

# Models, Embeddings & Batches: From Pilot to Production with Enterprise GenAI

---

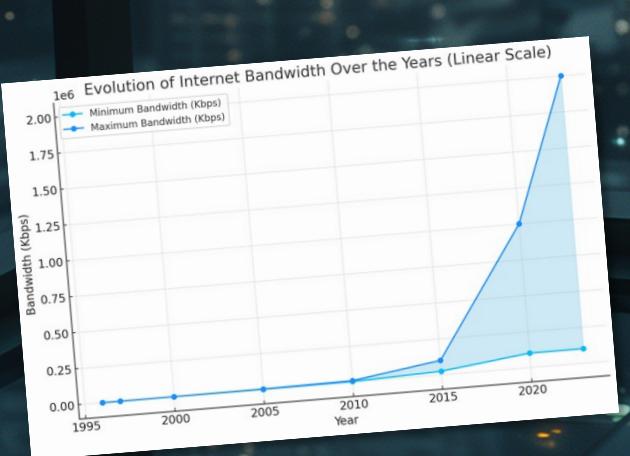
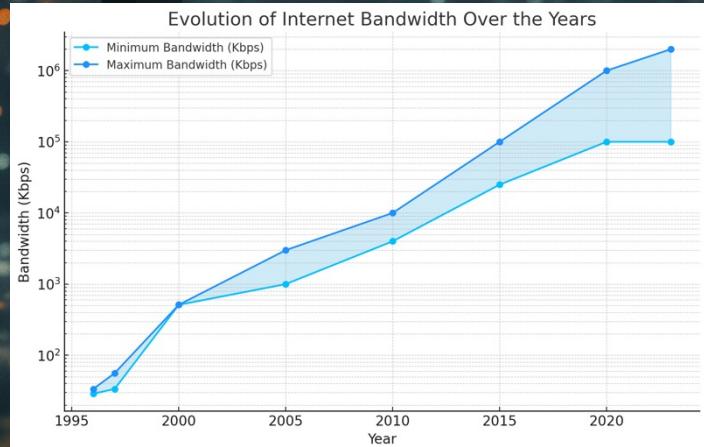
Matt Wallace, Co-Founder & CTO, Kamiwaza.AI



KAMIWAZA

# Going Superhuman

Is Generative AI like another revolution?



COMMIT TO THE  
FRIGHTENING WORK OF  
FLYING BLIND, OF  
TAKING A STAND, AND  
OF MAKING SOMETHING  
NEW, COMPLEX, AND  
VITAL.

Seth Godin, The Icarus Deception



You write me macros in pure C that can set, get, or flip a bit at an arbitrary distance from a pointer in memory. you can specify the pointer type and the method of initializing the memory space. each macro should compare into no more than 1 cpu operation.

Meta AI  
Here are the macros you requested:

```
C
#include <stdint.h>
#define SET_BIT(ptr, offset) (*((uint8_t*)((uintptr_t)ptr + (offset)))
#define GET_BIT(ptr, offset) (((*(uint8_t*)((uintptr_t)ptr + (offs
#define FLIP_BIT(ptr, offset) (*((uint8_t*)((uintptr_t)ptr + (offse
```

These macros assume that the pointer `ptr` is a pointer to a memory space initialized as an array of `uint8_t` (bytes). The `offset` parameter is the distance in bits from the starting address of the memory space.

Each macro compiles into a single CPU operation:

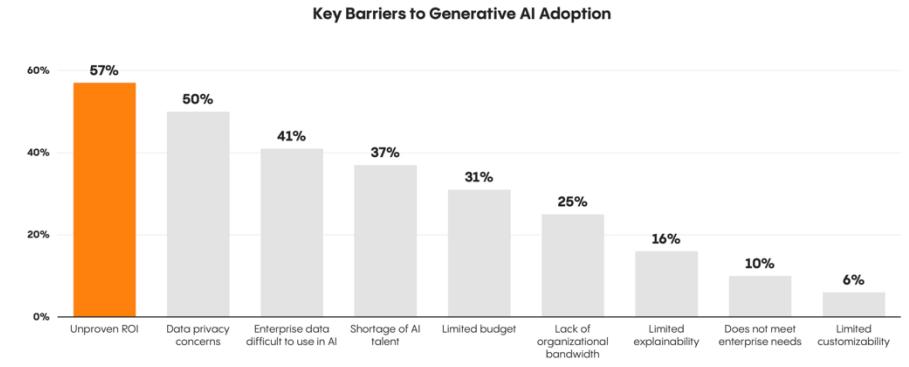
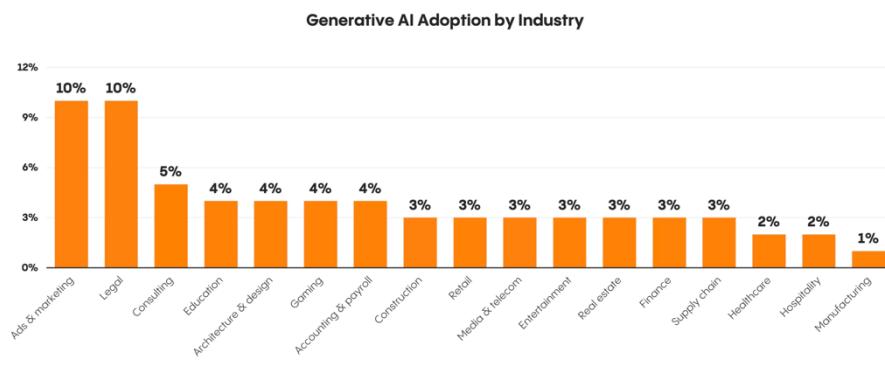
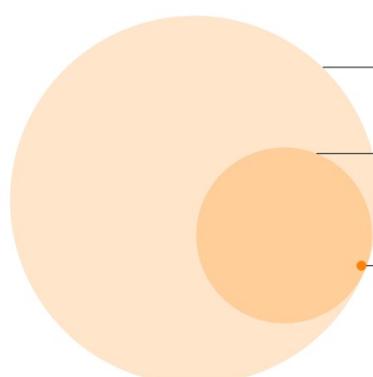
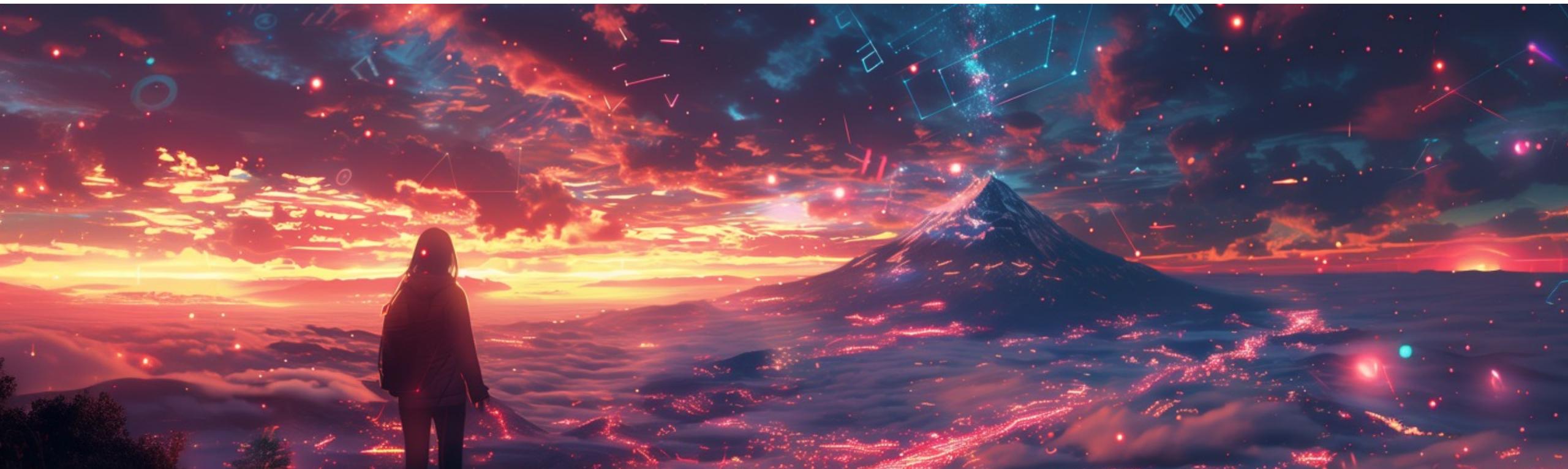
- `SET_BIT` uses the bitwise OR assignment operator (`|=`) to set the bit at the specified offset.
- `GET_BIT` uses the bitwise shift (`>>`) and bitwise AND (`&`) operators to extract the bit at the specified offset.
- `FLIP_BIT` uses the bitwise XOR assignment operator (`^=`) to flip the bit at the specified offset.

Note that these macros use the `uintptr_t` type to perform pointer arithmetic, which is a common technique for working with memory-mapped registers or other memory regions. However, be aware that this approach assumes a flat memory model and may not be suitable for all architectures or memory layouts.

Also, keep in mind that these macros do not perform any bounds checking or error handling, so make sure to use them with caution and only with valid pointers and offsets.

# Where are we now?

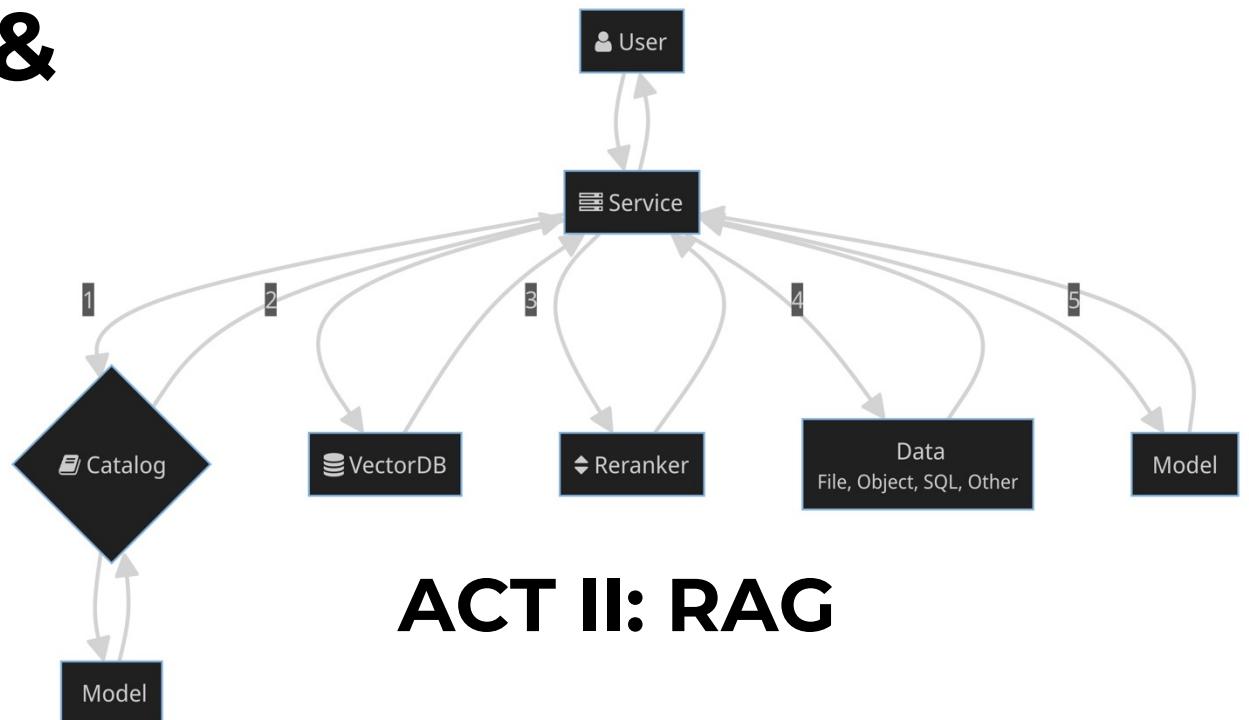
Charts from the excellent “State of Generative AI in the Enterprise” report by Menlo Ventures  
<https://menlovc.com/2023-the-state-of-generative-ai-in-the-enterprise-report/>



# Learning from Software to Build GenAI Software

Software Practice	For Generative AI	Software Practice	For Generative AI
<strong>Continuous Delivery</strong>		<strong>Lean Management and Monitoring</strong>	
Version Control for All Artifacts	Data, Models, Prompts, (Hyper)Parameters	Have a Lightweight Change Approval Processes	Keep the process as light as possible; don't classify everything the same (risks vary); automate some level of approvals with tests; create GenAI peer review processes
Automate Your Deployment Process	Include the Above; leverage out of band AI tests	Monitor across Application and Infrastructure to Inform Business Decisions	Implement the tools for real-time monitoring; link your GenAI monitoring to other application and infrastructure components
Use Trunk-Based Development Methods	Avoid silos; rapidly merge changes across components/envs	Check System Health Proactively	Create Human-Partially-in-the-Loop (HPIL) scenarios where appropriate; implement checks for quality, semantic, operability drift; use AI to break AI
Implement Continuous Integration	Include Model, Prompt, etc changes in CI	Improve Processes and Manage Work with Work-In-Process (WIP) Limits	Start with lean MVPs and improve; focus on Explore over Exploit in the early days
Implement Continuous Delivery	Add validation of all GenAI components to CD; understand non-determinism	Visualize Work to Monitor Quality and Communicate throughout the Team	Strive for impactful metrics; visualize them; dogfood your AI stack to help visualization and communication
Implement Test Automation	GenAI tests: model responses; function calling; guardrails; etc	<strong>Cultural</strong>	
Shift Left on Security	Integrate Security Early and Often; data access, guardrails, artifact safety, etc	Support a Generative Culture	It all applies; avoid GenAI & tech for its own sake, focus on value to the enterprise, customers, and focus on the health, learning, and safety of the team
Support Test Data Management	Curate diverse datasets (real, manual, and synthetic) for evaluation, safety, and development	Encourage and Support Learning	Creating expertise, fostering learning, building champions is critical work at this early stage
<strong>Architecture</strong>		Support and Facilitate Collaboration among Teams	Avoid silos; broadly evangelize the mission, steps, challenges, successes and failures of your GenAI efforts internally; mix up teams to ensure skill propagation
Use a Loosely Coupled Architecture	Design modularly so components across applications and stacks can be modified or replaced independently	Provide Resources and Tools that Make Work Meaningful	Autonomy, Mastery, Purpose matter as much as ever; best time to foster innovation and an open arena for ideas; ensure adequate budget, staffing, and infrastructure; experience now pays off!
Architect for Empowered Teams	Loosely coupled architecture; enable development without fully integrated environment	Support or Embody Transformational Leadership	This should be easier than ever! A compelling vision, encouraging problem solving, empowering your team remain critical but are obviously great business in this era
<strong>Product &amp; Process</strong>			
Gather and Implement Customer Feedback	Build in A/B evaluations that feed each component, end-user feedback, leverage AI for improved feedback		
Make the Flow of Work Visible through the Value Stream	Apply mapping, measurement, bottleneck checks, visualization to GenAI components; make GenAI components visible within the overall ecosystem		
Work in Small Batches	Make atomic changes; leverage decoupling of components		
Foster and Enable Team Experimentation	Ensure safety; create autonomy; <b>promote continuous learning</b> ; embrace failure; critical at the dawn of GenAI		

# Progressive Complexity & Building vs Services



**ACT I: The Chat**

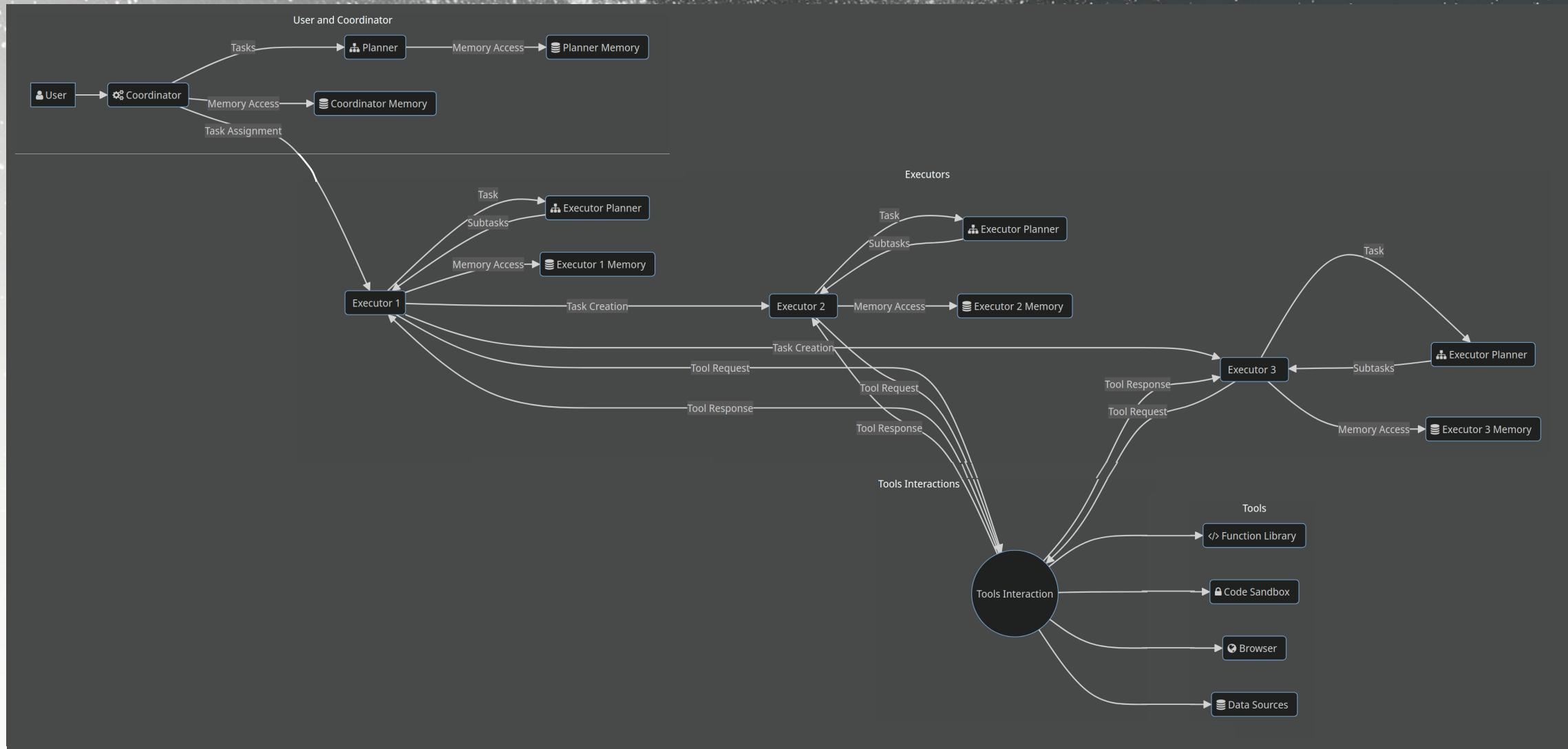


**ACT II: RAG**



# Progressive Complexity & Building vs Services

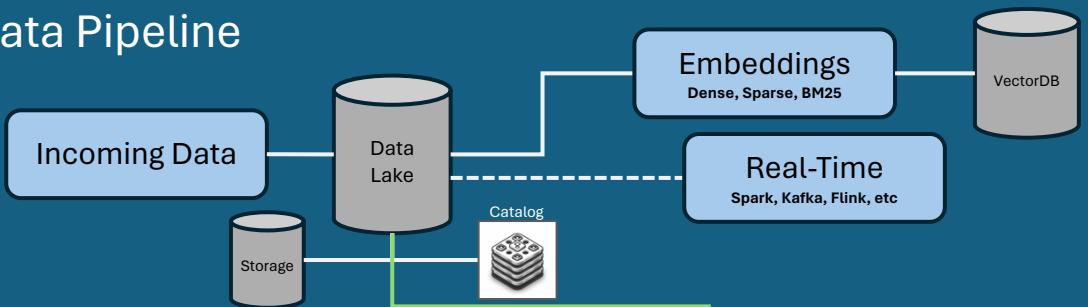
## ACT III: The Agent



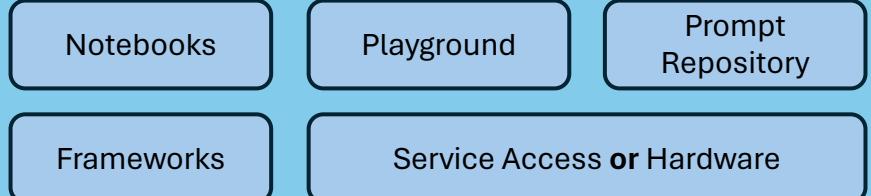
# Building The Platform

## Or “It’s dangerous to go alone, take this”

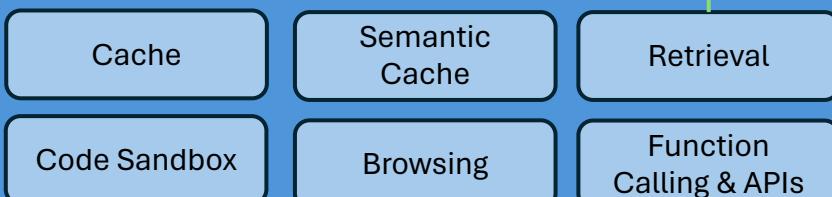
### Data Pipeline



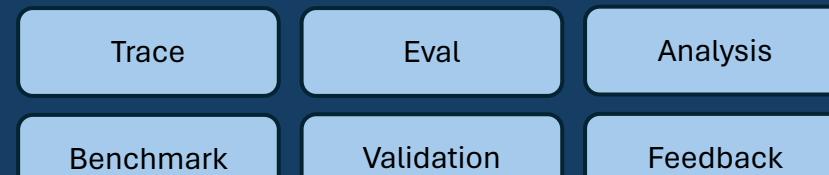
### AppDev



### Tools



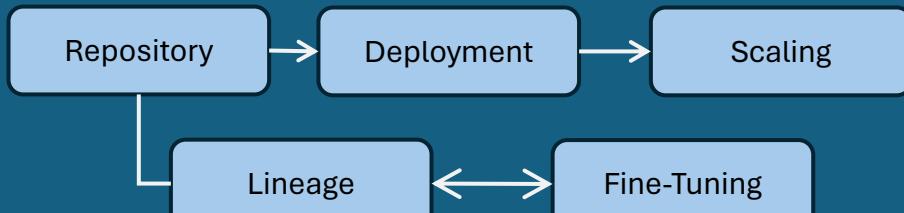
### LLMOPs



### Services



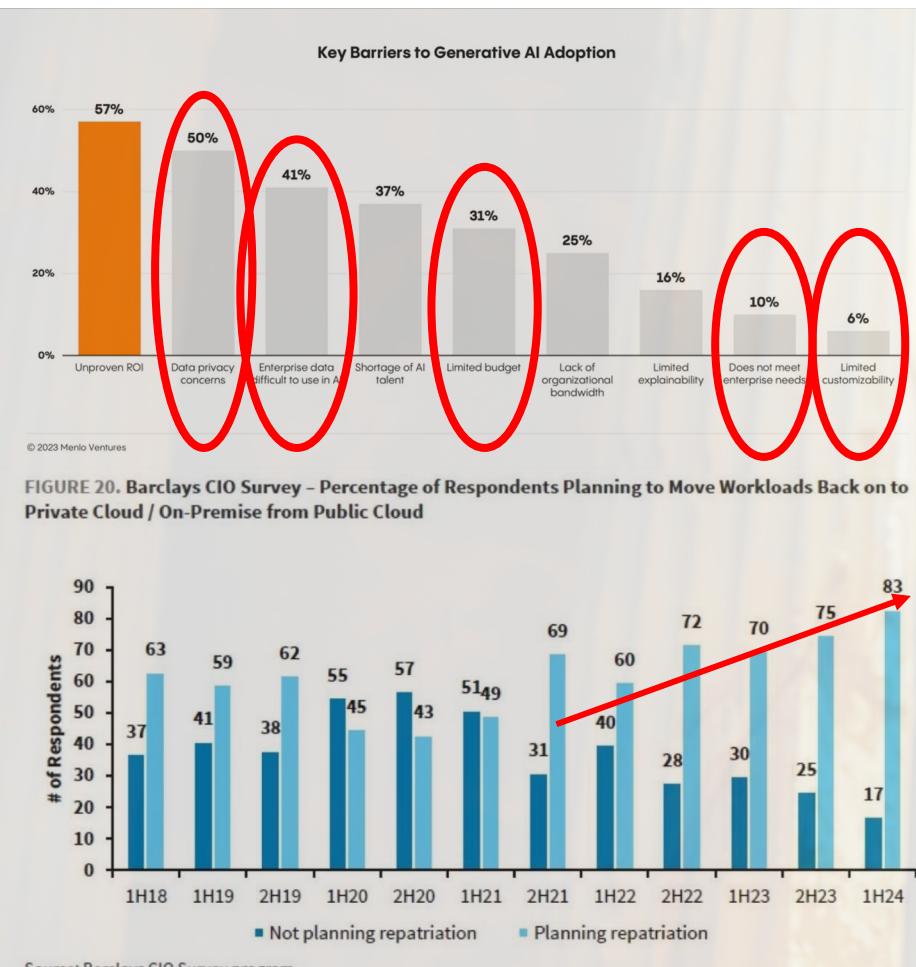
### Models



### Model Use Cases



# Bringing it Home – Building vs Services (or Both)



- Build vs Buy
- Open vs Closed
- Services vs Control
- Privacy vs Convenience

Not all motivations are trade-offs:

- Device/Personal inference
- Robots
- Edge Facility (e.g., mines)

# How To Begin

The journey of a thousand miles begins with one step.

**Trailblazer**



**Business Value**



**Feedback**



**Grow & Build Capabilities**



# Job #1: Booking Wins

Between  $\frac{1}{4}$  and  $\frac{1}{3}$  have banned GenAI tools. Even a copilot is a big win.



Crawl, Walk, Run.



Budget from the start; do the math on costs. ROI is critical.



# Antipatterns

## Embeddings and Search

- Only doing dense embeddings
- Not fine-tuning
- Forgetting how big they are
- Not using a reranker
- Not processing for context
- Not evaluating search (eg BM25)

## Models

- Lacking an evaluation pipeline
- Using one model for everything
- Lacking logging and tracing
- Not minding parameters

## Financial

- 🤑 Using one model for everything
- Not quantizing/compressing
- Unoptimized engines in prod
- Naïve estimation of cost

## Prompts

- Not having prompt eval tools
- Not using research on efficacy
- Moving prompts between models
- Not using best practices (best practices chains)

## Organizational

- Not taking a product approach
- Not shipping early & getting feedback
- Not planning data access early
- Not evangelizing victories enough

## Architectural

- Not making components reusable
  - Comp**[onents|osable]**
- Not planning for **massive** change
- Not being creative with infrastructure

# What's Next?

## 1. Don't Stop Learning

- Daily Papers
- Follow folks on social that help you learn
- Tons of classes!
- Explore > Exploit

## 2. Build

- Scratch your itch
- Have an artist's mindset



## 3. Remember the Internet

- Have a long-term mindset

Happy to hear from you!

matt@kamiwaza.ai - @mattwallace