

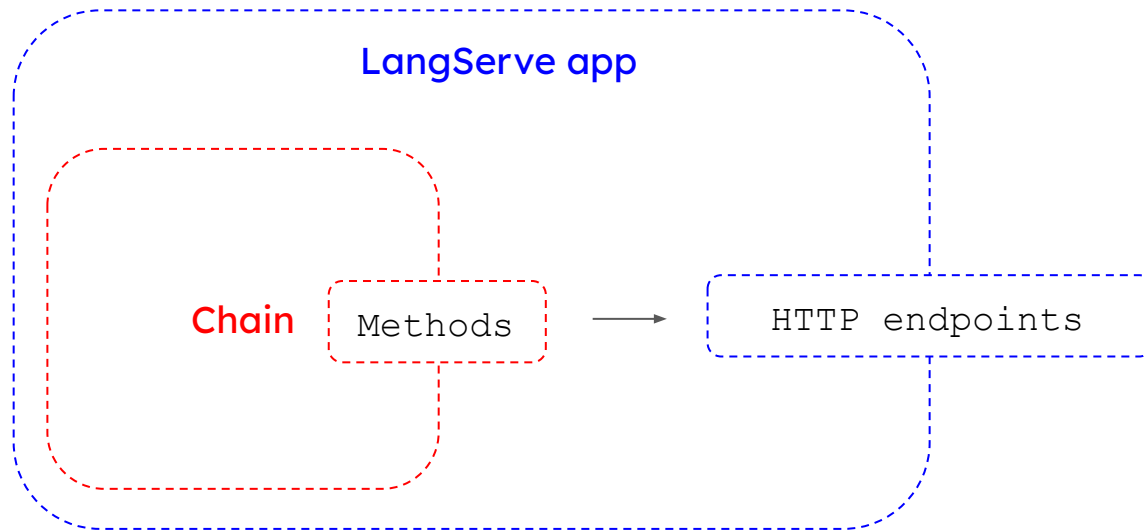
# Deploying templates w/ hosted LangServe: neo4j-semantic-layer example

Lance Martin  
Software Engineer, LangChain  
[@RLanceMartin](https://twitter.com/RLanceMartin)

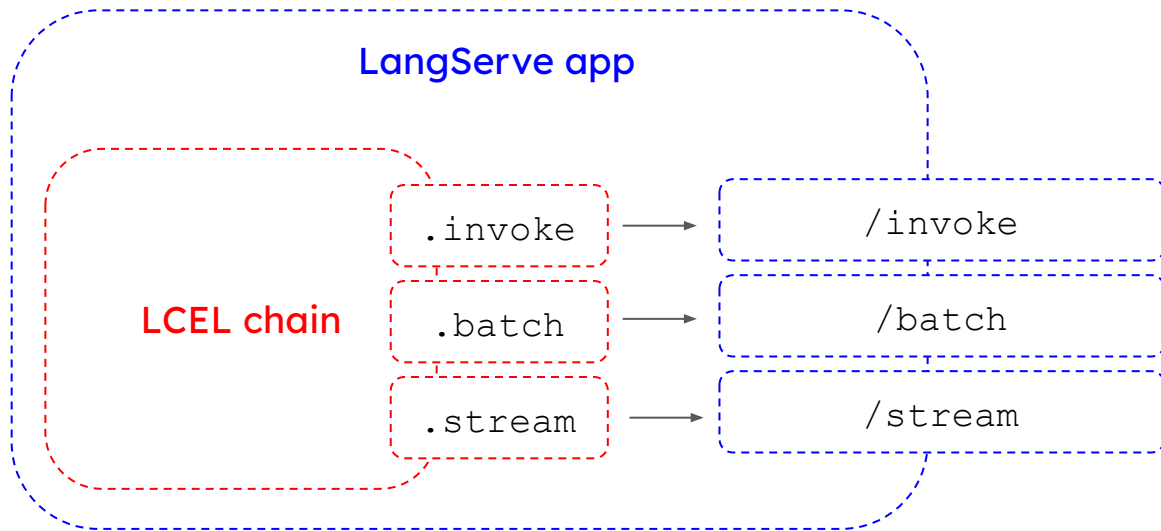
## Create a LangServe app

```
$ conda create -n langserve-test-env python=3.11
$ conda activate langserve-test-env
$ pip install -U "langchain-cli[serve]" "langserve[all]"
$ langchain app new .
```

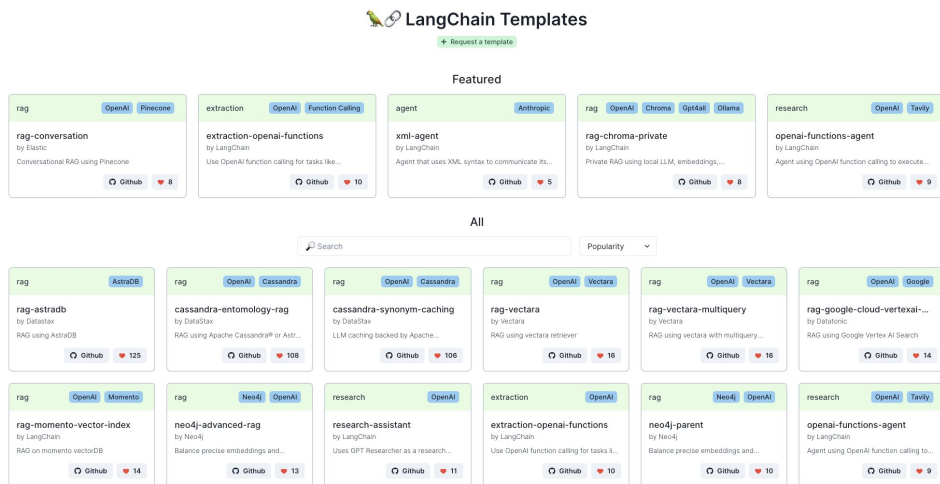
LangServe creates a web service for any chain



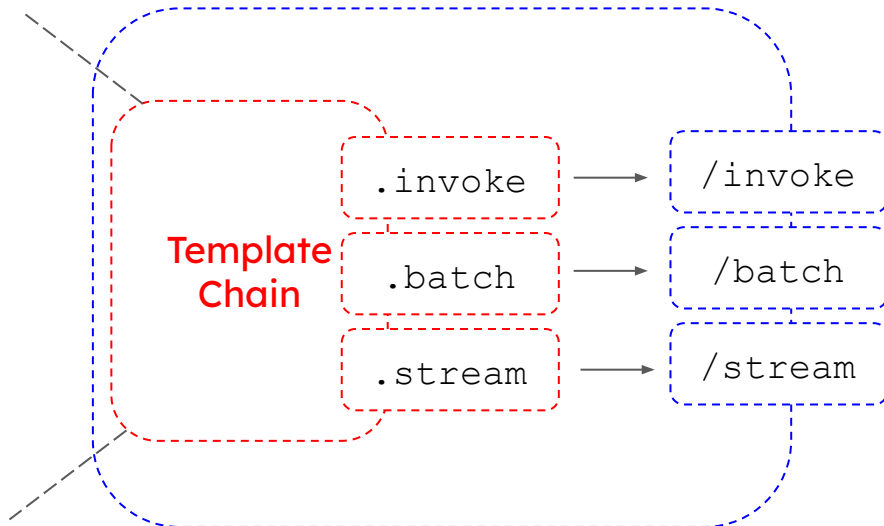
## HTTP endpoints that expose methods of LCEL chain



# Library of templates that can be immediately deployed!



## Web service (LangServe)



## Choose our template

```
$ langchain app new .
```

```
What package would you like to add? (leave blank to skip) :neo4j-semantic-layer
```

```
1 added. Any more packages (leave blank to end)?:
```

```
Would you like to install these templates into your environment with pip? [y/N] y
```

## App structure

`/app`

`/server.py`

Add LCEL chain methods as  
endpoints in web service

`/packages`

`/neo4j-semantic-layer`

`/neo4j-semantic-layer`

`/__init__.py`

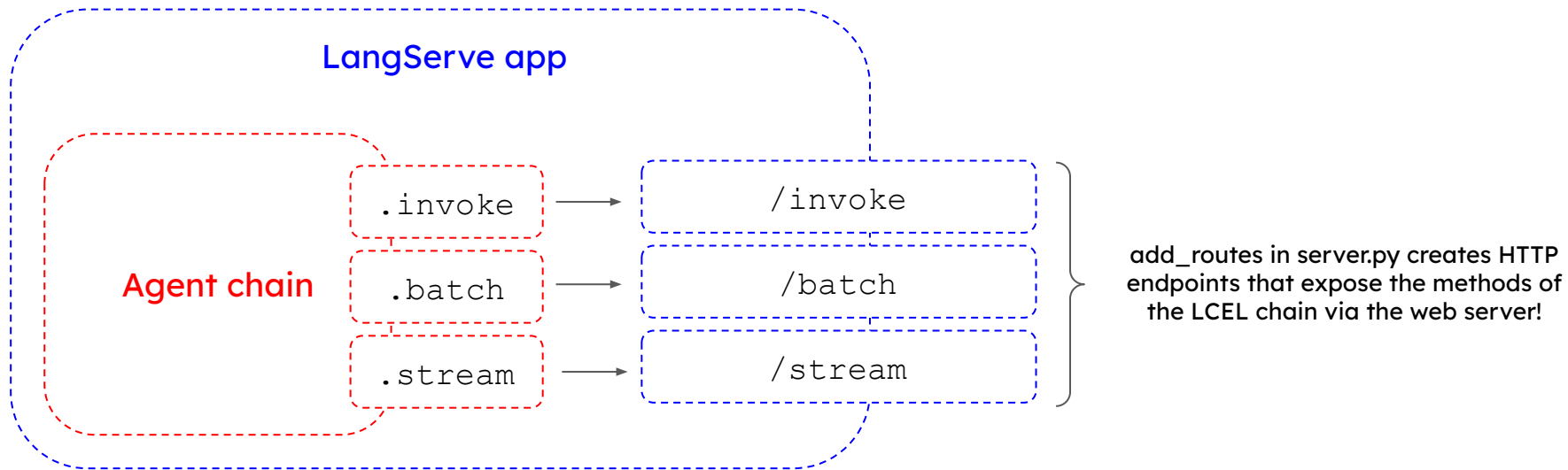
`/agent.py`

`/pyproject.toml`

Neo4j-semantic-layer  
template is a package  
and agent.py has our  
LCEL chain defined

`/pyproject.toml`

## App structure



### server.py

```
from neo4j_semantic_layer import agent_executor as neo4j_semantic_agent
add_routes(app, neo4j_semantic_agent, path="/neo4j-semantic-layer")
```



## Run the server locally

```
$ poetry install  
$ langchain serve
```

## Use hosted LangServe

<https://github.com/langchain-ai/neo4j-semantic-layer>

# Appendix

## LCEL enables composition of chains

```
from langchain_openai import ChatOpenAI
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.output_parsers import StrOutputParser
```

```
prompt = ChatPromptTemplate.from_template("Tell me a short joke about {topic}")
```

```
model = ChatOpenAI(model="gpt-3.5-turbo")
```

```
output_parser = StrOutputParser()
```

```
chain = prompt | model | output_parser
```

LCEL objects have a runnable interface w/ common invocation methods

```
chain.invoke({"topic": "bears"})
```

```
AIMessage(content="Why don't bears wear shoes?\n\nBecause they already have bear feet!")
```

# Many use-cases can be expressed using LCEL objects

## Cookbook

Example code for accomplishing common tasks with the LangChain Expression Language (LCEL). These examples show how to compose different Runnable (the core LCEL interface) components to achieve various tasks. If you're just getting acquainted with LCEL, the [Prompt + LLM](#) page is a good place to start.

### Prompt + LLM

The most common and valuable composition is taking:

### RAG

Let's look at adding in a retrieval step to a prompt and LLM, whic...

### Multiple chains

Runnables can easily be used to string together multiple Chains

### Querying a SQL DB

We can replicate our SQLiteDatabaseChain with Runnables.

### Agents

You can pass a Runnable into an agent.

### Code writing

Example of how to use LCEL to write Python code.

### Routing by semantic similarity

With LCEL you can easily add custom routing

### Adding memory

This shows how to add memory to an arbitrary chain. Right now, ...

### Adding moderation

This shows how to add in moderation (or other safeguards) arou...

### Managing prompt size

Agents dynamically call tools. The results of those tool calls are ...

## Challenges w/ production

	Prototyping	Production
Async Support	Often synchronous (e.g., Jupyter)	Often Required
Streaming	Often optional	
Parallel execution		
Retries / fallback		
Access intermediate results		
I / O validation		
Ease of deployment		
Observability		

LCEL enables same code for prototyping + production!

	Prototyping	Production
Async Support	Every <b>LCEL</b> object supports these!	
Streaming		
Parallel execution		
Retries / fallback		
Access intermediate results		
I / O validation		
Ease of deployment		
Observability		



## LangServe addresses I/O validation and deployment

	Prototyping	Production
Async Support	LCEL	
Streaming		
Parallel execution		
Retries / fallback		
Access intermediate results	LCEL + <b>LangServe</b>	
I / O validation		
Ease of deployment		
Observability		

## Interact with the app in various ways

### LangServe app: FastAPI

/invoke

/batch

/stream

### SDK, requests, curl, Playground

```
from langchain.schema import SystemMessage, HumanMessage
from langchain.prompts import ChatPromptTemplate
from langchain.schema.runnable import RunnableMap
from langserve import RemoteRunnable

openai = RemoteRunnable("http://localhost:8000/openai/")
anthropic = RemoteRunnable("http://localhost:8000/anthropic/")
joke_chain = RemoteRunnable("http://localhost:8000/joke/")

joke_chain.invoke({"topic": "parrots"})
```

```
import requests

response = requests.post(
    "http://localhost:8000/joke/invoke",
    json={'input': {'topic': 'cats'}}
)

response.json()
```

```
curl --location --request POST 'http://localhost:8000/joke/invoke' \
--header 'Content-Type: application/json' \
--data-raw '{
  "input": {
    "topic": "cats"
  }
}'
```

#### LangServe Playground

Try it

Inputs

INPUT\*

is a required property

## Challenges w/ production

	Prototyping	Production
Async Support	LCEL	
Streaming		
Parallel execution		
Retries / fallback		
Access intermediate results	LCEL + LangServe	
I / O validation		
Ease of deployment		
Observability	LCEL + LangServe + <b>LangSmith</b>	

# Use hosted LangServe for a managed server

← Create deployment

## Create New Deployment

Launch a server that's publicly available in one-click

### Deployment details

Name it and link to the GitHub repository

 langchain-ai ▾

🔍 Select a repo...

GitHub Repo URL \*

 Paste URL...

This field is required

Name \*

ht-gripping-baseline-11

Subdirectory (optional)

.

Git Reference (optional)


main

### Environment Variables

Use environment variables to specify API keys to LLM providers and other services

+ Add

## Both expose the agent methods and playground!

 **LangServe** Playground

Try it

Inputs Reset

INPUT\*

i am interested in film noir. suggest some of the most famous film noir and provide a description of the major themes.

CHAT HISTORY +

HUMAN

i am interested in film noir. suggest some of the most famous film noir and provide a description of the major themes.



AI


Here are some famous Film-Noir movies:

1. **Sunset Blvd. (a.k.a. Sunset Boulevard)**: This is a classic film noir about a struggling screenwriter who is drawn into the world of an aging silent film star who is descending into madness. The movie explores themes of ambition, desperation, and the dark side of Hollywood.
2. **High and Low (Tengoku to jigoku)**: This is a Japanese film noir directed by Akira Kurosawa. The story revolves around a businessman who is faced with a moral


## It also integrates with LangSmith


Trace


Stats  Most relevant 

 /neo4j-semantic-layer 🟢 SUCCESS

🕒 27.73s 💬 870


 ChatOpenAI 2.37s

 Recommender 0.03s

 ChatOpenAI 25.21s

🕒 Some runs have been hidden. [Show 14 hidden runs](#)

### ChatOpenAI

[Run](#) [Feedback](#) [Metadata](#) 

#### Functions & Tools

Information >

Recommender >

#### Input

**SYSTEM**

You are a helpful assistant that finds information about movies and recommends them. If tools require follow up questions, make sure to ask the user for clarification. Make sure to include any available options that need to be clarified in the follow up questions

**HUMAN**

i am interested in film noir. suggest some of the most famous film noir and provide a description of the major themes.