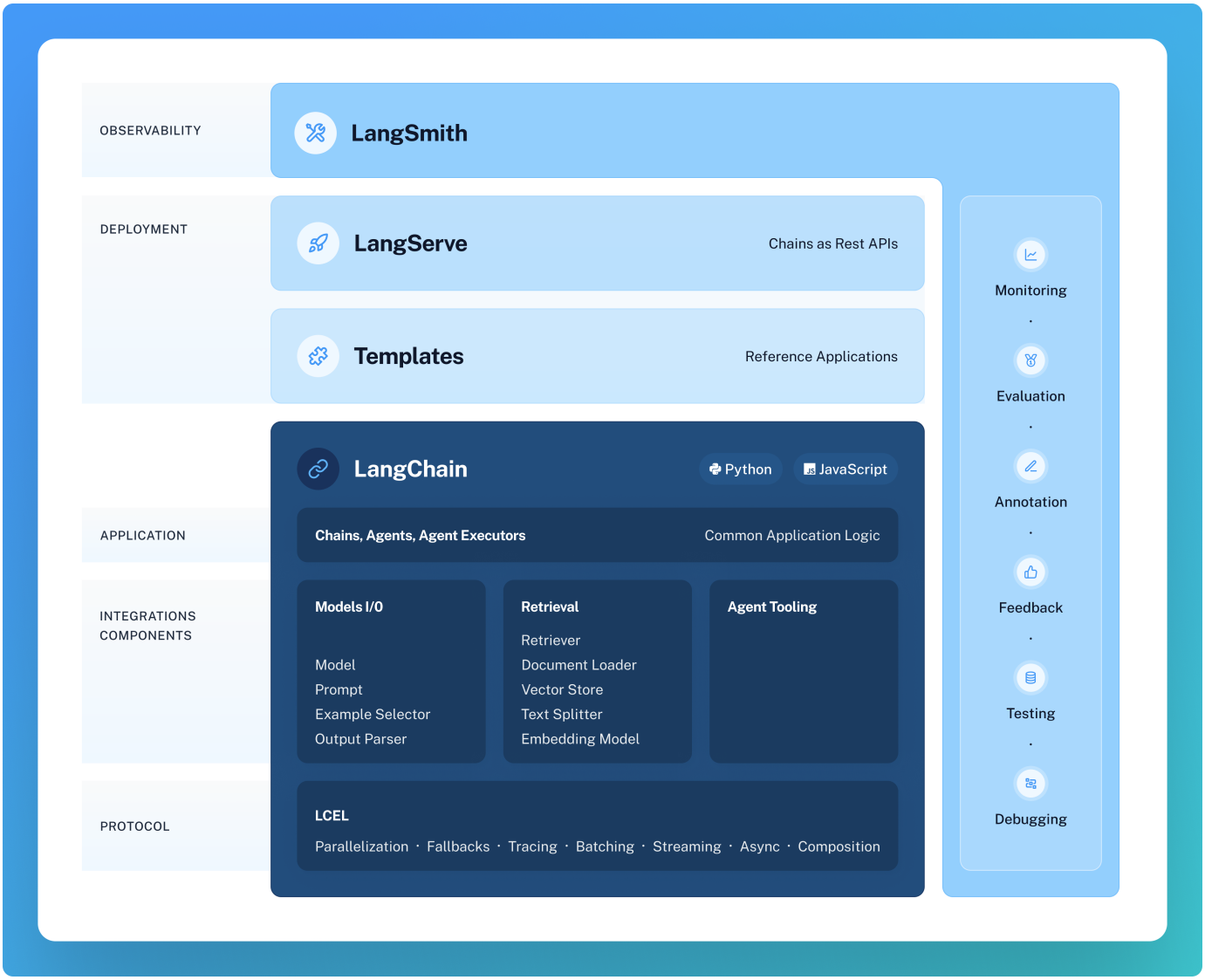
Introduction

LangChain is a framework for developing applications powered by language models. It enables applications that:

- **Are context-aware**: connect a language model to sources of context (prompt instructions, few shot examples, content to ground its response in, etc.)
- **Reason**: rely on a language model to reason (about how to answer based on provided context, what actions to take, etc.)

This framework consists of several parts.

- **LangChain Libraries**: The Python and JavaScript libraries. Contains interfaces and integrations for a myriad of components, a basic run time for combining these components into chains and agents, and off-the-shelf implementations of chains and agents.
- **LangChain Templates**: A collection of easily deployable reference architectures for a wide variety of tasks. (*Python only*)
- **LangServe**: A library for deploying LangChain chains as a REST API. (*Python only*)
- **LangSmith**: A developer platform that lets you debug, test, evaluate, and monitor chains built on any LLM framework and seamlessly integrates with LangChain.



Together, these products simplify the entire application lifecycle:

- **Develop**: Write your applications in LangChain/LangChain.js. Hit the ground running using Templates for reference.
- **Productionize**: Use LangSmith to inspect, test and monitor your chains, so that you can constantly improve and deploy with confidence.
- **Deploy**: Turn any chain into an API with LangServe.

LangChain Libraries

The main value props of the LangChain packages are:

1. **Components**: composable tools and integrations for working with language models. Components are modular and easy-to-use, whether you are using the rest of the LangChain framework or not

2. **Off-the-shelf chains**: built-in assemblages of components for accomplishing higher-level tasks

Off-the-shelf chains make it easy to get started. Components make it easy to customize existing chains and build new ones.

Get started

Here's how to install LangChain, set up your environment, and start building.

We recommend following our Quickstart guide to familiarize yourself with the framework by building your first LangChain application.

Read up on our Security best practices to make sure you're developing safely with LangChain.

> **NOTE**
>
> These docs focus on the JS/TS LangChain library. Head here for docs on the Python LangChain library.

LangChain Expression Language (LCEL)

LCEL is a declarative way to compose chains. 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.

- **Overview**: LCEL and its benefits
- **Interface**: The standard interface for LCEL objects
- **How-to**: Key features of LCEL
- **Cookbook**: Example code for accomplishing common tasks

Modules

LangChain provides standard, extendable interfaces and integrations for the following modules:

Model I/O

Interface with language models

Retrieval

Interface with application-specific data

Agents

Let models choose which tools to use given high-level directives

Examples, ecosystem, and resources

Use cases

Walkthroughs and techniques for common end-to-end use cases, like:

- Document question answering
- RAG
- Agents
- and much more...

Integrations

LangChain is part of a rich ecosystem of tools that integrate with our framework and build on top of it. Check out our growing list of integrations.

API reference

Head to the reference section for full documentation of all classes and methods in the LangChain and LangChain Experimental packages.

Developer's guide

Check out the developer's guide for guidelines on contributing and help getting your dev environment set up.

Community

Head to the Community navigator to find places to ask questions, share feedback, meet other developers, and dream about the future of LLM's.