GENAI DAY / MAY 14, 2026 / CLOSING KEYNOTE

Agentic AI isn't magic. It's a stack.

A field guide to the six layers powering every agent you'll build.

What I wish I'd known two years ago. / What changed in the last six months. / Where you start tomorrow.

Zaki Medina

Co-Founder, COO & CTO / REVIVE Healthcare Group

Quick check.

Where are you on your agent journey?

01

Curious

Heard the word. Not sure what's hype.

02

Experimenting

Wired up your first agent in a notebook.

03

Building

Trying to ship something real to a team.

04

Scaling

Now managing 5, 10, 20+ agents in production.

Today is for everyone at stage 1, 2, or 3 — and a sanity check for stage 4.

We've seen this pattern twice.
This is the third shift.

2006–2015

On-prem → Cloud

EC2. S3. Lambda. New primitives, new winners.

2012–2016

Apps → APIs

Microservices. API-first. New stack literacy required.

2024–2027

Humans → Agents

LLMs become the primary user of software. We're here.

YOU ARE HERE

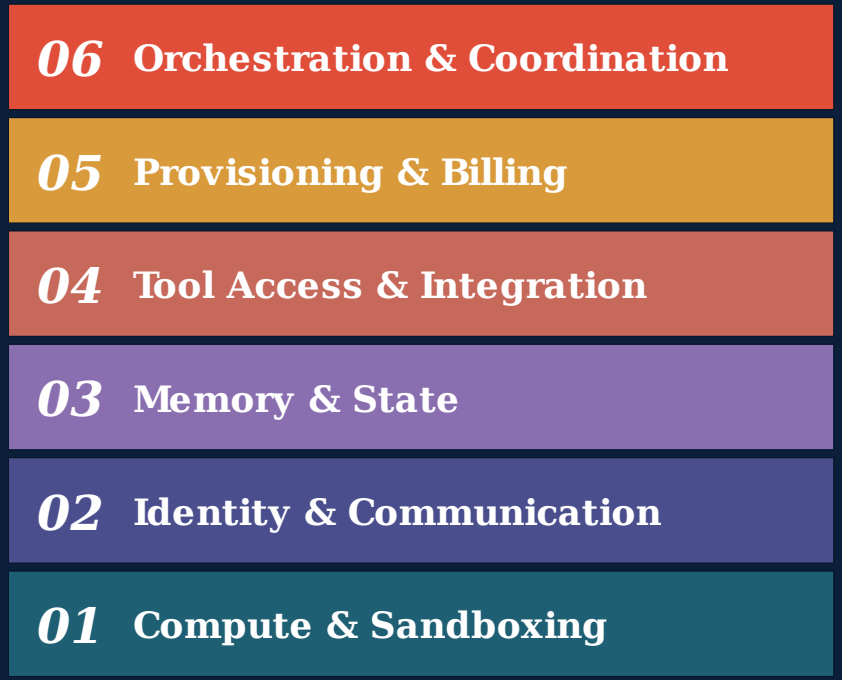
Six layers under every agent you'll build.

Think system calls, not Lego.

Every agent you build sits on top of these six primitives — the same way every app you've ever written sits on top of memory, file I/O, and the network.

Some layers are load-bearing walls. Some are shims that won't last 18 months. One — the most important one — barely exists yet.

Framework adapted from Nate Bjones, 2026.



THE FOUNDATIONS

Layers 1–3: where the agent lives, thinks, and remembers.

01 Compute & Sandboxing

Where your agent's code actually runs. Isolated, auditable, disposable. (E2B, Daytona, Modal.)

LOAD-BEARING

02 Identity & Communication

How your agent proves who it is and talks to other systems. Today: email. Tomorrow: native protocols.

TRANSITIONAL

03 Memory & State

What your agent remembers across sessions, days, conversations. Not "save the chat" — active curation.

EARLY

THE WORKFLOW

Layers 4–6: how the agent gets work done — and where it breaks.

04 Tool Access & Integration

How your agent reaches into the SaaS your team already pays for. Slack, Jira, Salesforce, GitHub. (Composio.)

GROWING

05 Provisioning & Billing

How an agent buys things — provisions a database, pays a vendor — without your card details leaking.

EMERGING

06 Orchestration & Coordination

How 50 agents work together reliably with audit, fallback, and cost controls. The biggest gap in the stack.

MOSTLY MISSING

THE GAP

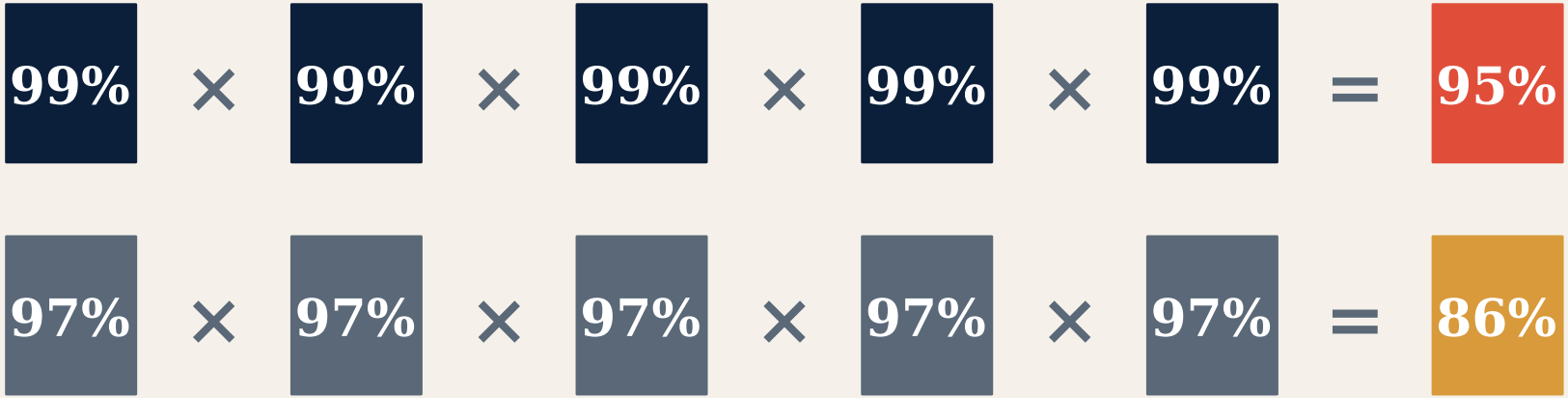
Orchestration is the next Kubernetes-shaped opportunity.

- **Scheduling & lifecycle**
- **Merge & conflict resolution**
- **Supervision hierarchies**
- **Financial observability**
- **Standard failure modes**

All five missing from the stack today. Add up to billions in value when somebody builds them.

> ***If you're early in your career and looking for where to bet — this is the layer.***

Reliability compounds — *in the wrong direction.*



Five dependencies at 99% uptime ≠ a 99% system. The microservices fragility problem — except agents are non-deterministic. Design for compounding failure.

OUR STORY, SHORT VERSION

We've been at this almost two years.

MID 2024

We started.

Built the first prediction agent. Believed our own hype.

MID 2025

We struggled.

Tool-calling reliability. Memory. Governance gaps. Real costs.

LAST 6 MO

Things clicked.

Foundation models matured. Stack patterns stabilized. We shipped.

> ***The real breakthroughs are happening right now. You are not late.***

Three things flipped — *and the work got fun again.*

01

Foundation-model tool-calling got reliable.

Mid-cap models now hit tools deterministically enough to be the backbone of a production agent — not just a demo.

02

The stack patterns finally crystallized.

Predict → Source → Fulfill → Learn. Agents propose, systems decide. Overlay, don't replace. Battle-tested patterns now, not guesses.

03

We stopped treating humans-in-the-loop as a UX problem.

It is load-bearing architecture. The moment we treated it that way, adoption tripled and trust followed.

Five rules
we now build every agent against.

1. **Predict, don't react.**
   Build closed loops, not dashboards. The whole economic case lives here.
2. **Primary-path first. Fallback is not default.**
   Codify priority — or the agent will pick its own (and pick wrong).
3. **Agents propose. Systems decide.**
   Put a deterministic gate between any two autonomous agents. Always.
4. **Overlay. Don't rip and replace.**
   Sit on top of what exists. Integration is the moat. Replacement is the fantasy.
5. **The network compounds.**
   Don't build an agent. Build the network of agents on data nobody can copy.

THE NEW BUILDER SKILLS

Three skills replace ten old ones.

Context engineering *(not prompt engineering)*

What you feed the agent matters more than how you word the request. Curate the inputs; the prompt is the last 10%.

Eval-driven development *(not test-driven development)*

Unit tests can't catch non-deterministic agents. Success rate per task and cost per task replace pass/fail as your core loop.

Stack literacy *(not framework loyalty)*

Know which layer is your advantage, which you rent, which is about to commoditize. Don't get caught loving the wrong abstraction.

MONDAY MORNING

Five things you can do *before lunch on Monday.*

01 ***Audit your stack.***

Map your agent project against the six layers. Mark which you own, rent, and are exposed on.

02 ***Pick ONE workflow.***

Not a platform. Not a strategy. One actual workflow inside one team. Predict → act → measure.

03 ***Build your eval set first.***

Twenty real inputs with known good outputs. Before you build the agent. This is your new unit test.

04 ***Put a gate between agents.***

Even if today you only have one. Build the gate now — you'll have a second agent within six months.

05 ***Find your overlay path.***

What existing system does this agent sit on top of? Integration > replacement. Always.

THE HOPEFUL PART

You are not late.

You are exactly on time.

The stack is forming right now. The breakthroughs are six months old. The orchestration layer hasn't even been built yet.

> ***Whatever industry you're in, this is the best time in twenty years to build something that matters.***

ONE THING TO REMEMBER

Build to the stack.
Not to the hype.

The builders who read the stack will define the next decade. That's the room I want you in.

THANK YOU

Tell me about your hardest workflow.

Zaki Medina

Co-Founder, COO & CTO / REVIVE Healthcare Group

zmedina@revivehealthcaregroup.com

revivehealthcaregroup.com / linkedin.com/in/zakimedina