/ Dow.


Host
ORCHESTRATOR
managing agents
coordinate data flow
MCP CLIENT
1 TRIAGE — LLM
2 RESEARCH — X
3 ACTION — LLM
4 SUPERVISOR — LLM
MCP SERVER
TOOLS — 1 Search docs
2 Search incidents
3 Search status
DATA - LAYER, — Connection to DB

List of tools & arguments



Steps:

1. User submits ticket via CLI
2. CLI calls orchestrator.run_ticket_flow()
3. Orchestrator invokes TriageAgent
4. Triage calls LLM to classify ticket
5. Orchestrator invokes ResearchAgent with plan
6. Research calls MCP tools (multiple times)
7. Orchestrator invokes ActionAgent
8. Orchestrator invokes SupervisorAgent
9. CLI displays final output to user

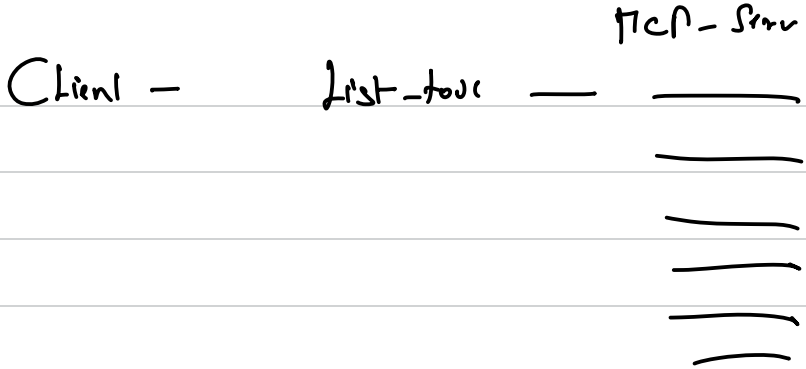
app = Server( )

1. @app.list_tools
   - Function_name
   - schema — Json
   - for ∀ tools

2. @app.call_tool
   - call the function.
   - ∀ the tool

Observability

→ trace — Log — Agent typ
- query
- respon


MCP - Serv
Client -
list_tool