LangChain Expression Language (LCEL)

LangChain Expression Language, or LCEL, is a declarative way to easily compose chains together. LCEL was designed from day 1 to **support putting prototypes in production, with no code changes**, from the simplest “prompt + LLM” chain to the most complex chains (we’ve seen folks successfully run LCEL chains with 100s of steps in production). To highlight a few of the reasons you might want to use LCEL:

Streaming support When you build your chains with LCEL you get the best possible time-to-first-token (time elapsed until the first chunk of output comes out). For some chains this means eg. we stream tokens straight from an LLM to a streaming output parser, and you get back parsed, incremental chunks of output at the same rate as the LLM provider outputs the raw tokens.

Async support Any chain built with LCEL can be called both with the synchronous API (eg. in your Jupyter notebook while prototyping) as well as with the asynchronous API (eg. in a LangServe server). This enables using the same code for prototypes and in production, with great performance, and the ability to handle many concurrent requests in the same server.

Optimized parallel execution Whenever your LCEL chains have steps that can be executed in parallel (eg if you fetch documents from multiple retrievers) we automatically do it, both in the sync and the async interfaces, for the smallest possible latency.

Retries and fallbacks Configure retries and fallbacks for any part of your LCEL chain. This is a great way to make your chains more reliable at scale. We’re currently working on adding streaming support for retries/fallbacks, so you can get the added reliability without any latency cost.

Access intermediate results For more complex chains it’s often very useful to access the results of intermediate steps even before the final output is produced. This can be used to let end-users know something is happening, or even just to debug your chain. You can stream intermediate results, and it’s available on every LangServe server.

Input and output schemas Input and output schemas give every LCEL chain Pydantic and JSONSchema schemas inferred from the structure of your chain. This can be used for validation of inputs and outputs, and is an integral part of LangServe.

Seamless LangSmith tracing integration As your chains get more and more complex, it becomes increasingly important to understand what exactly is happening at every step. With LCEL, **all** steps are automatically logged to LangSmith for maximum observability and debuggability.

Seamless LangServe deployment integration Any chain created with LCEL can be easily deployed using LangServe.