

## **Streaming in Langchain is a little confusing**



Docs aren't super clear on how to set it up



Several different config options control streaming and some override others



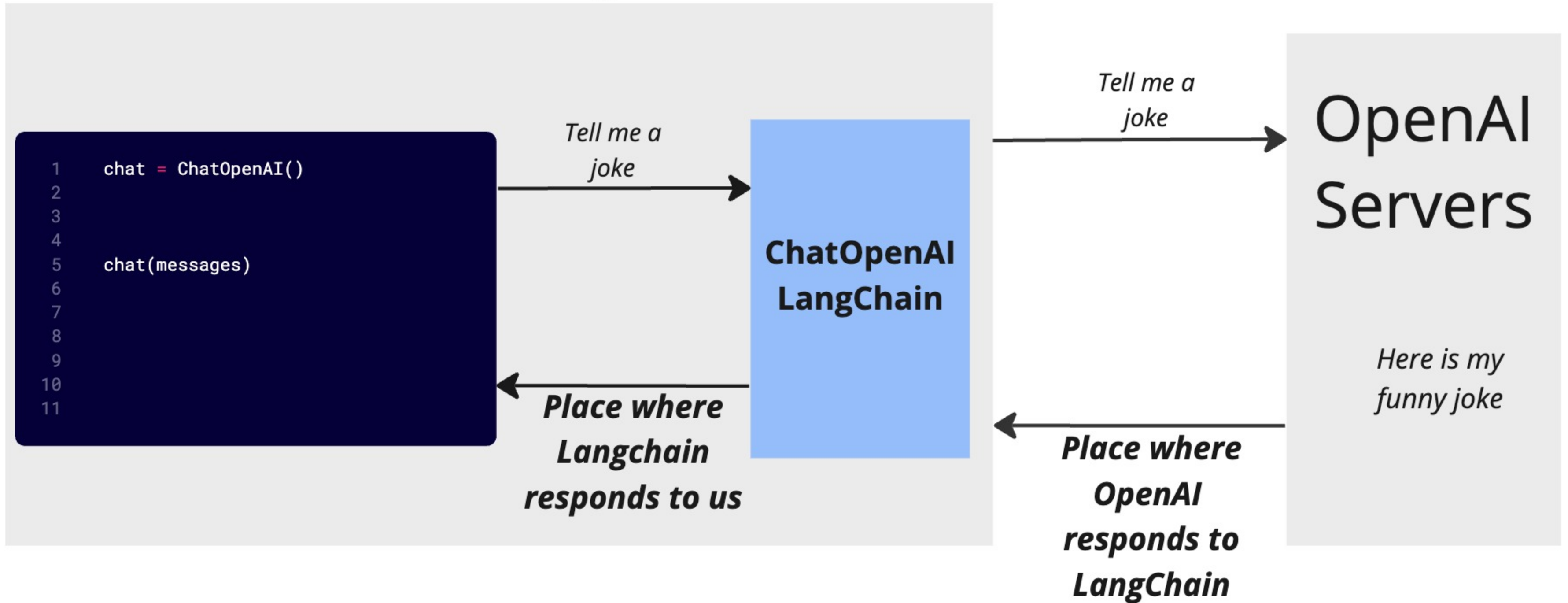
We're going to write some code in a temp test file to figure this stuff out

**LLMs are happy to stream!**

**Chains are unhappy and don't want to stream!**

*Our job will be to convince a  
chain to stream*

## Our Program

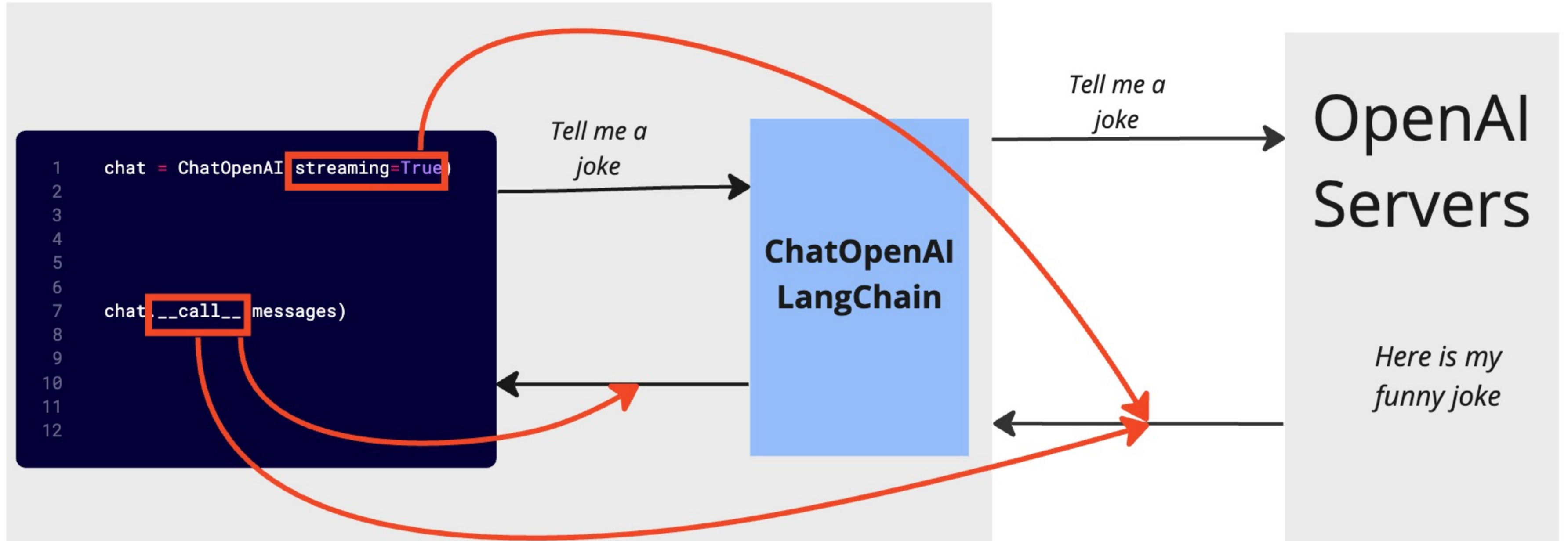


```
1 chat = ChatOpenAI(streaming=True)
2
3
4 chat.__call__(messages)
```

Controls how OpenAI  
responds to LangChain

Controls how OpenAI  
responds to LangChain  
**and**  
controls how LangChain  
responds to us

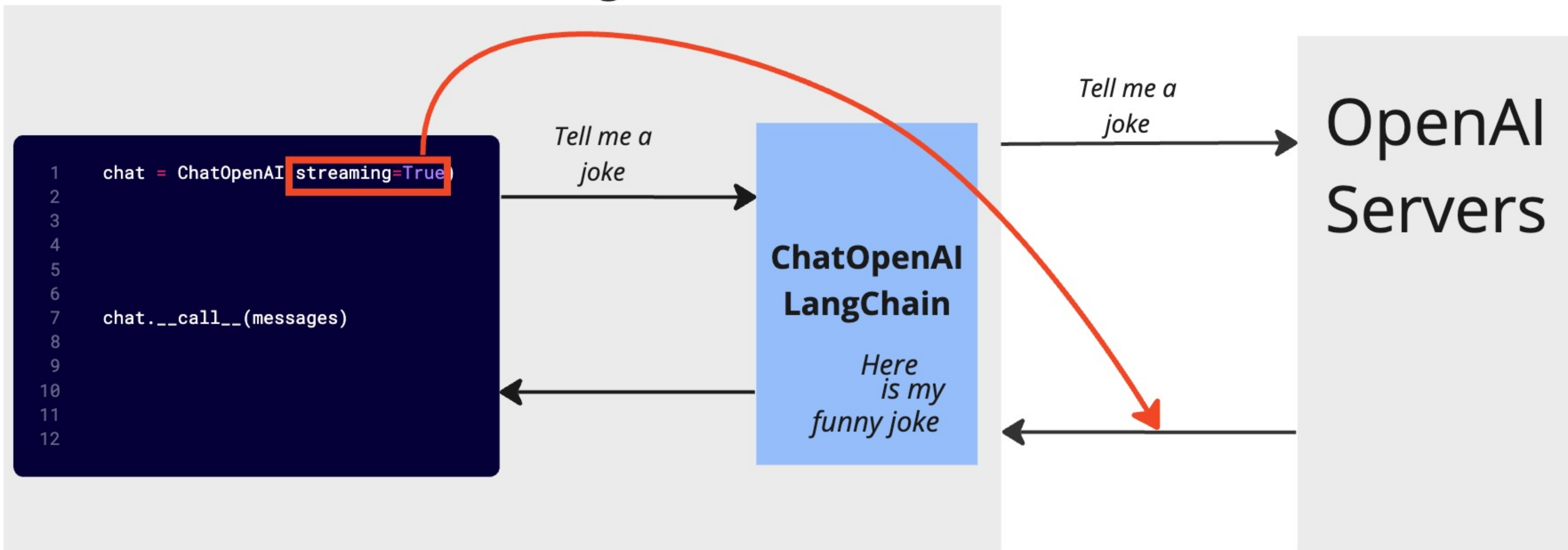
## Our Program





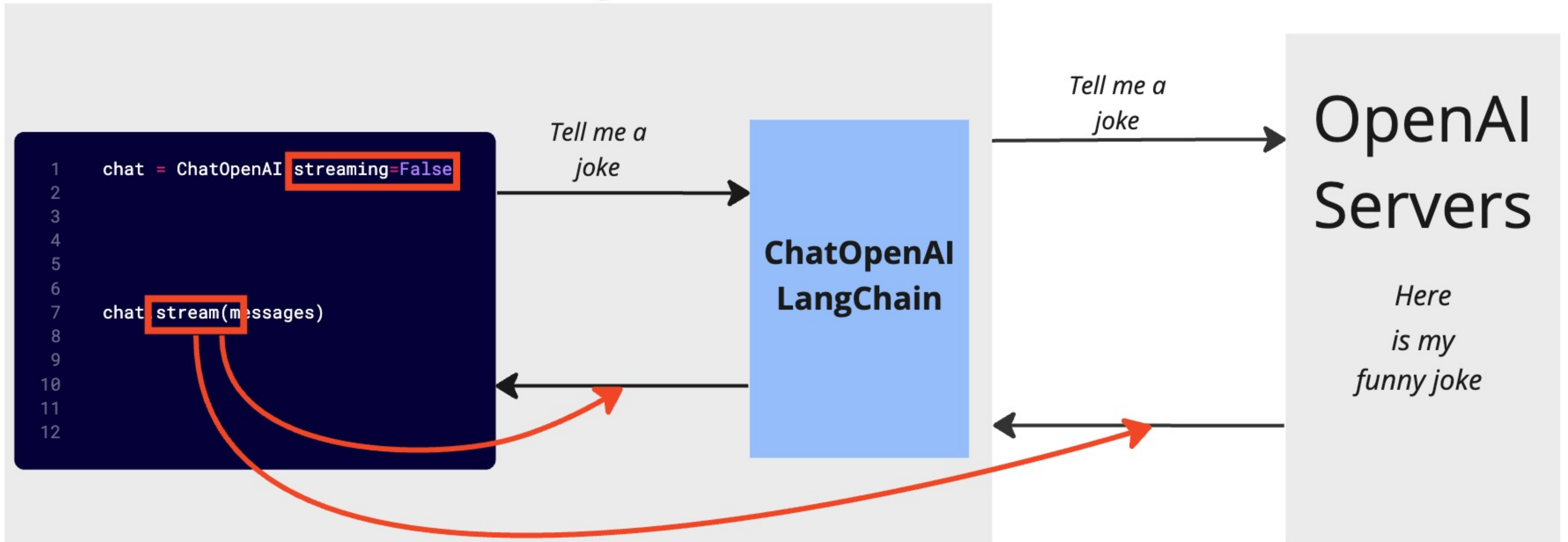
**streaming=True forces OpenAI to  
stream to LangChain**

## Our Program



**'stream' forces streaming  
everywhere *regardless of the  
'streaming' flag***

## Our Program



```
1 llm = ChatOpenAI(streaming=True)
2 chain = LLMChain(llm=llm)
3
4
5
6
7 chain.__call__("tell me a joke")
8
9
10
11
12
```

LLMChain

Tell me a  
joke

No, I don't  
want this

ChatOpenAI

asdf

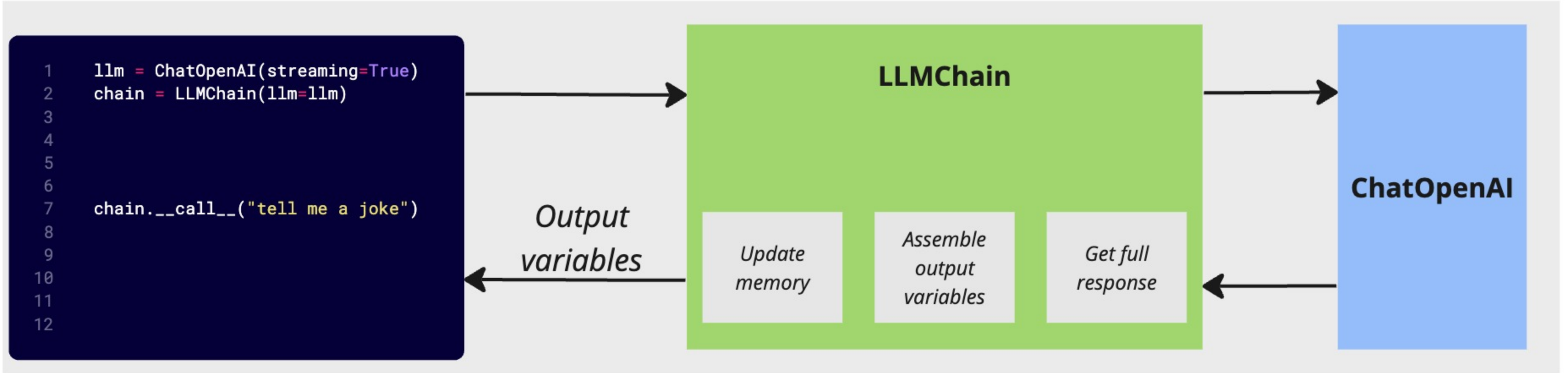
Tell me a  
joke

OpenAI  
Servers

is my  
funny joke

**Streaming!**





**LLM Chains *really* want to process the full response before returning anything to you**

```
1 llm = ChatOpenAI(streaming=True)
2 chain = LLMChain(llm=llm)
3
4
5
6
7 chain.__call__("tell me a joke")
8
9
10
11
12
```

*Tell me a  
joke*

LLMChain

*Tell me a  
joke*

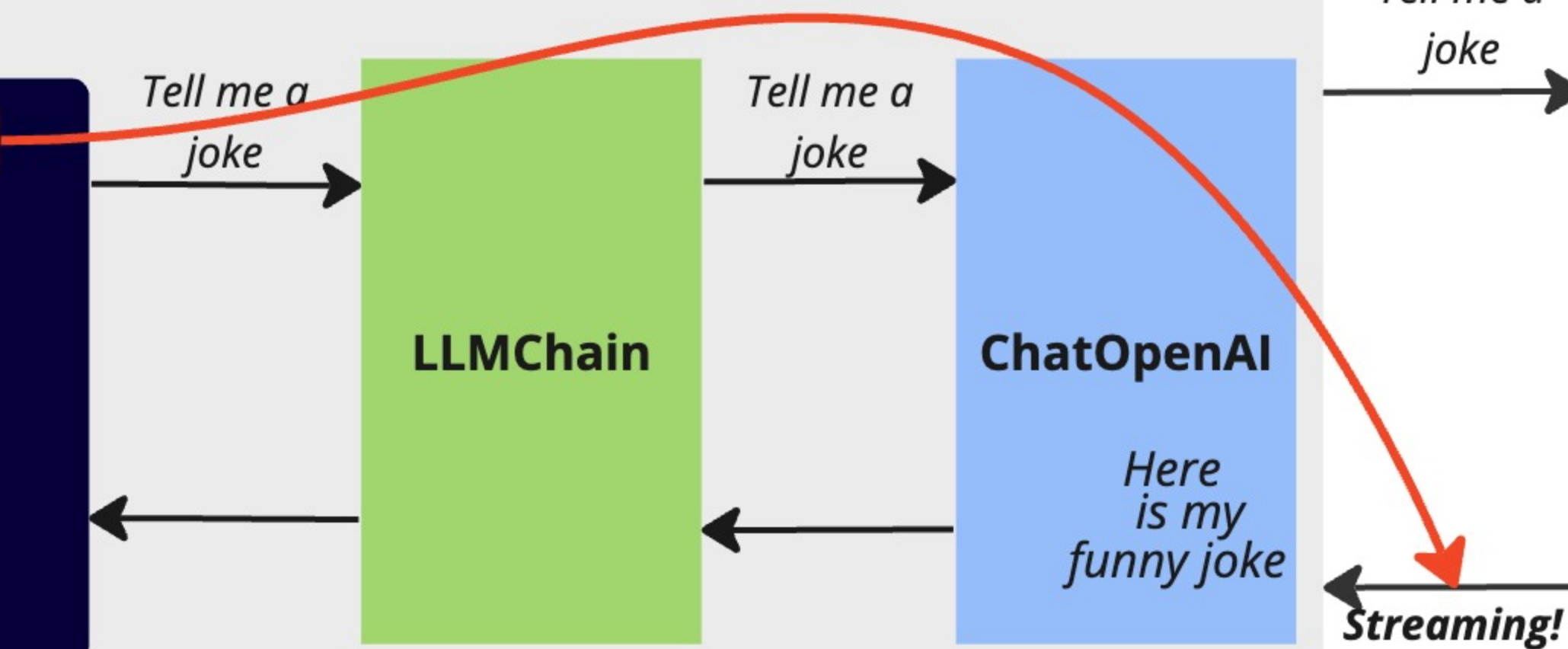
ChatOpenAI

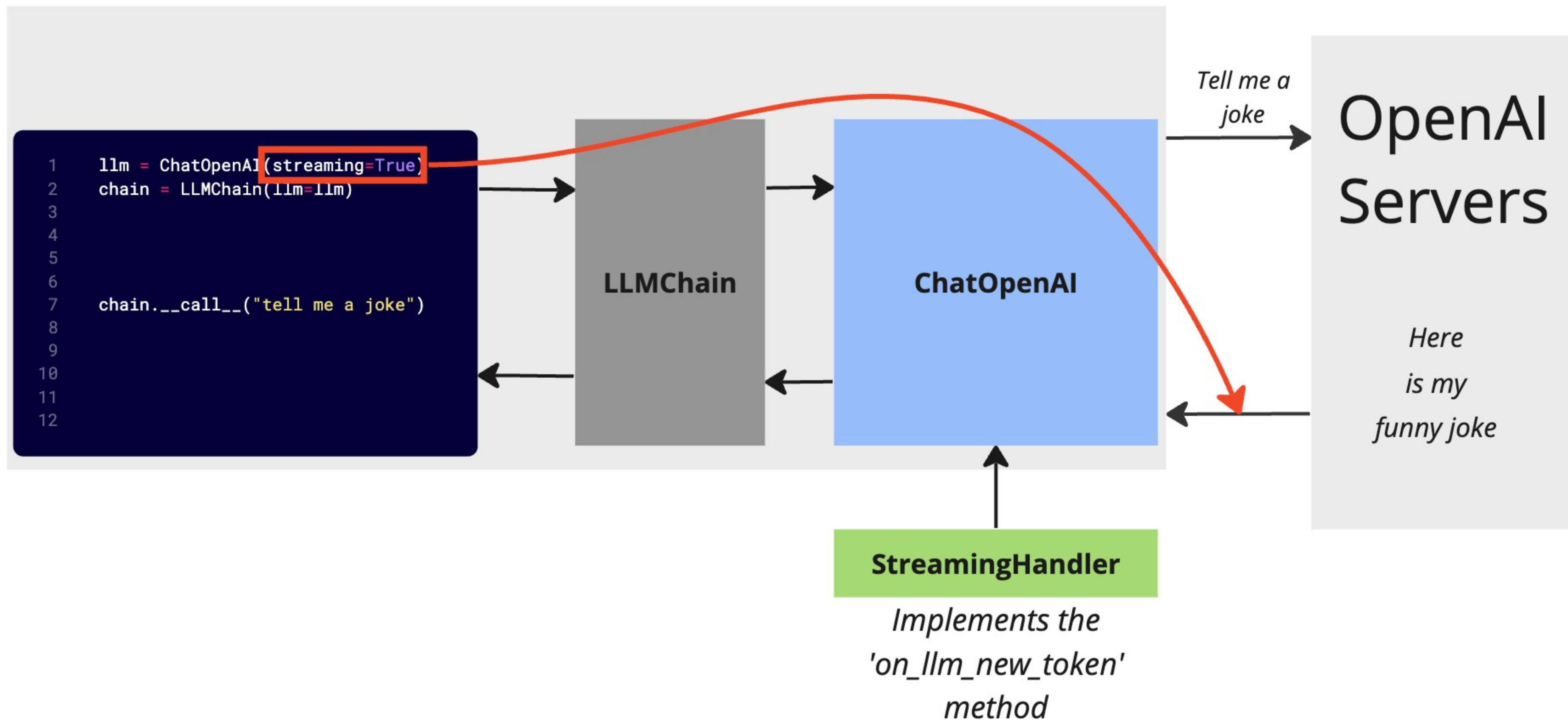
*Here  
is my  
funny joke*

*Tell me a  
joke*

OpenAI  
Servers

**Streaming!**





**Goal:**  
Make these work (just about) the same way

### *Streaming from a **LLM***

```
1 for message in chat.stream(messages):  
2     print(message.content)
```

Sure  
,  
'here  
's  
a  
joke  
for  
you  
:  
  
Why  
don  
't  
scientists  
trust  
atoms  
?

### *Streaming from a **chain***


```
1 input = {"content": "tell me a joke" }  
2 for output in chain.stream(input=input):  
3     print(output)
```

Sure  
,  
'here  
's  
a  
joke  
for  
you  
:  
  
Why  
don  
't  
scientists  
trust  
atoms  
?



## *Streaming from a **chain***

```
1 input = {"content": "tell me a joke" }  
2 for output in chain.stream(input=input):  
3     print(output)
```



```
Sure  
,  
here  
's  
a  
joke  
for  
you  
:  
  
Why  
don  
't  
scientists  
trust  
atoms  
?
```

We need to override the chain's 'stream' method

We need the 'stream' method to return a generator that produces strings

The 'stream' method should run the chain

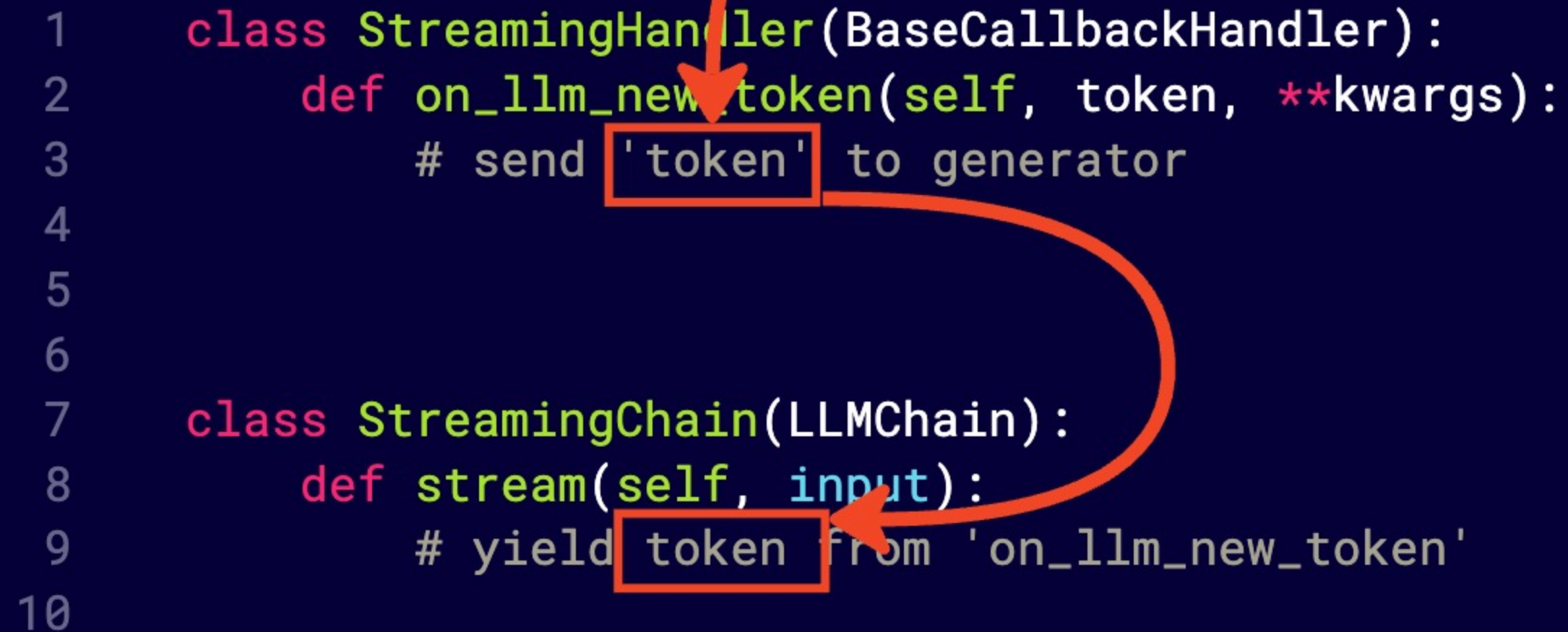
We need to somehow get info from 'on\_llm\_new\_token' into that generator

# OpenAI

*is my  
funny joke*

Here

```
1  class StreamingHandler(BaseCallbackHandler):
2      def on_llm_new_token(self, token, **kwargs):
3          # send 'token' to generator
4
5
6
7  class StreamingChain(LLMChain):
8      def stream(self, input):
9          # yield token from 'on_llm_new_token'
10
```



# OpenAI

*is my  
funny joke*

on\_llm\_new\_token

**queue**

*Here*

**Generator in 'stream' function**



Forever running while loop  
checking for elements in  
queue

```
1 class StreamingChain(LLMChain):
2     def stream(self, input):
3         self(input)
4         while True:
5             token = queue.get()
6             yield token
7
```

LLMChain

*Here  
is my  
funny joke*

ChatOpenAI

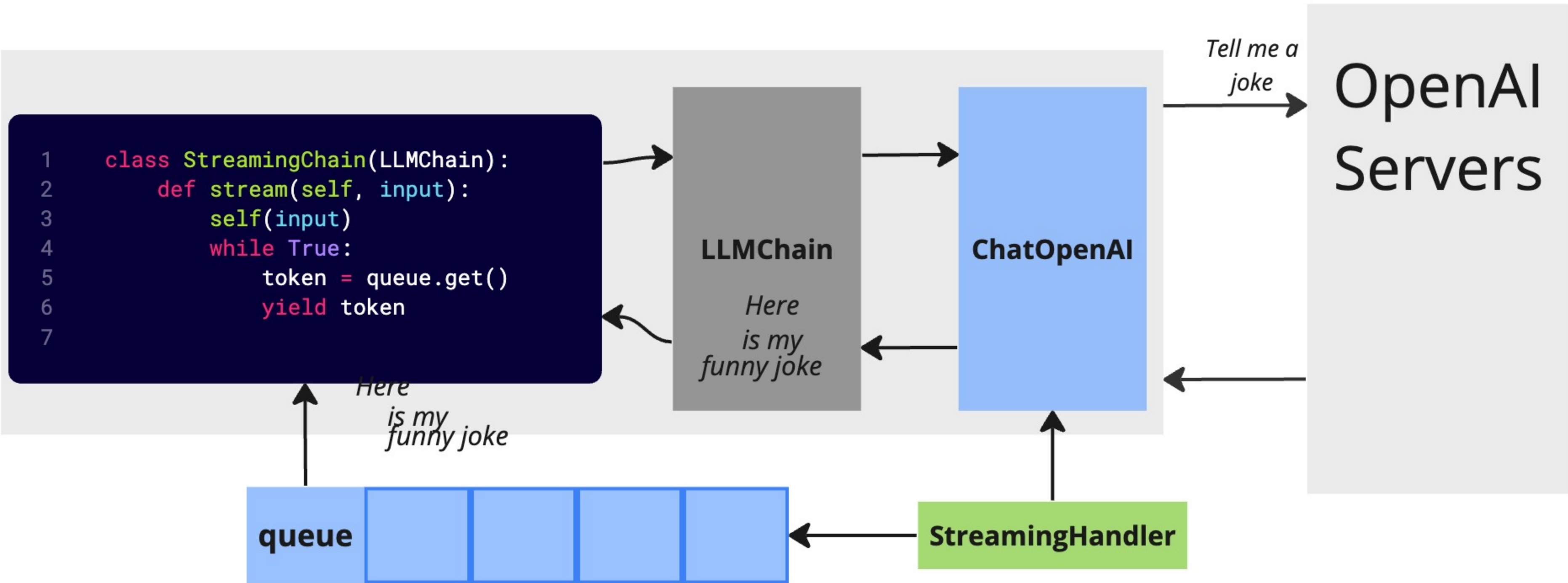
*Tell me a  
joke*

OpenAI  
Servers

*Here  
is my  
funny joke*

queue

StreamingHandler





*Main  
Thread*

```
1  def stream(self, input):  
2      def task():  
3          self(input)  
4  
5      Thread(target=task)  
6  
7      while True:  
8          token = queue.get()  
9          yield token
```

*Here*

**queue**

**StreamingHandler**

*New thread  
made just to run  
the chain*

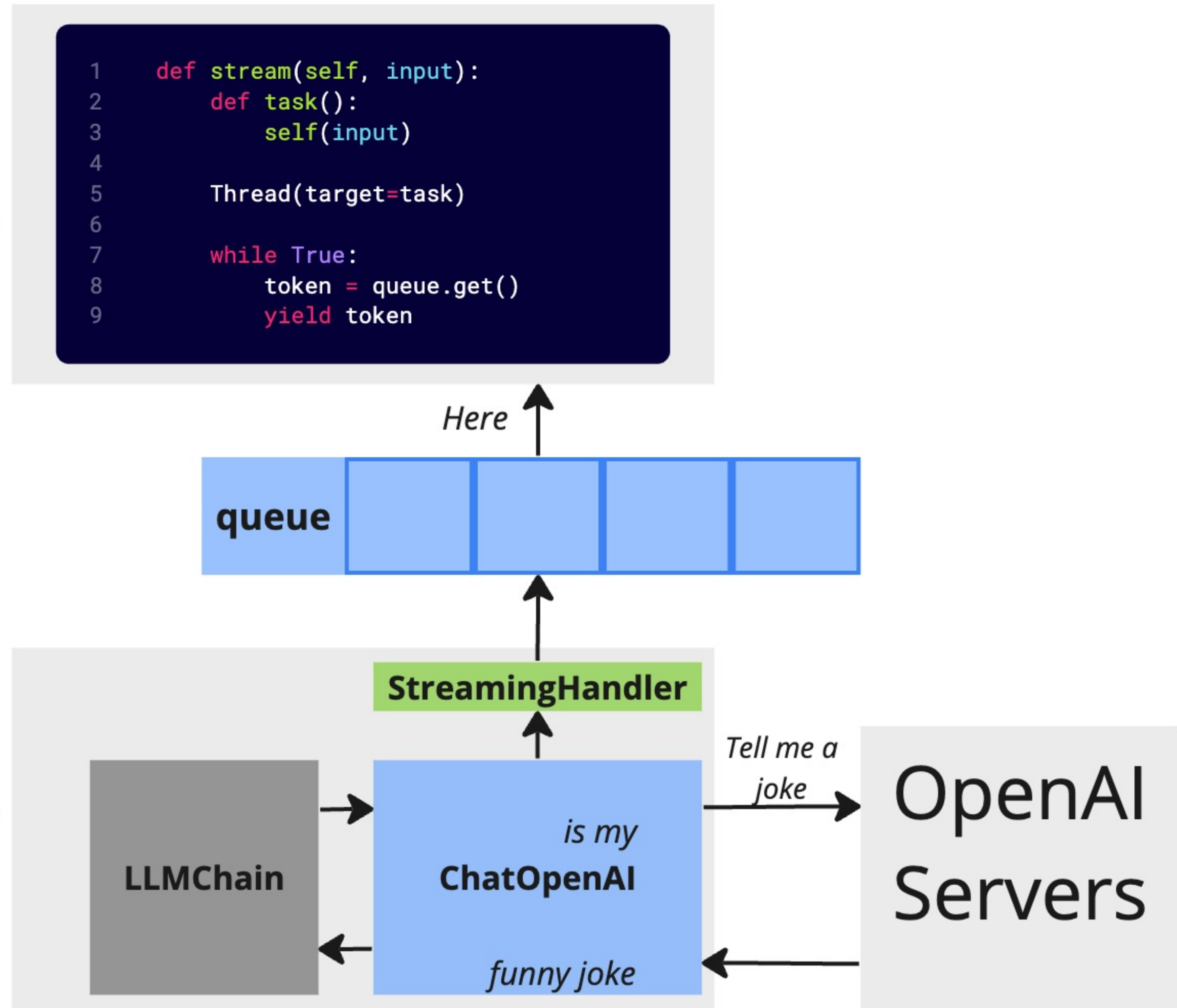
**LLMChain**

**ChatOpenAI**

*Tell me a  
joke*

**OpenAI  
Servers**

*funny joke*



```
1 def stream(self, input):
2     def task():
3         self(input)
4
5     Thread(target=task)
6
7     while True:
8         token = queue.get()
9         yield token
```

*Here is my funny joke*

