Composable AI

The Developer Friendly Future of AI Infrastructure

In software engineering, composition is a powerful concept. It allows for building complex systems from simple, interchangeable parts. Think Legos, Docker containers, React components. Langbase extends this concept to AI infrastructure with our **Composable AI** stack using Pipes (/pipe/) and Memory (/memory).

Why Composable AI?

Composable and personalized AI: With Langbase, you can compose multiple models together into pipelines. It's easier to think about, easier to develop for, and each pipe lets you choose which model to use for each task. You can see cost of every step. And allow your customers to hyper-personalize.

Effortlessly zero-config AI infra: Maybe you want to use a smaller, domain-specific model for one task, and a larger general-purpose model for another task. Langbase makes it easy to use the right primitives and tools for each part of the job and provides developers with a zero-config composable AI infrastructure.

That's a nice way of saying, *you get a unicorn-scale API in minutes, not months*.

> ***The most common problem** I hear about in Gen AI space is that my AI agents are too complex and I can't scale them, too much AI talking to AI. I don't have control, I don't understand the cost, and the impact of this change vs that. Time from new model to prod is too long. Feels static, my customers can't personalize it. ⌘ Langbase fixes all this. — AA (https://www.linkedin.com/in/MrAhmadAwais/)*

Interactive Example: Composable AI Email Agent

But how does Composable AI work?

Here's an interactive example of a composable AI Email Agent: Classifies, summarizes, responds. Click to send a spam or valid email and check how composable it is: Swap any pipes, any LLM, hyper-personalize (you or your users), observe costs. Everything is composable.

Example: Composable AI Email Agent

I have built an AI email agent that can read my emails, understand the sentiment, summarize, and respond to them. Let's break it down to how it works, hint several pipes working together to make smart personalized decisions.

1. I created a pipe: `email-sentiment` — this one reads my emails to understand the sentiment
2. `email-summarizer` pipe — it summarizes my emails so I can quickly understand them
3. `email-decision-maker` pipe — should I respond? is it urgent? is it a newsletter?
4. If `email-decision-maker` pipe says *yes*, then I need to respond. This invokes the final pipe
5. `email-writer` pipe — writes a draft response to my emails with one of the eight formats I have

Why Composable AI is powerful?

Ah, the power of composition. I can swap out any of these pipes with a new one.

- **Flexibility**: Swap components without rewriting everything
- **Reusability**: Build complex systems from simple, tested parts
- **Scalability**: Optimize at the component level for better performance
- **Observability**: Monitor and debug each step of your AI pipeline

Control flow

- Maybe I want to use a different sentiment analysis model
- Or maybe I want to use a different summarizer when I'm on vacation
- I can chose a different LLM (small or large) based on the task
- BTW I definitely use a different `decision-maker` pipe on a busy day.

Extensibility

- **Add more when needed**: I can also add more pipes to this pipeline. Maybe I want to add a pipe that checks my calendar or the weather before I respond to an email. You get the idea. Always bet on composition.
- **Eight Formats to write emails**: And I have several formats. Because Pipes are composable, I have eight different versions of `email-writer` pipe. I have a pipe `email-pick-writer` that picks the correct pipe to draft a response with. Why? I talk to my friends differently than my investors, reports, managers, vendors — you name it.

Long-term memory and context awareness

- By the way, I have all my emails in an `emails-store` memory, which any of these pipes can refer to if needed. That's managed semantic RAG (/memory) over all the emails I have ever received.
- And yes, my `emails-smart-spam` memory knows all the pesky smart spam emails that I don't want to see in my inbox.

Cost & Observability

- Because each intent and action is mapped out Pipe — which is an excellent primitive for using LLMs, I can see everything related to cost, usage, and effectiveness of each pipe. I can see how many emails were processed, how many were responded to, how many were marked as spam, etc.
- I can switch LLMs for any of these actions, fork a pipe (https://langbase.com/docs/features/fork), and see how it performs. I can version my pipes and see how the new version performs against the old one.
- And we're just getting started …

Why Developers Love It

- **Modular**: Build, test, and deploy pipes x memorysets independently
- **Extensible**: API-first no dependency on a single language
- **Version Control Friendly**: Track changes at the pipe level
- **Cost-Effective**: Optimize resource usage for each AI task
- **Stakeholder Friendly**: Collaborate with your team on each pipe and memory. All your R&D team, engineering, product, GTM (marketing, sales), and even stakeholders can collaborate on the same pipe. It's like a Google Doc x GitHub for AI. That's what makes it so powerful.

Each pipe and memory are like a docker container. You can have any number of pipes and memorysets.

Can't wait to share more exciting examples of composable AI. We're cookin!!

We'll share more on this soon. Follow us on Twitter (https://twitter.com/LangbaseInc) and LinkedIn (https://www.linkedin.com/company/langbase/) for updates.
