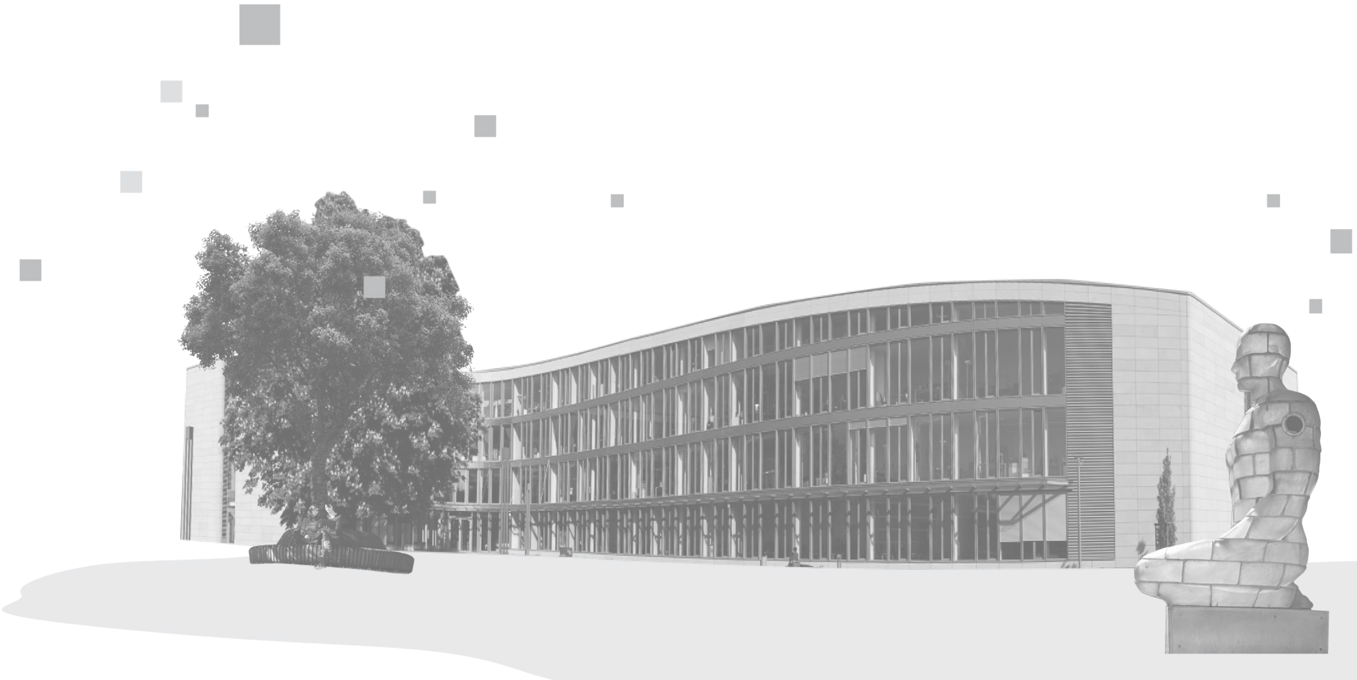
HPI Hasso Plattner Institut
Digital Engineering · Universität Potsdam

LLaMa

Example Generation

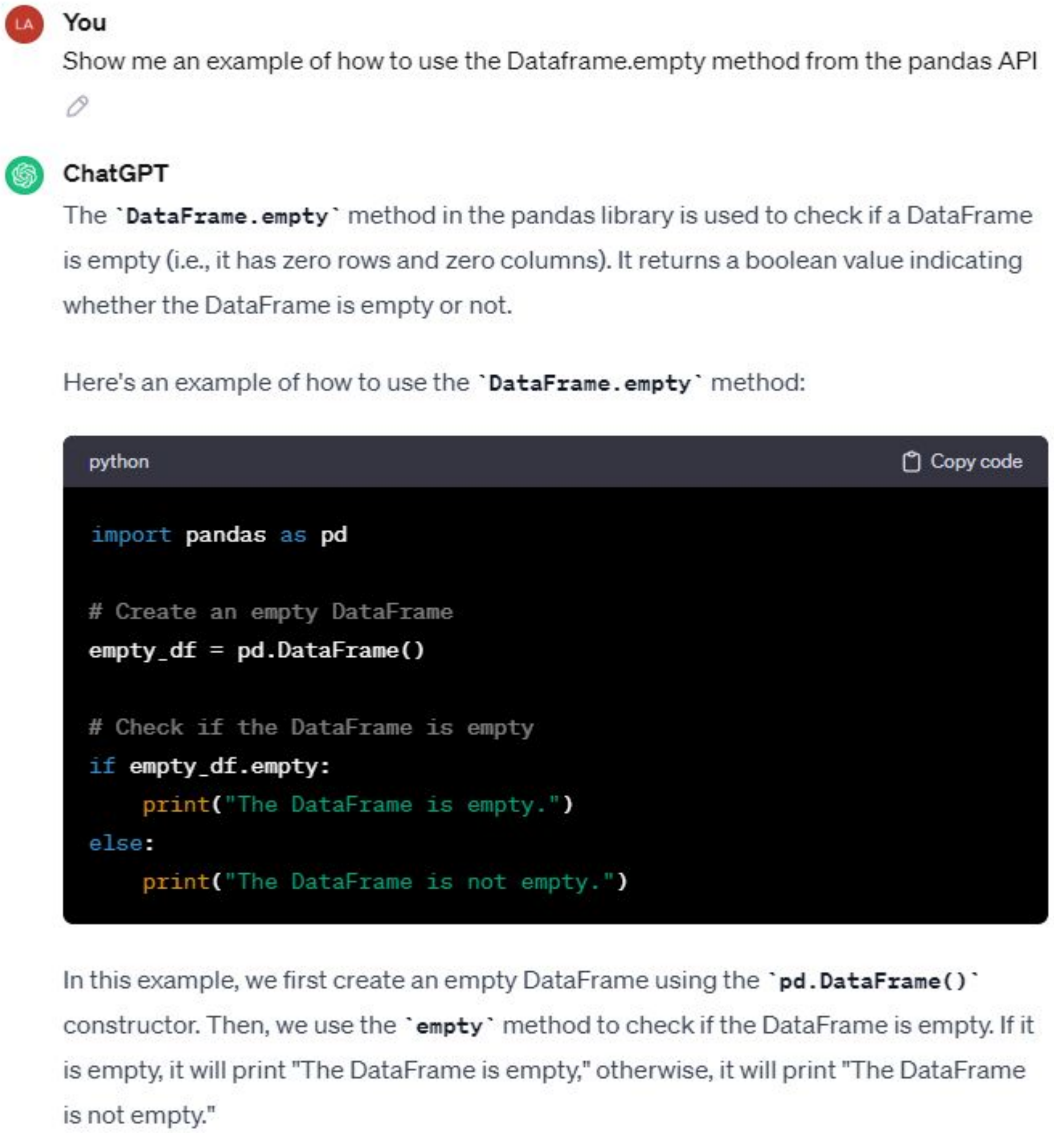
Future of Programming
WiSe 2023 / 2024
Software Architecture Group

Christian Raue / Lara Kursawe



Design IT.
Create Knowledge.

Question: Can LLMs generate helpful examples of API interfaces?



You

Show me an example of how to use the Dataframe.empty method from the pandas API

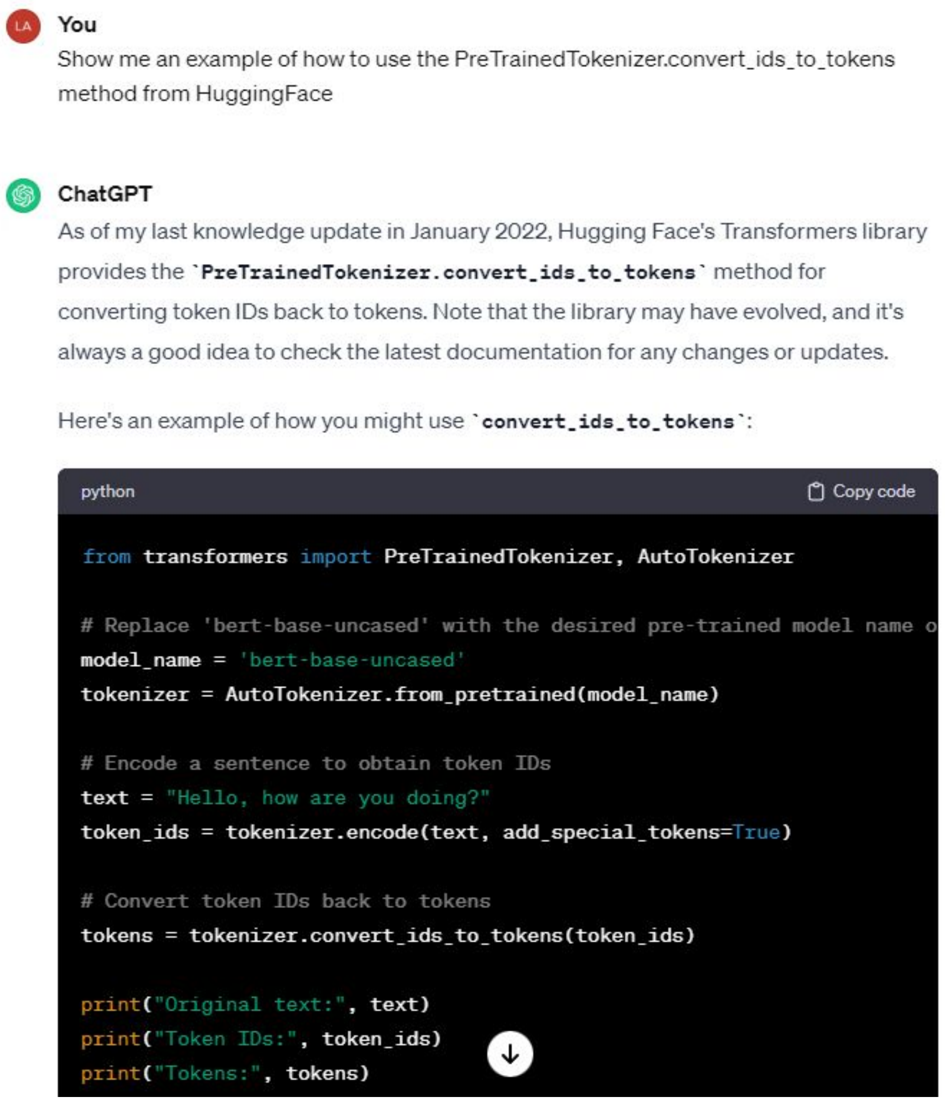
ChatGPT

The `DataFrame.empty` method in the pandas library is used to check if a DataFrame is empty (i.e., it has zero rows and zero columns). It returns a boolean value indicating whether the DataFrame is empty or not.

Here's an example of how to use the `DataFrame.empty` method:

```python
import pandas as pd

# Create an empty DataFrame
empty_df = pd.DataFrame()

# Check if the DataFrame is empty
if empty_df.empty:
    print("The DataFrame is empty.")
else:
    print("The DataFrame is not empty.")
```

In this example, we first create an empty DataFrame using the `pd.DataFrame()` constructor. Then, we use the `empty` method to check if the DataFrame is empty. If it is empty, it will print "The DataFrame is empty," otherwise, it will print "The DataFrame is not empty."



You

Show me an example of how to use the PreTrainedTokenizer.convert_ids_to_tokens method from HuggingFace

ChatGPT

As of my last knowledge update in January 2022, Hugging Face's Transformers library provides the `PreTrainedTokenizer.convert_ids_to_tokens` method for converting token IDs back to tokens. Note that the library may have evolved, and it's always a good idea to check the latest documentation for any changes or updates.

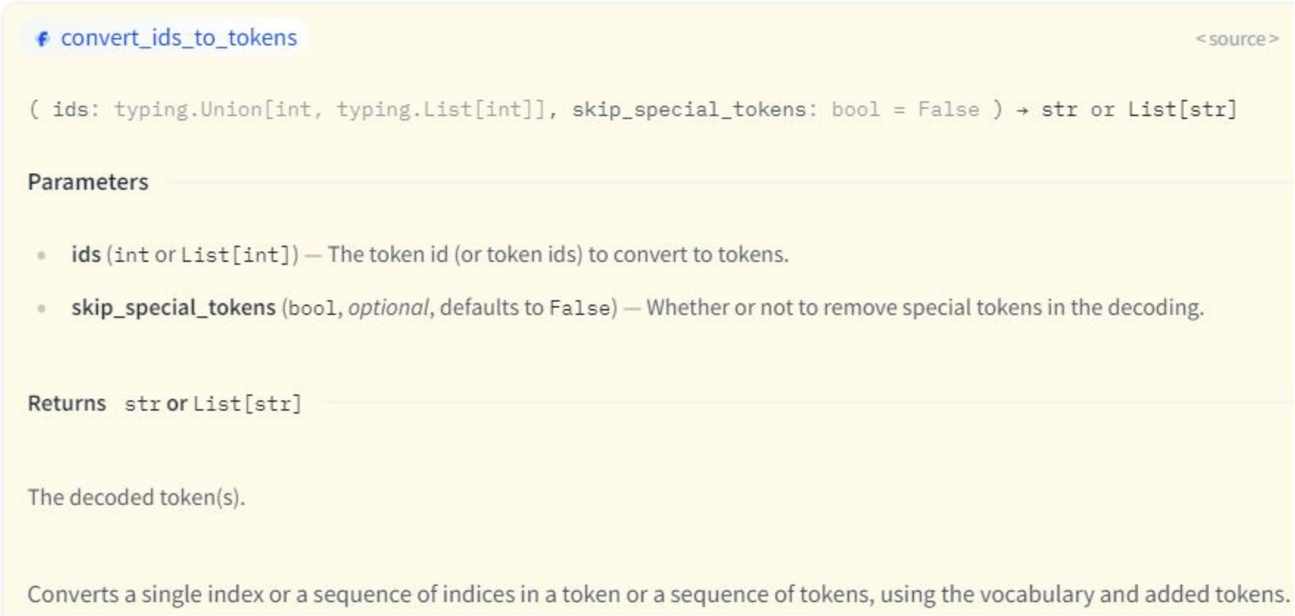
Here's an example of how you might use `convert_ids_to_tokens`:

```python
from transformers import PreTrainedTokenizer, AutoTokenizer

# Replace 'bert-base-uncased' with the desired pre-trained model name o
model_name = 'bert-base-uncased'
tokenizer = AutoTokenizer.from_pretrained(model_name)

# Encode a sentence to obtain token IDs
text = "Hello, how are you doing?"
token_ids = tokenizer.encode(text, add_special_tokens=True)

# Convert token IDs back to tokens
tokens = tokenizer.convert_ids_to_tokens(token_ids)

print("Original text:", text)
print("Token IDs:", token_ids)
print("Tokens:", tokens)
```

Question: Can LLMs generate helpful examples of API interfaces?

Yes!

Thank you for your attention!



convert_ids_to_tokens <source>

`( ids: typing.Union[int, typing.List[int]], skip_special_tokens: bool = False ) → str or List[str]`

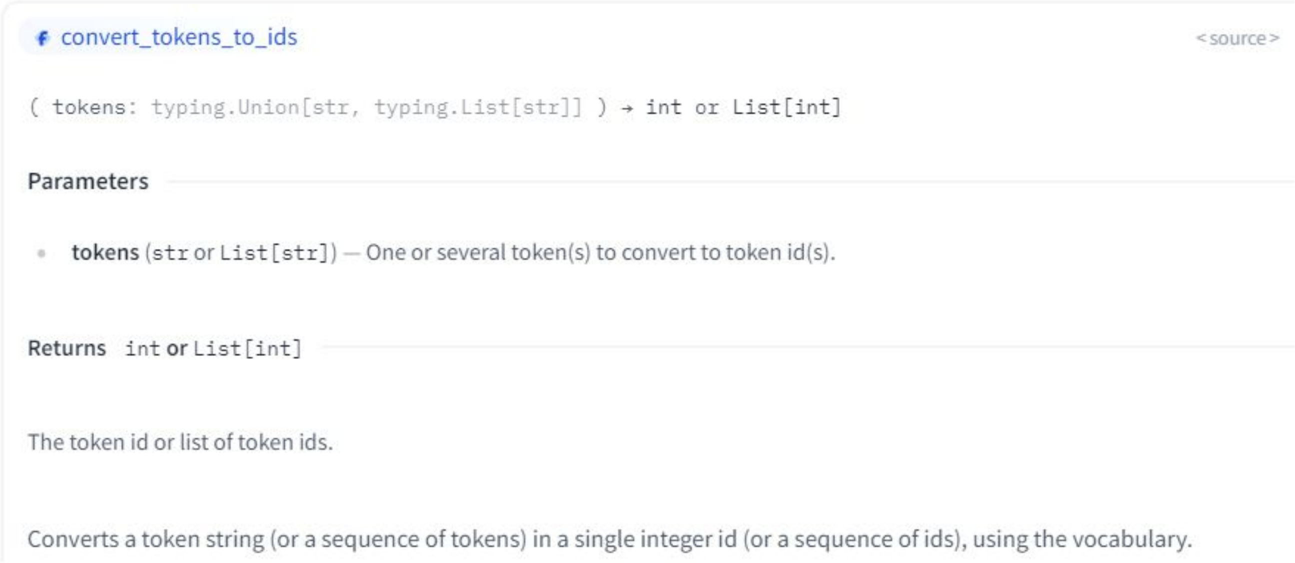
Parameters

- **ids** (`int` or `List[int]`) — The token id (or token ids) to convert to tokens.
- **skip_special_tokens** (`bool`, *optional*, defaults to `False`) — Whether or not to remove special tokens in the decoding.

Returns `str` **or** `List[str]`

The decoded token(s).

Converts a single index or a sequence of indices in a token or a sequence of tokens, using the vocabulary and added tokens.



convert_tokens_to_ids <source>

`( tokens: typing.Union[str, typing.List[str]] ) → int or List[int]`

Parameters

- **tokens** (`str` or `List[str]`) — One or several token(s) to convert to token id(s).

Returns `int` **or** `List[int]`

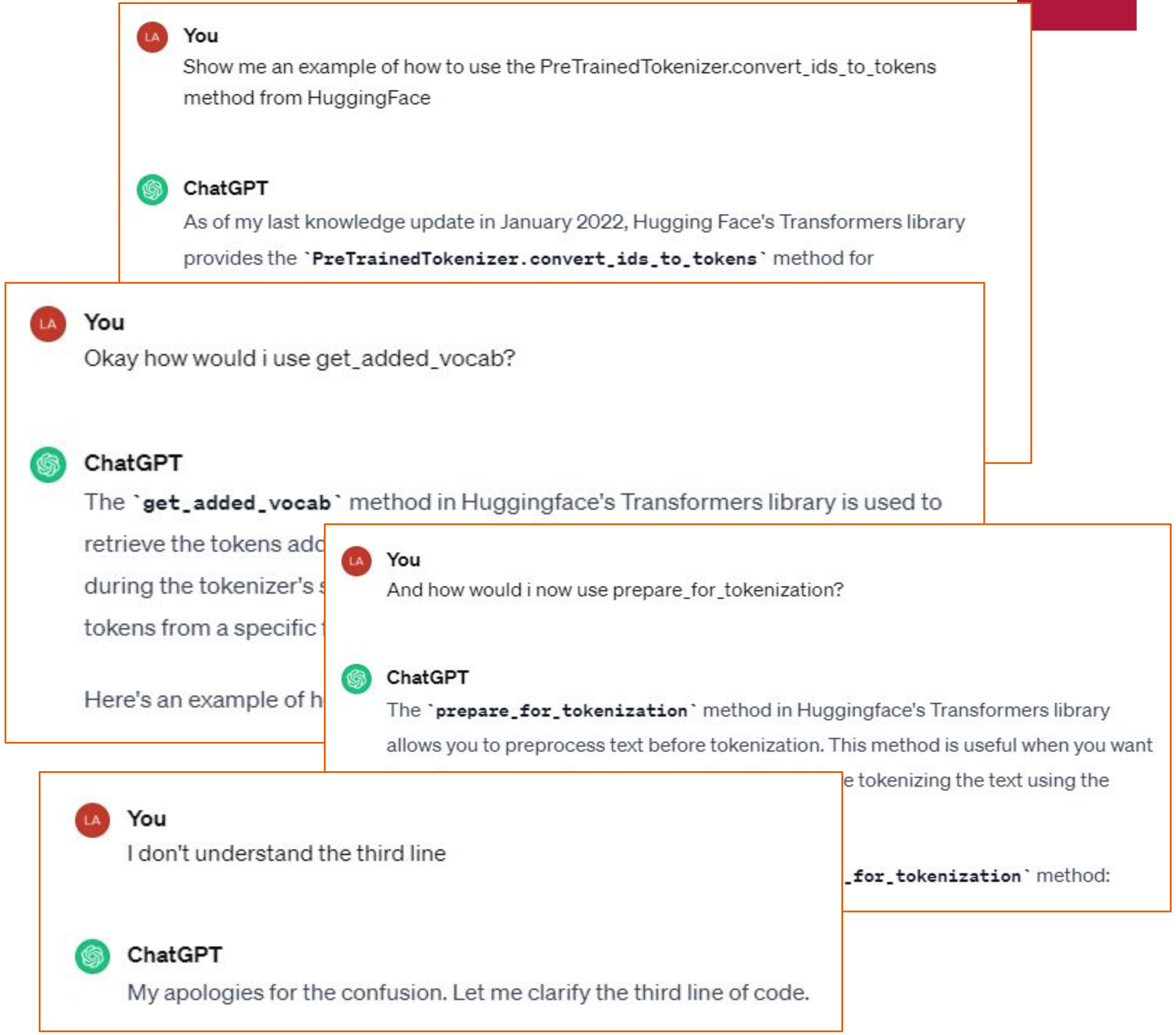
The token id or list of token ids.

Converts a token string (or a sequence of tokens) in a single integer id (or a sequence of ids), using the vocabulary.

Idea

Chatbots

- Helpful when working and learning new tools, but often cumbersome to use
    - Might need long prompts
    - First results may be unhelpful
    - Multiple follow up prompts can be annoying
- Different LLMs give different results (of different quality)



You
Show me an example of how to use the PreTrainedTokenizer.convert_ids_to_tokens method from HuggingFace

ChatGPT
As of my last knowledge update in January 2022, Hugging Face's Transformers library provides the `PreTrainedTokenizer.convert_ids_to_tokens` method for

You
Okay how would i use get_added_vocab?

ChatGPT
The `get_added_vocab` method in Huggingface's Transformers library is used to retrieve the tokens add
during the tokenizer's s
tokens from a specific t

Here's an example of h

You
And how would i now use prepare_for_tokenization?

ChatGPT
The `prepare_for_tokenization` method in Huggingface's Transformers library allows you to preprocess text before tokenization. This method is useful when you want
e tokenizing the text using the

_for_tokenization` method:

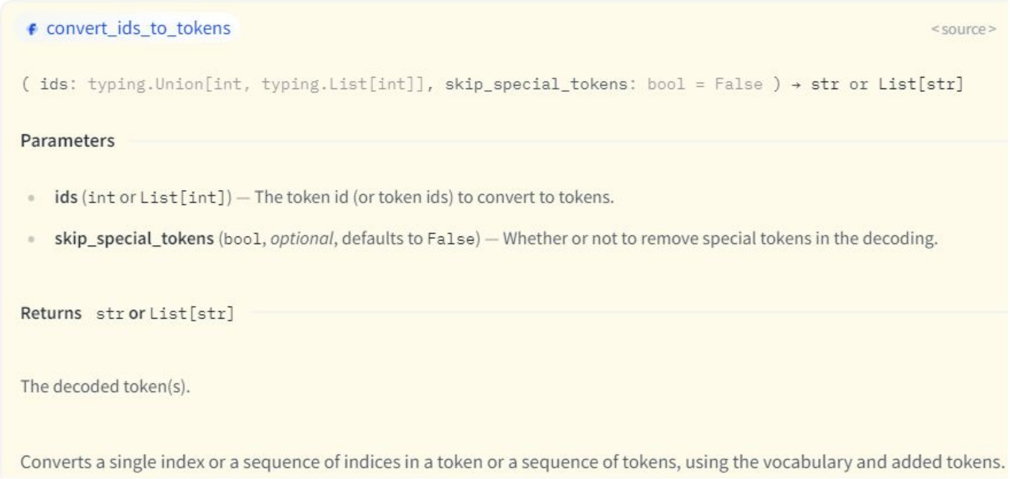
You
I don't understand the third line

ChatGPT
My apologies for the confusion. Let me clarify the third line of code.

Idea

Chatbots

- Helpful when working and learning new tools, but often cumbersome to use
    - Might need long prompts
    - First results may be unhelpful
    - Multiple follow up prompts can be annoying
- Different LLMs give different results (of different quality)



convert_ids_to_tokens

`( ids: typing.Union[int, typing.List[int]], skip_special_tokens: bool = False ) → str or List[str]`

Parameters

- **ids** (`int` or `List[int]`) — The token id (or token ids) to convert to tokens.
- **skip_special_tokens** (`bool`, *optional*, defaults to `False`) — Whether or not to remove special tokens in the decoding.

Returns `str` **or** `List[str]`

The decoded token(s).

Converts a single index or a sequence of indices in a token or a sequence of tokens, using the vocabulary and added tokens.

APIs

- Some API references are very detailed and helpful, others are not

xLLaMa

DEMO

Who would use xLLaMa?

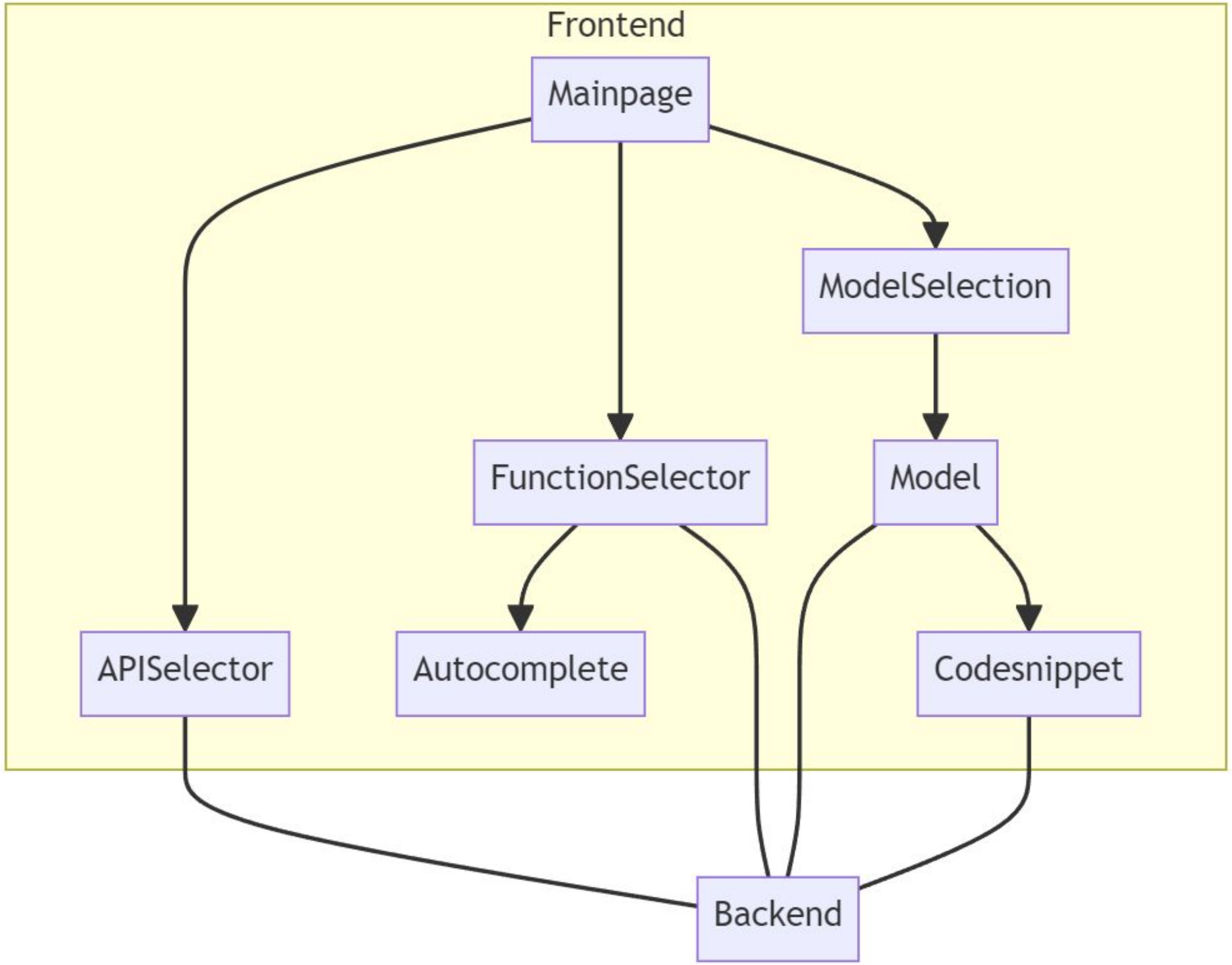
Students and Software Developers who…

- … are new to one of the frameworks / APIs
- … find themselves often wondering “wait, how does this work again…?”
- … want quick results, without having to type a lot to an LLM
- … want multiple different results to choose from
- … use multiple different IDEs and source-code editors, or editors without integrated AI assistants

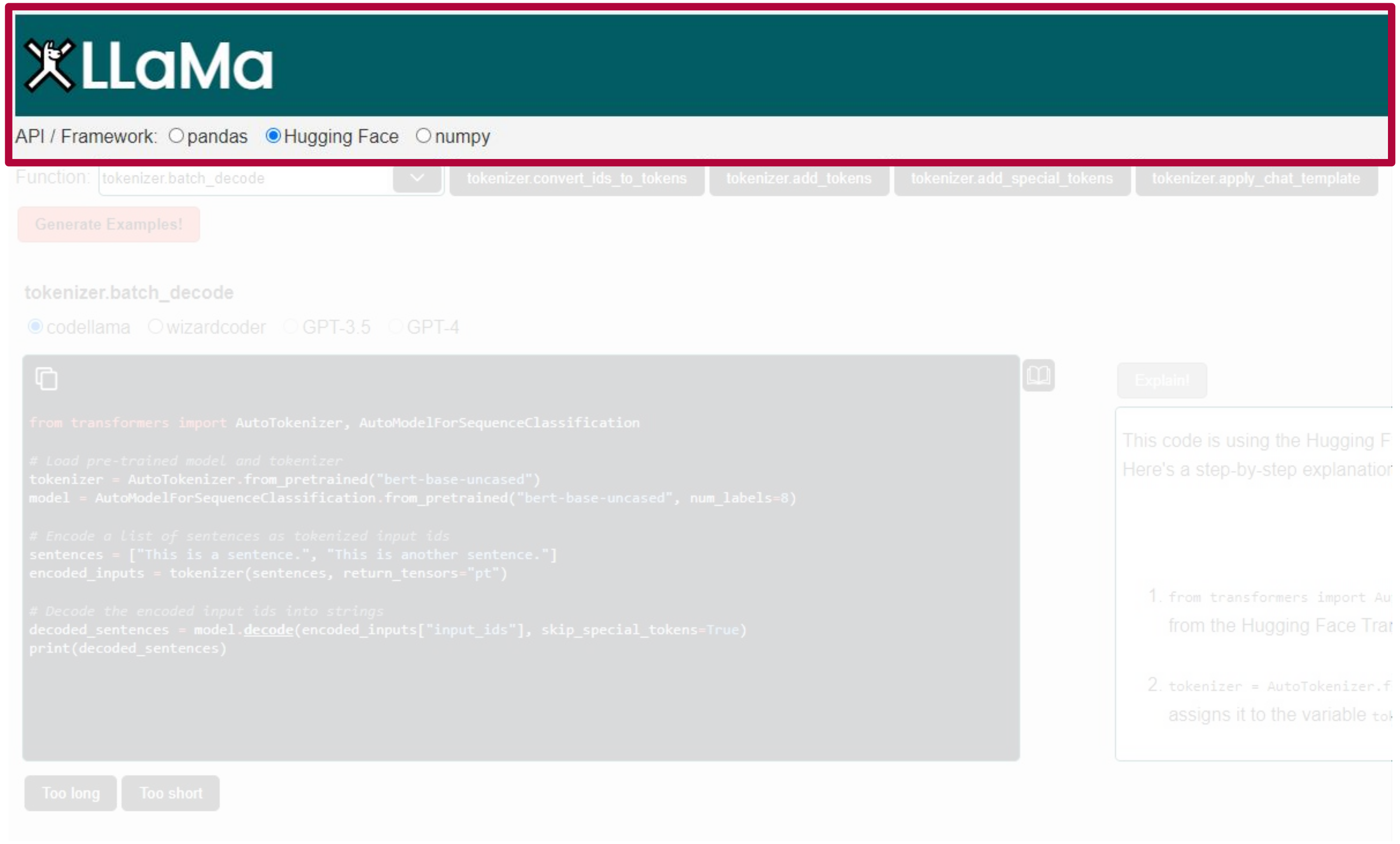
Architecture - Frontend

Vue.js

- Builds on top of standard HTML, CSS, and JavaScript
- Component-based programming model



Architecture - Frontend


xLLaMa
tokenizer.convert_ids_to_tokens
tokenizer.add_tokens
tokenizer.add_special_tokens
tokenizer.apply_chat_template
Generate Examples!
Explain!
Too long
Too short

API / Framework: ○ pandas ● Hugging Face ○ numpy

Function: tokenizer.batch_decode

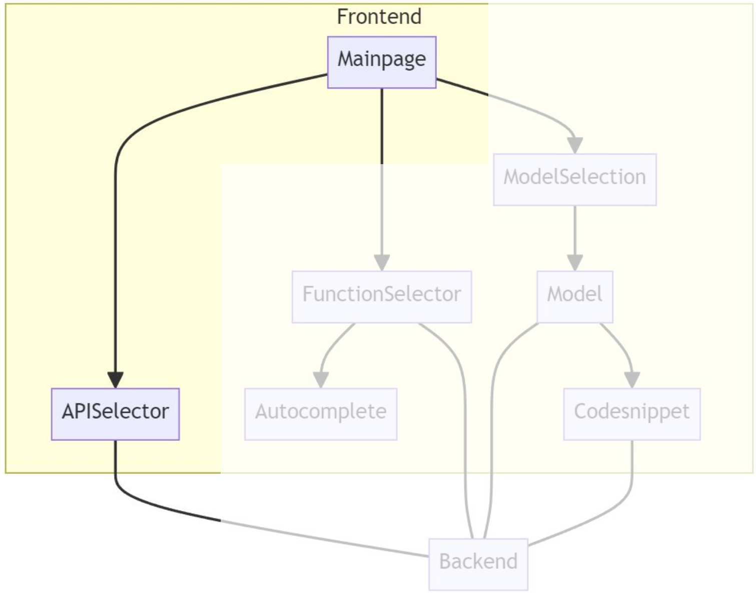
tokenizer.batch_decode

● codellama ○ wizardcoder ○ GPT-3.5 ○ GPT-4

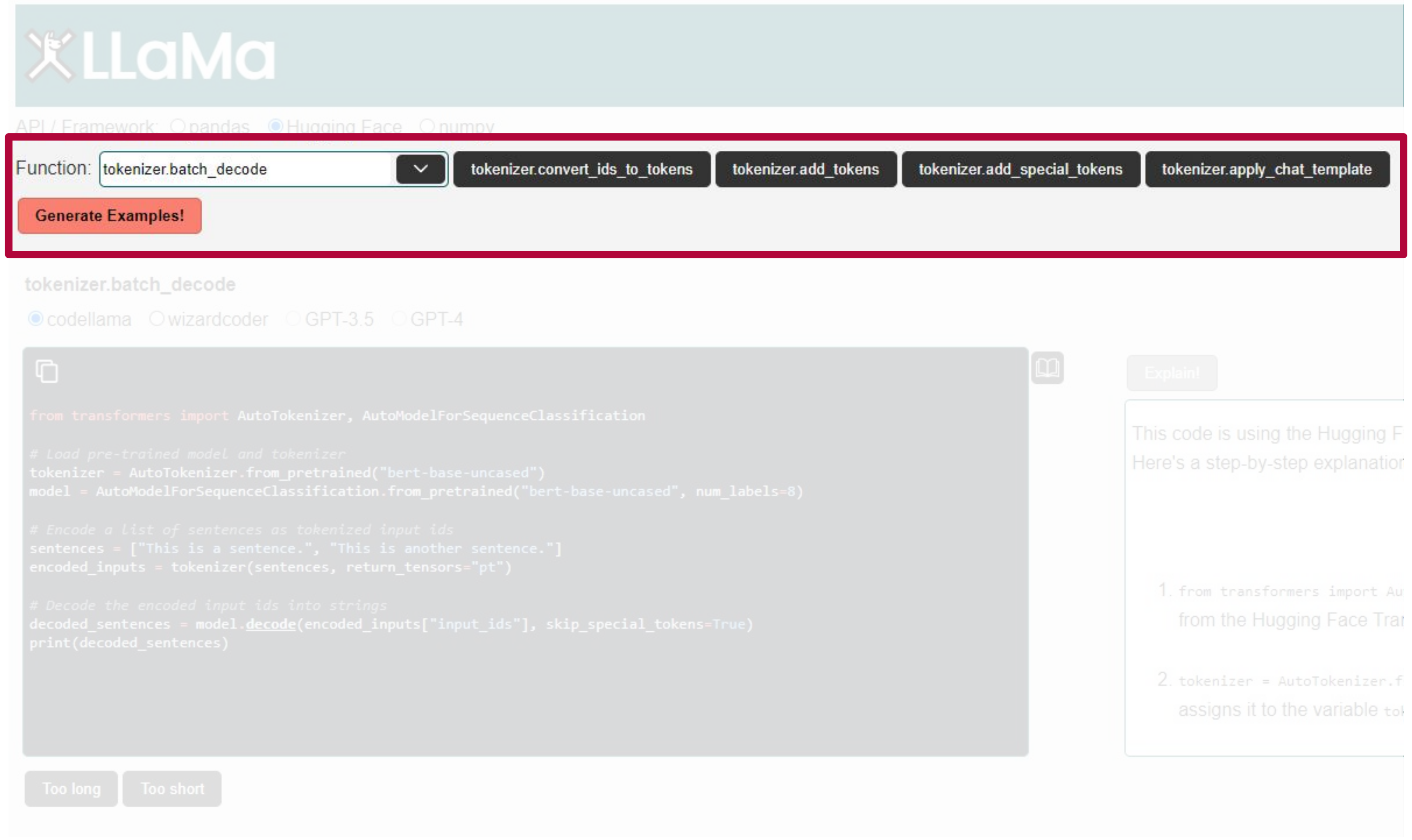
```
from transformers import AutoTokenizer, AutoModelForSequenceClassification

# Load pre-trained model and tokenizer
tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
model = AutoModelForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=8)

# Encode a list of sentences as tokenized input ids
sentences = ["This is a sentence.", "This is another sentence."]
encoded_inputs = tokenizer(sentences, return_tensors="pt")

# Decode the encoded input ids into strings
decoded_sentences = model.decode(encoded_inputs["input_ids"], skip_special_tokens=True)
print(decoded_sentences)
```

This code is using the Hugging F
Here's a step-by-step explanatio

1. `from transformers import Au`
   from the Hugging Face Tra
2. `tokenizer = AutoTokenizer.f`
   assigns it to the variable `to`



Architecture - Frontend



API / Framework: ○ pandas ● Hugging Face ○ numpy

Function: tokenizer.batch_decode

tokenizer.convert_ids_to_tokens
tokenizer.add_tokens
tokenizer.add_special_tokens
tokenizer.apply_chat_template
Generate Examples!

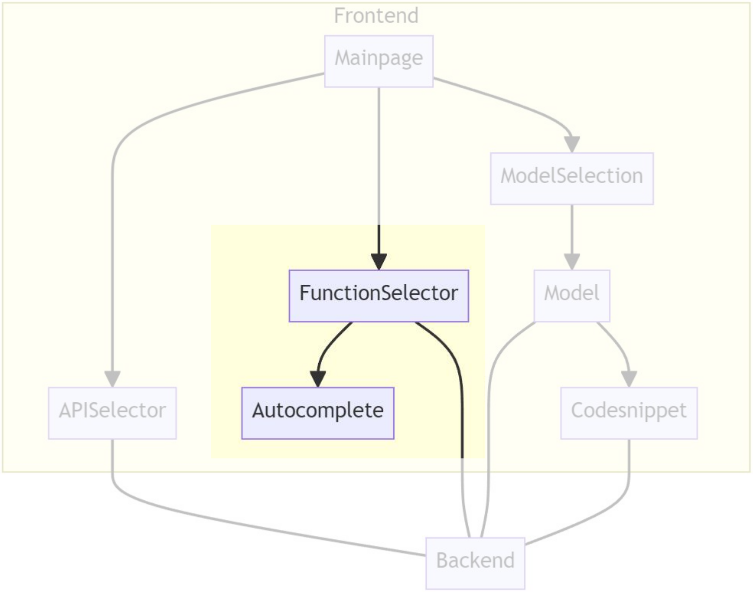
tokenizer.batch_decode

● codellama ○ wizardcoder ○ GPT-3.5 ○ GPT-4

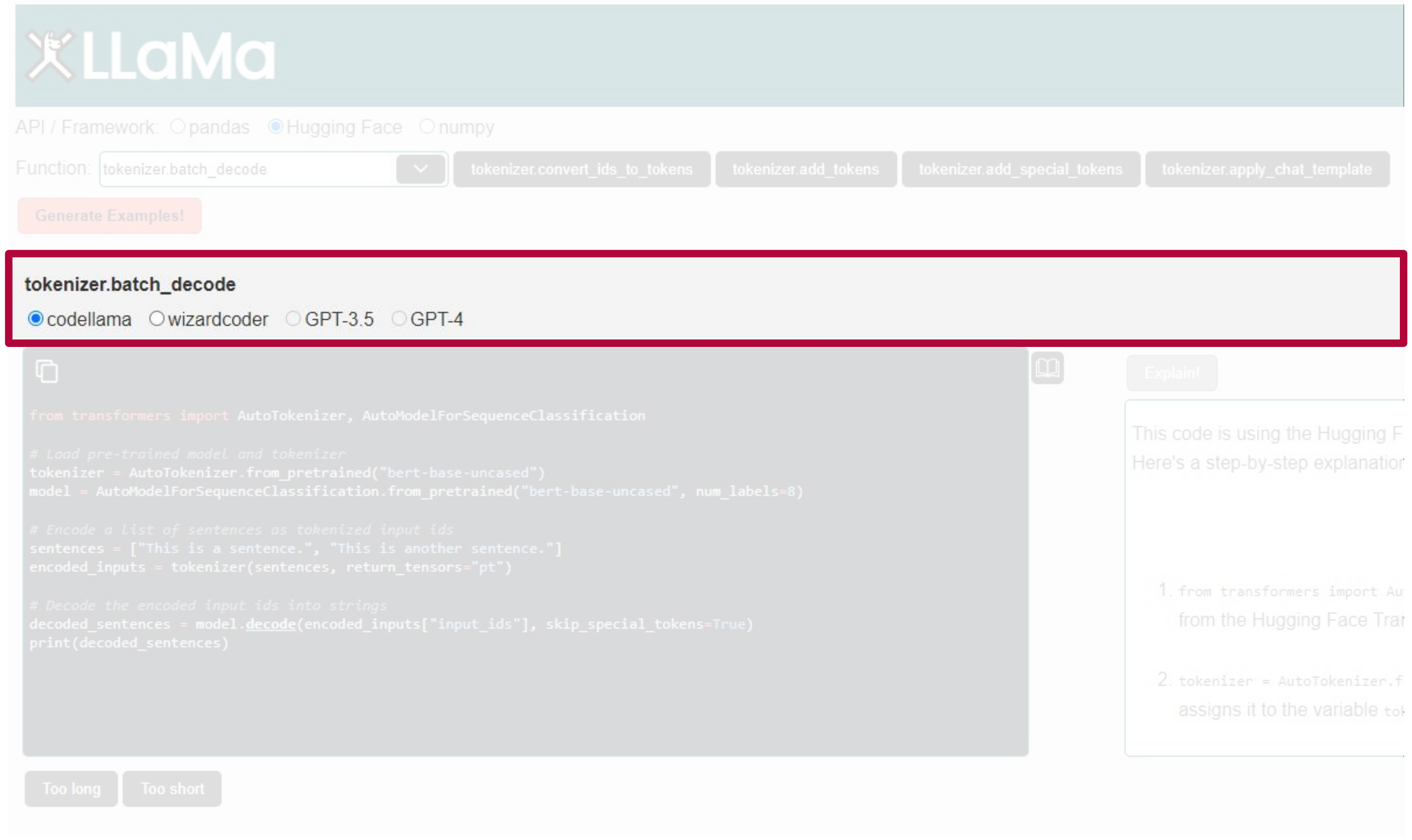
```python
from transformers import AutoTokenizer, AutoModelForSequenceClassification

# Load pre-trained model and tokenizer
tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
model = AutoModelForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=8)

# Encode a list of sentences as tokenized input ids
sentences = ["This is a sentence.", "This is another sentence."]
encoded_inputs = tokenizer(sentences, return_tensors="pt")

# Decode the encoded input ids into strings
decoded_sentences = model.decode(encoded_inputs["input_ids"], skip_special_tokens=True)
print(decoded_sentences)
```

This code is using the Hugging F
Here's a step-by-step explanatio

1. `from transformers import Au`
   from the Hugging Face Tra
2. `tokenizer = AutoTokenizer.f`
   assigns it to the variable `to`



Architecture - Frontend


xLLaMa
API / Framework: ○ pandas ● Hugging Face ○ numpy
Function: tokenizer.batch_decode
tokenizer.convert_ids_to_tokens
tokenizer.add_tokens
tokenizer.add_special_tokens
tokenizer.apply_chat_template
Generate Examples!

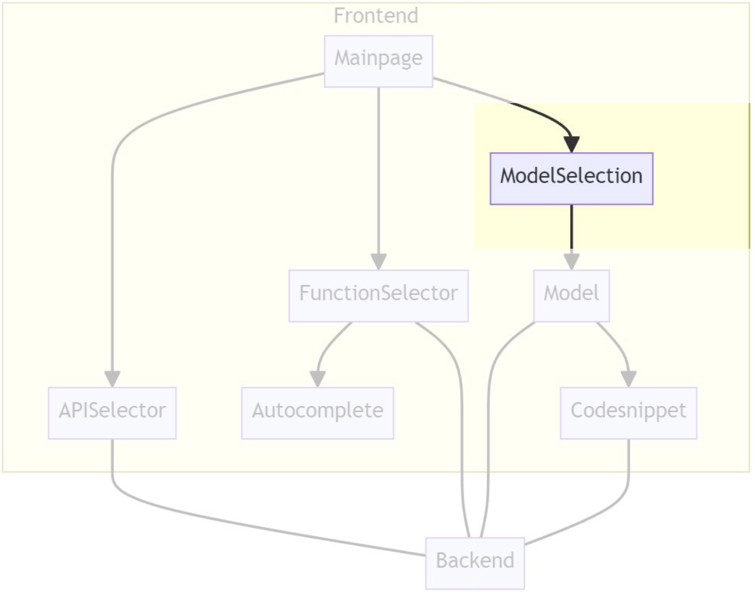
tokenizer.batch_decode

● codellama ○ wizardcoder ○ GPT-3.5 ○ GPT-4

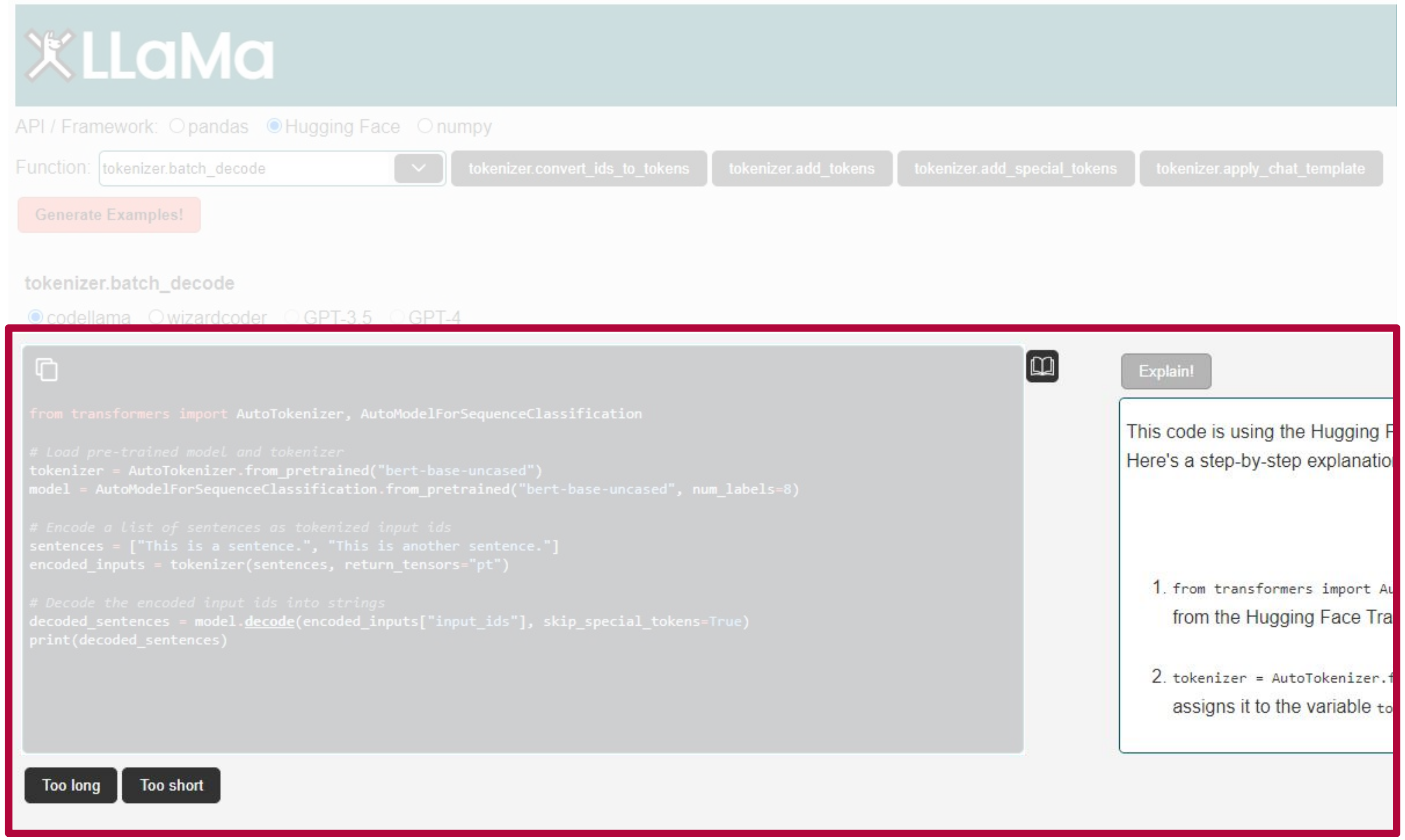
```
from transformers import AutoTokenizer, AutoModelForSequenceClassification

# Load pre-trained model and tokenizer
tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
model = AutoModelForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=8)

# Encode a list of sentences as tokenized input ids
sentences = ["This is a sentence.", "This is another sentence."]
encoded_inputs = tokenizer(sentences, return_tensors="pt")

# Decode the encoded input ids into strings
decoded_sentences = model.decode(encoded_inputs["input_ids"], skip_special_tokens=True)
print(decoded_sentences)
```

This code is using the Hugging F
Here's a step-by-step explanatio

1. `from transformers import Au`
   from the Hugging Face Tra
2. `tokenizer = AutoTokenizer.f`
   assigns it to the variable `to`



Architecture - Frontend


LLaMa
tokenizer.convert_ids_to_tokens
tokenizer.add_tokens
tokenizer.add_special_tokens
tokenizer.apply_chat_template
Generate Examples!
Explain!
Too long
Too short

API / Framework: ○ pandas ● Hugging Face ○ numpy

Function: tokenizer.batch_decode

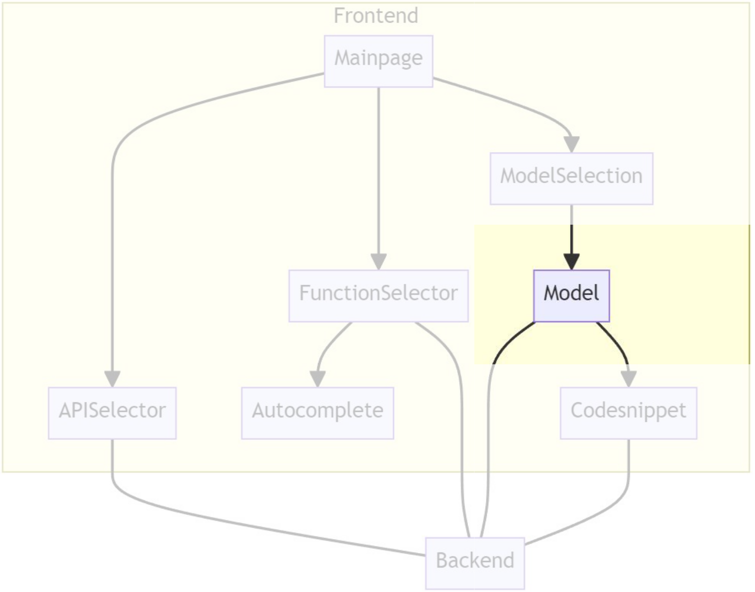
tokenizer.batch_decode

● codellama ○ wizardcoder ○ GPT-3.5 ○ GPT-4

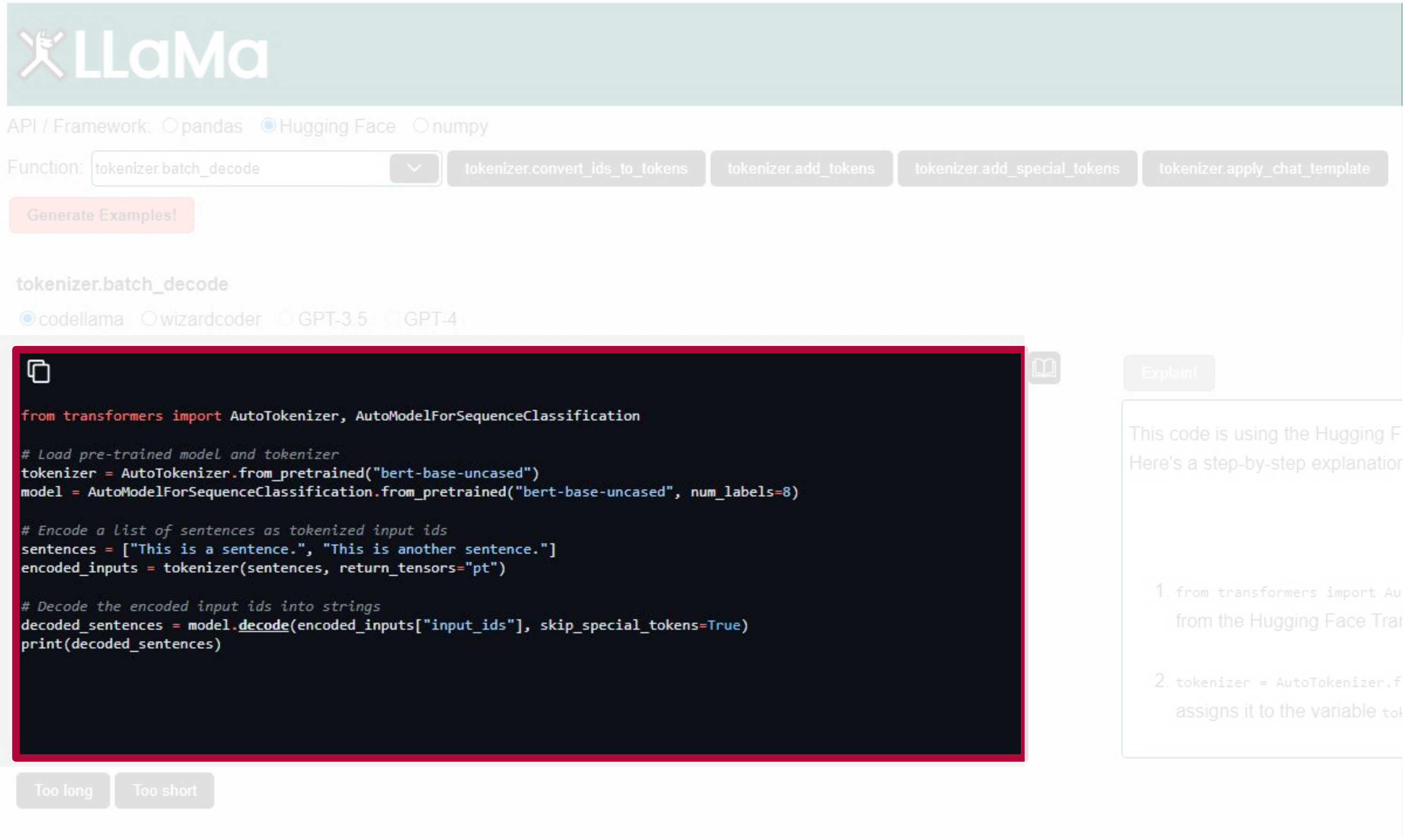
```python
from transformers import AutoTokenizer, AutoModelForSequenceClassification

# Load pre-trained model and tokenizer
tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
model = AutoModelForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=8)

# Encode a list of sentences as tokenized input ids
sentences = ["This is a sentence.", "This is another sentence."]
encoded_inputs = tokenizer(sentences, return_tensors="pt")

# Decode the encoded input ids into strings
decoded_sentences = model.decode(encoded_inputs["input_ids"], skip_special_tokens=True)
print(decoded_sentences)
```

This code is using the Hugging F
Here's a step-by-step explanatio

1. `from transformers import Au` from the Hugging Face Tra
2. `tokenizer = AutoTokenizer.f` assigns it to the variable `to`



Architecture - Frontend


LLaMa
Generate Examples!
tokenizer.convert_ids_to_tokens
tokenizer.add_tokens
tokenizer.add_special_tokens
tokenizer.apply_chat_template
Explain!
Too long
Too short

API / Framework: ○ pandas ● Hugging Face ○ numpy

Function: tokenizer.batch_decode

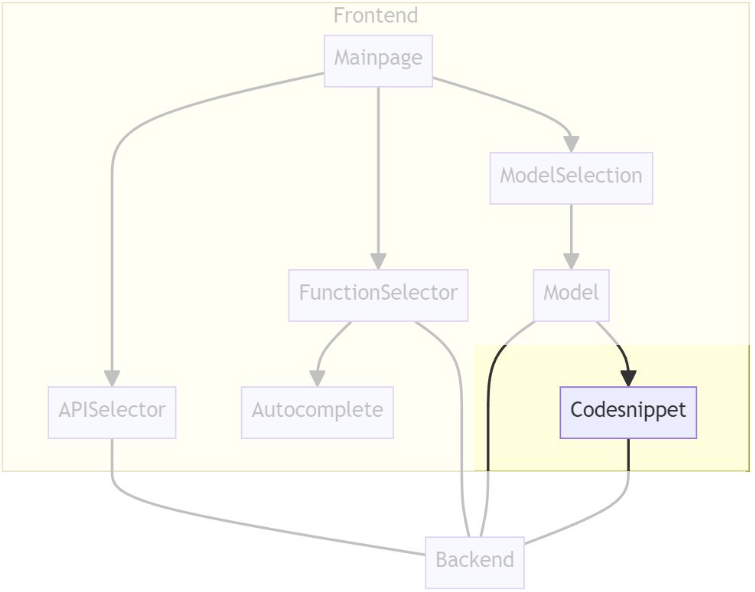
tokenizer.batch_decode

● codellama ○ wizardcoder ○ GPT-3.5 ○ GPT-4

```python
from transformers import AutoTokenizer, AutoModelForSequenceClassification

# Load pre-trained model and tokenizer
tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
model = AutoModelForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=8)

# Encode a list of sentences as tokenized input ids
sentences = ["This is a sentence.", "This is another sentence."]
encoded_inputs = tokenizer(sentences, return_tensors="pt")

# Decode the encoded input ids into strings
decoded_sentences = model.decode(encoded_inputs["input_ids"], skip_special_tokens=True)
print(decoded_sentences)
```

This code is using the Hugging F
Here's a step-by-step explanatio

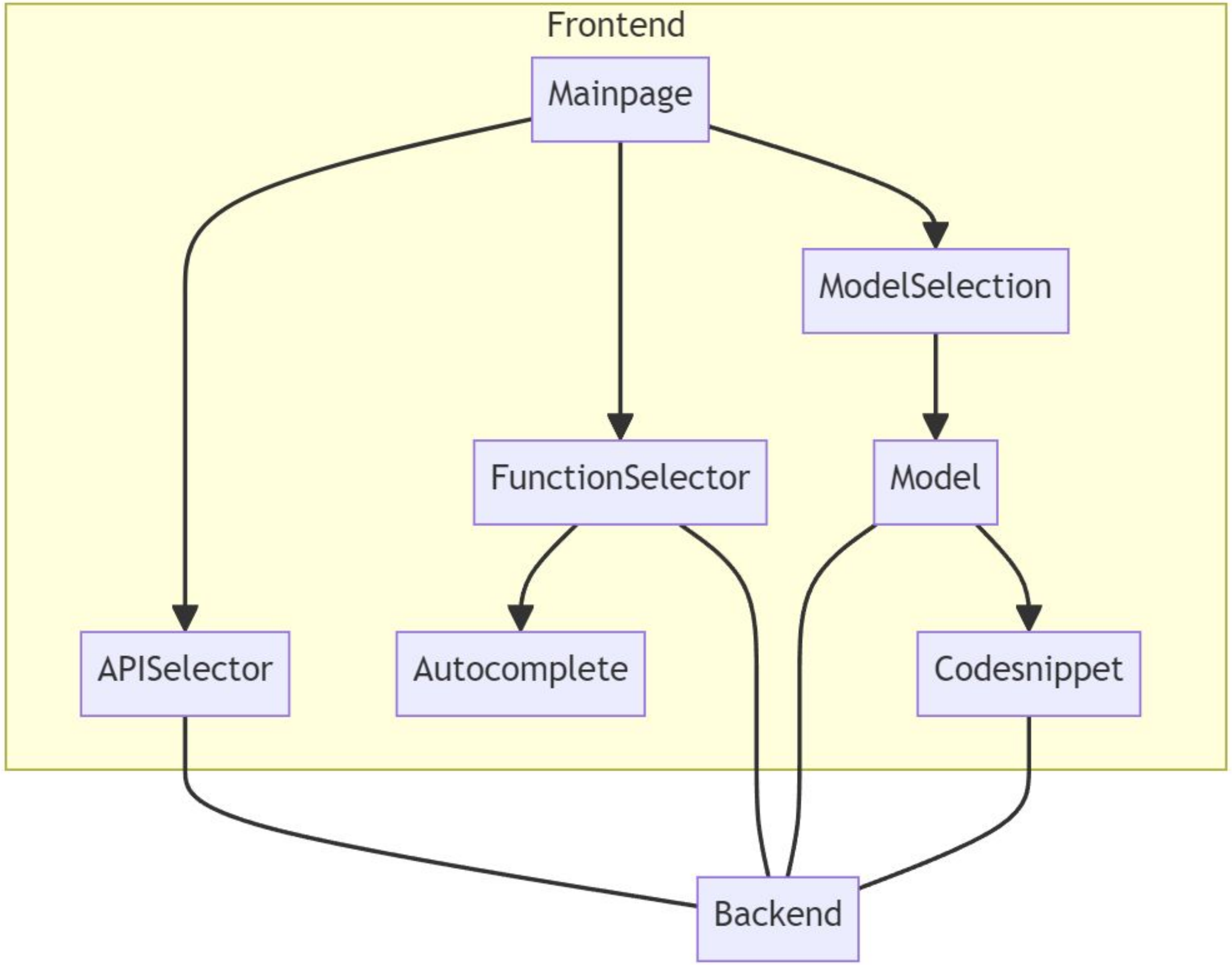
1. `from transformers import Au`
from the Hugging Face Trai
2. `tokenizer = AutoTokenizer.f`
assigns it to the variable `to`



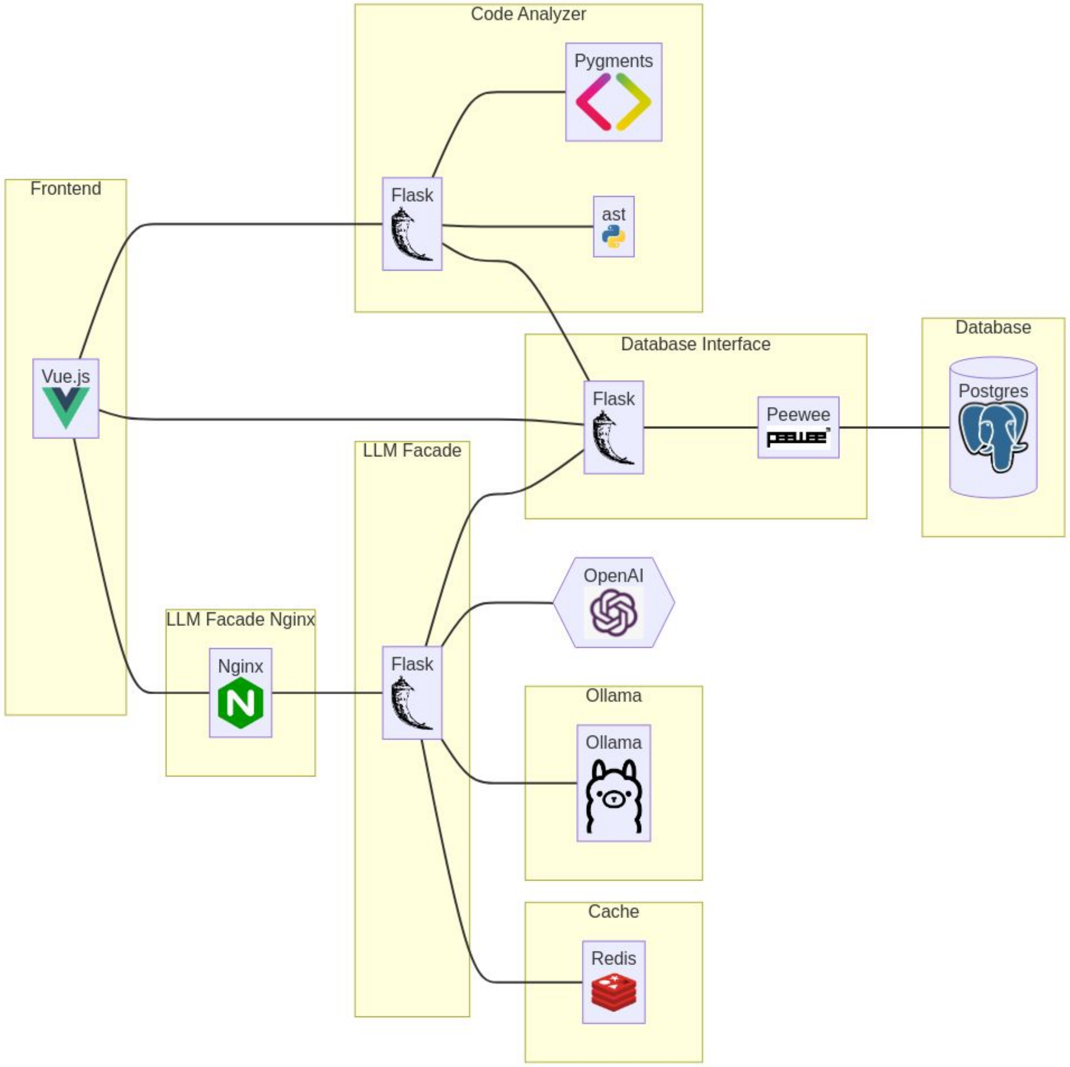
Architecture - Frontend

Vue.js

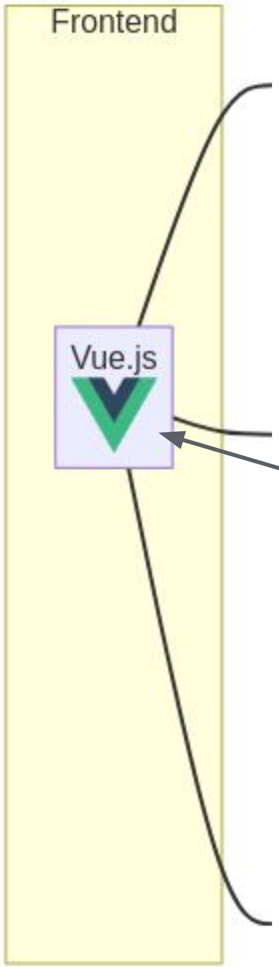
- Builds on top of standard HTML, CSS, and JavaScript
- Component-based programming model



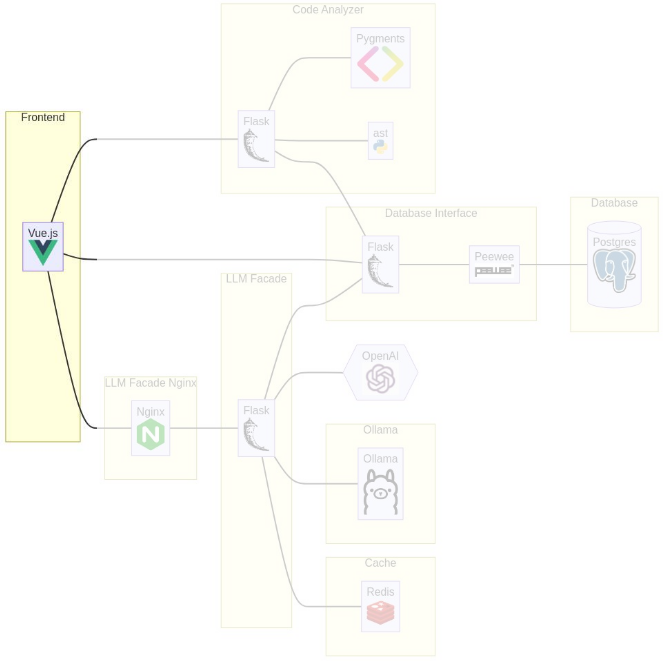
Architecture - Backend



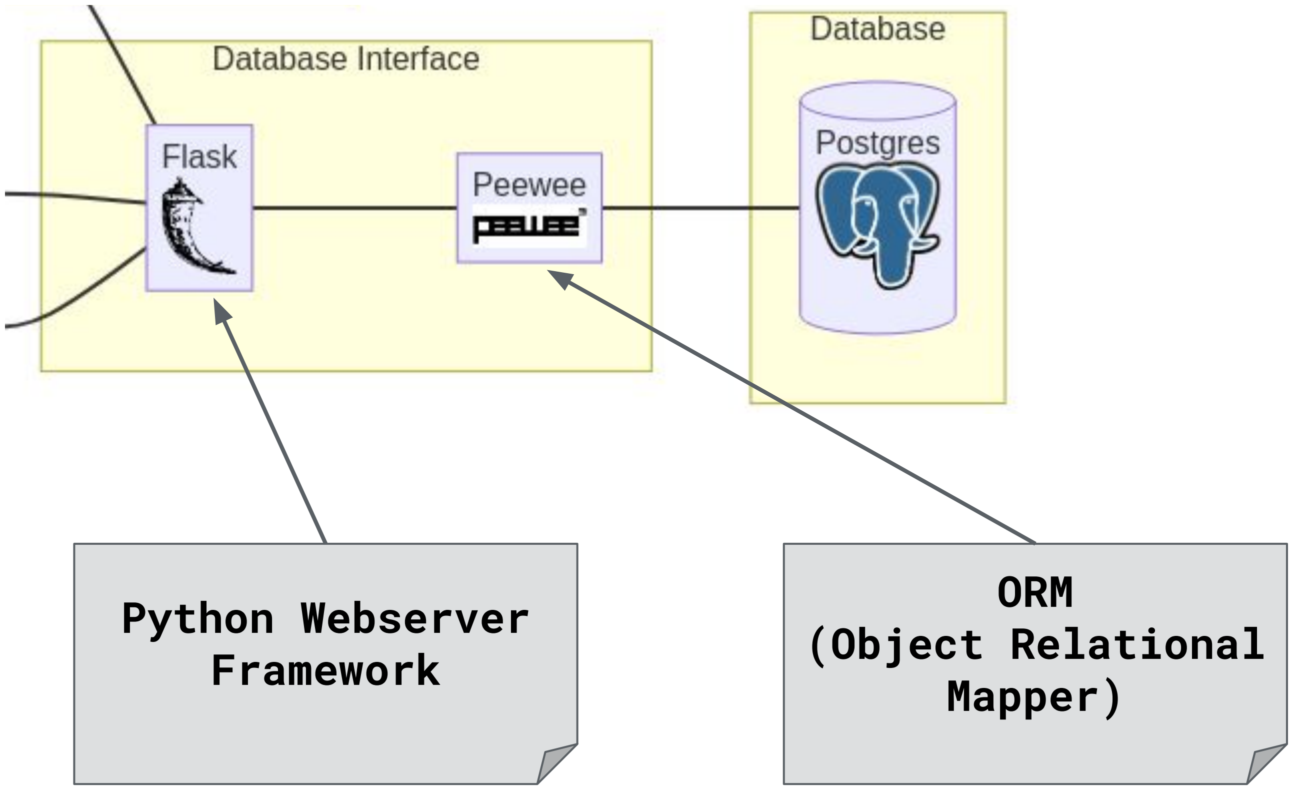
Architecture - Backend



Javascript Frontend Framework

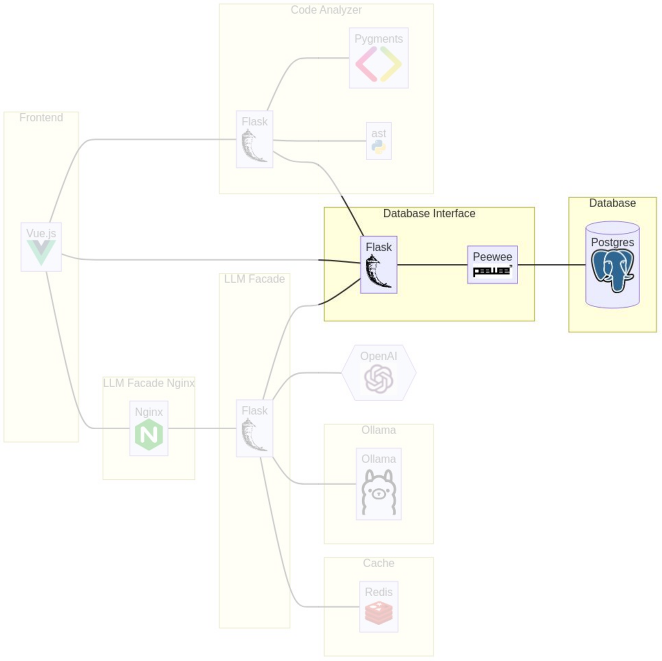


Architecture - Backend

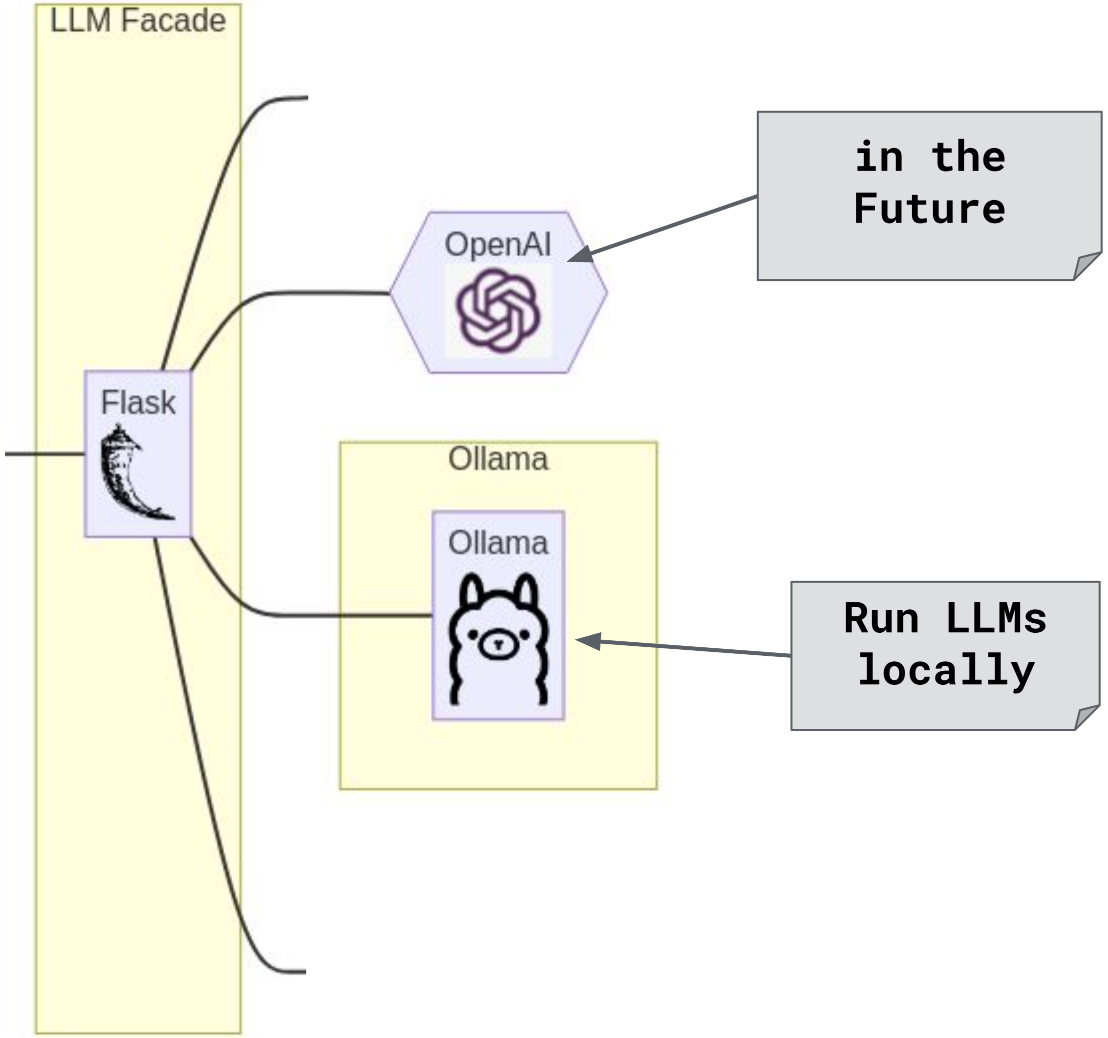


Python Webserver Framework

ORM (Object Relational Mapper)

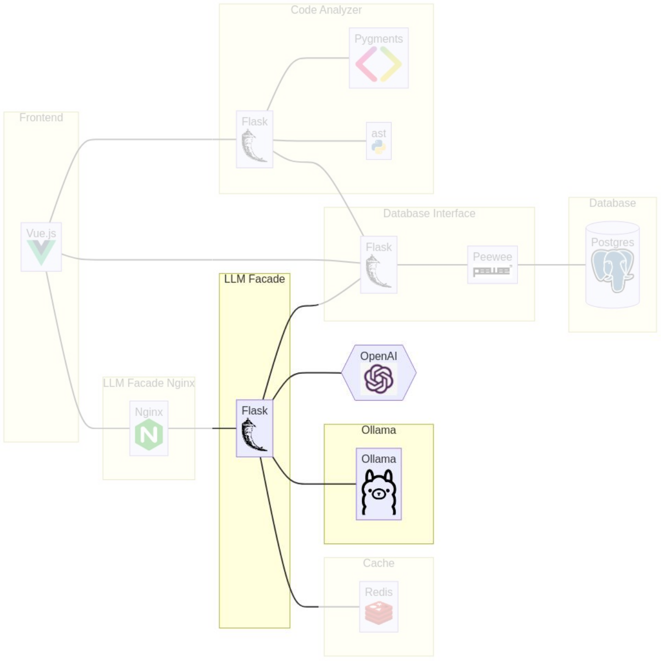


Architecture - Backend

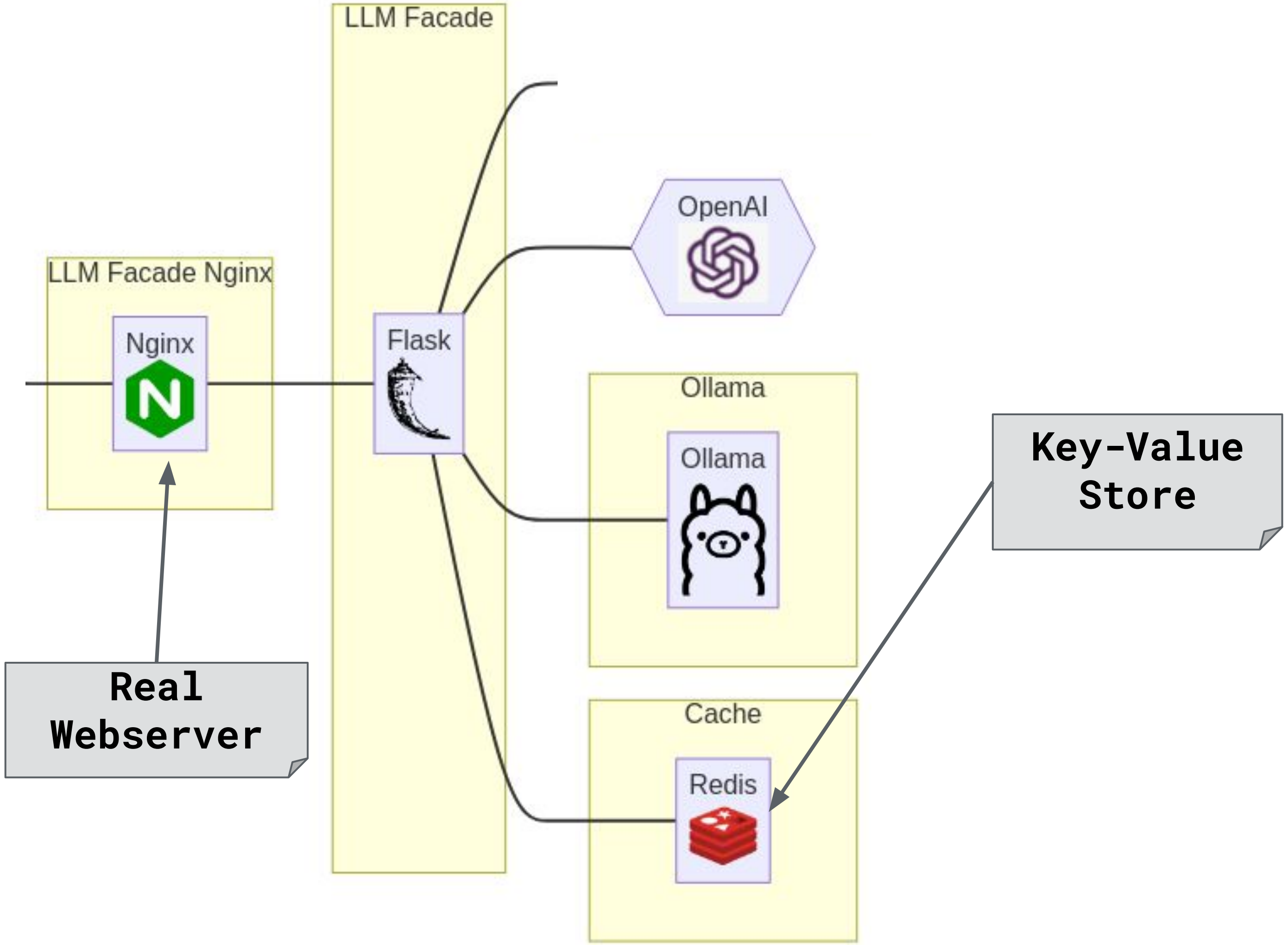


in the Future

Run LLMs locally

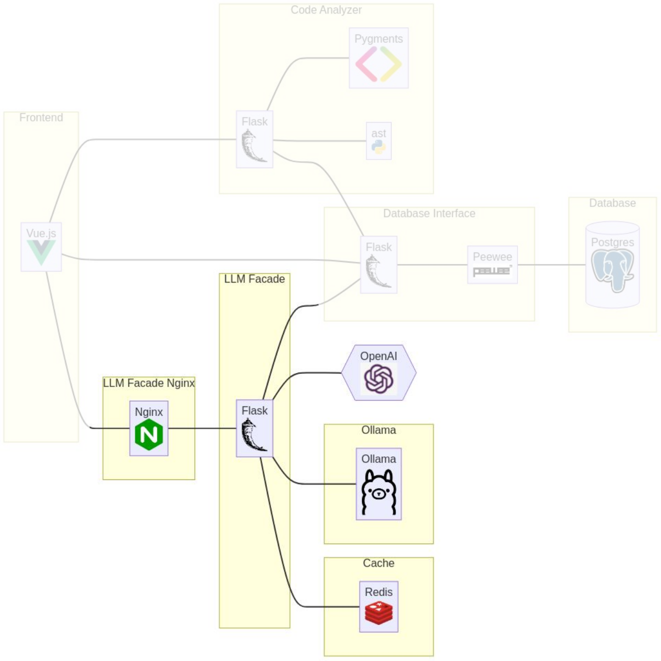


Architecture - Backend

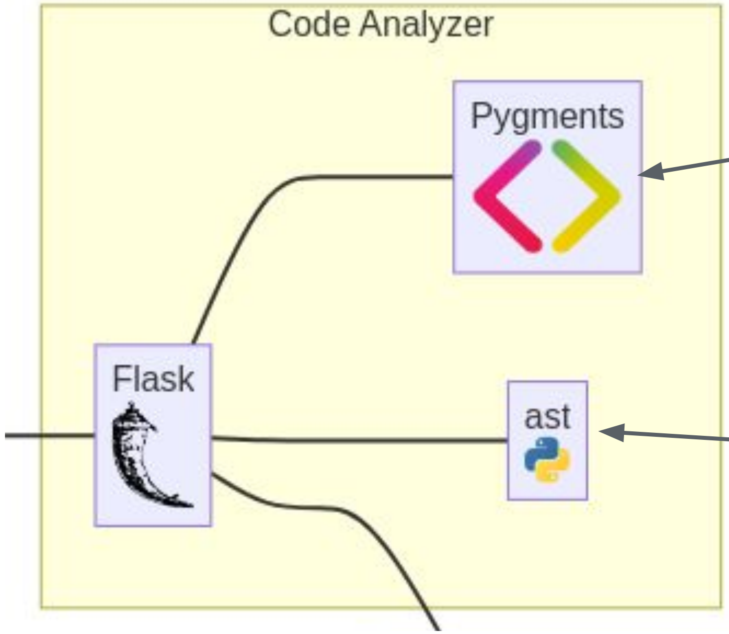


Real Webserver

Key-Value Store

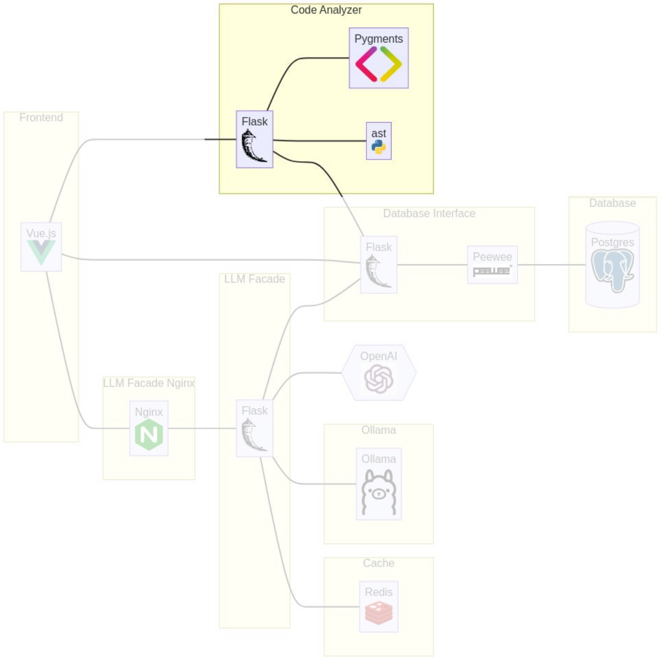


Architecture - Backend

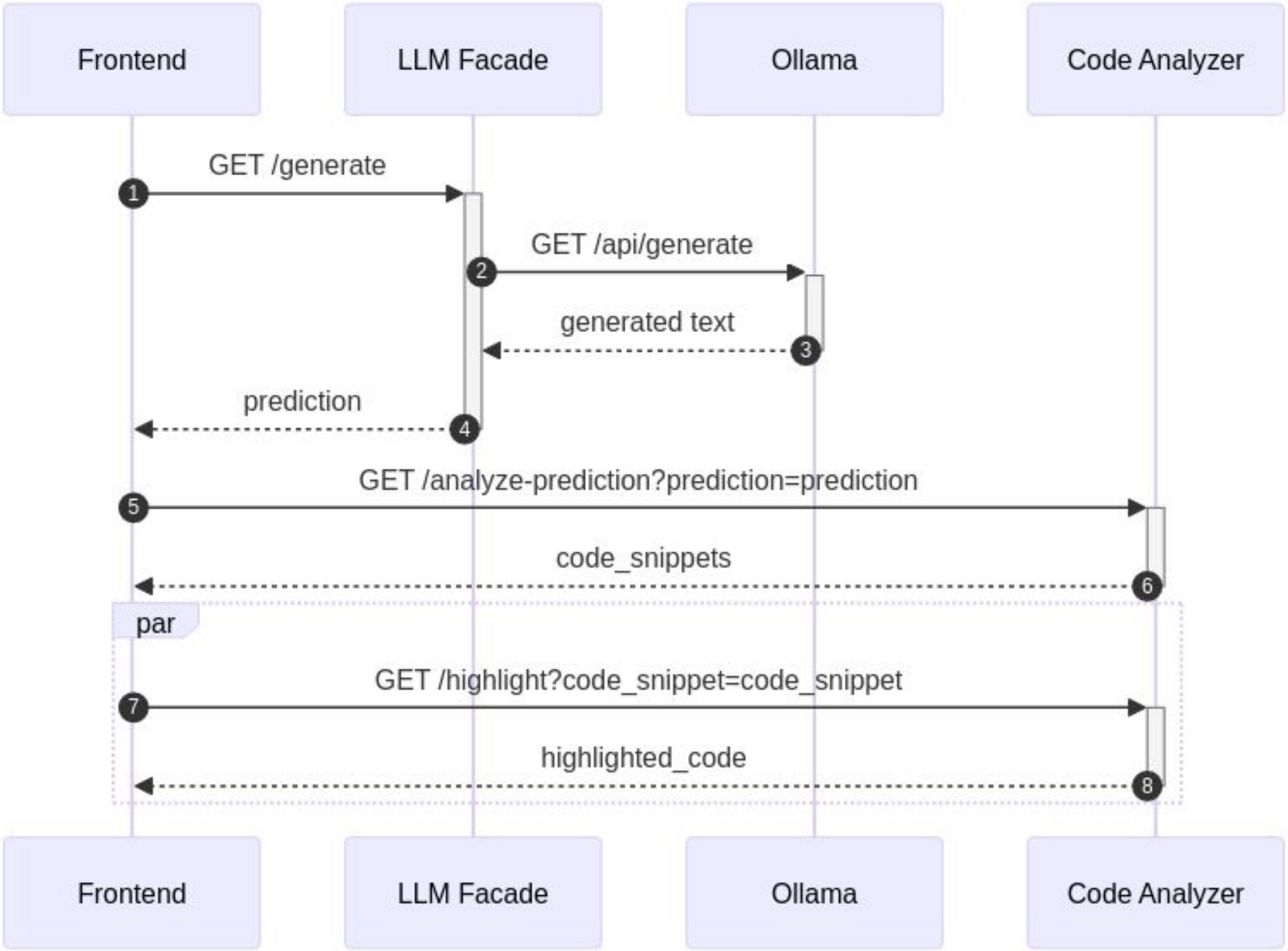


Syntax Highlighting

Abstract Syntax Trees



Example Generation

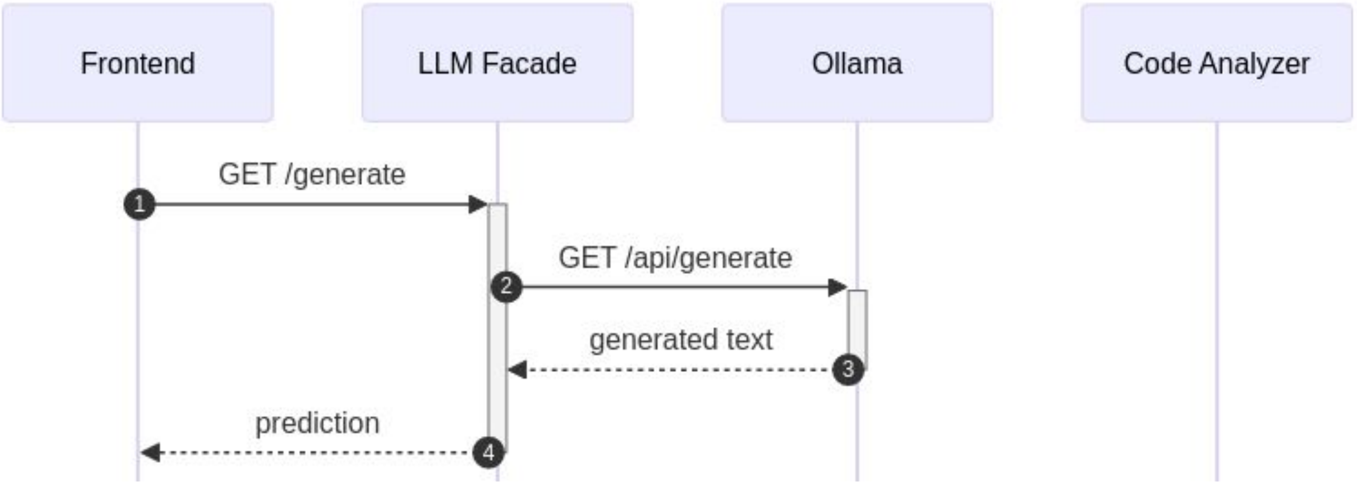


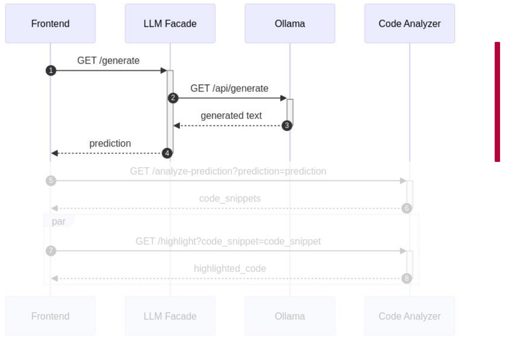
Example Generation

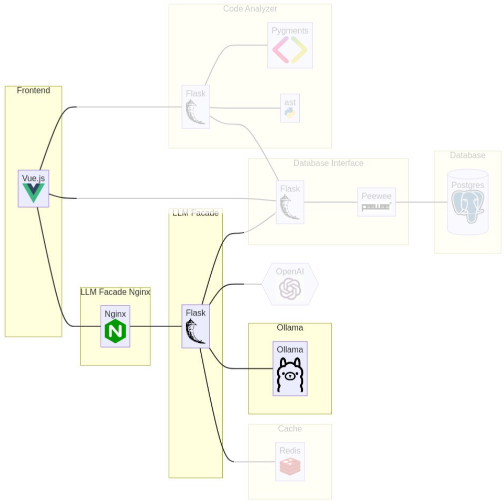
DataFrame.at

● codellama ○ wizardcoder ○ GPT-3.5 ○ GPT-4

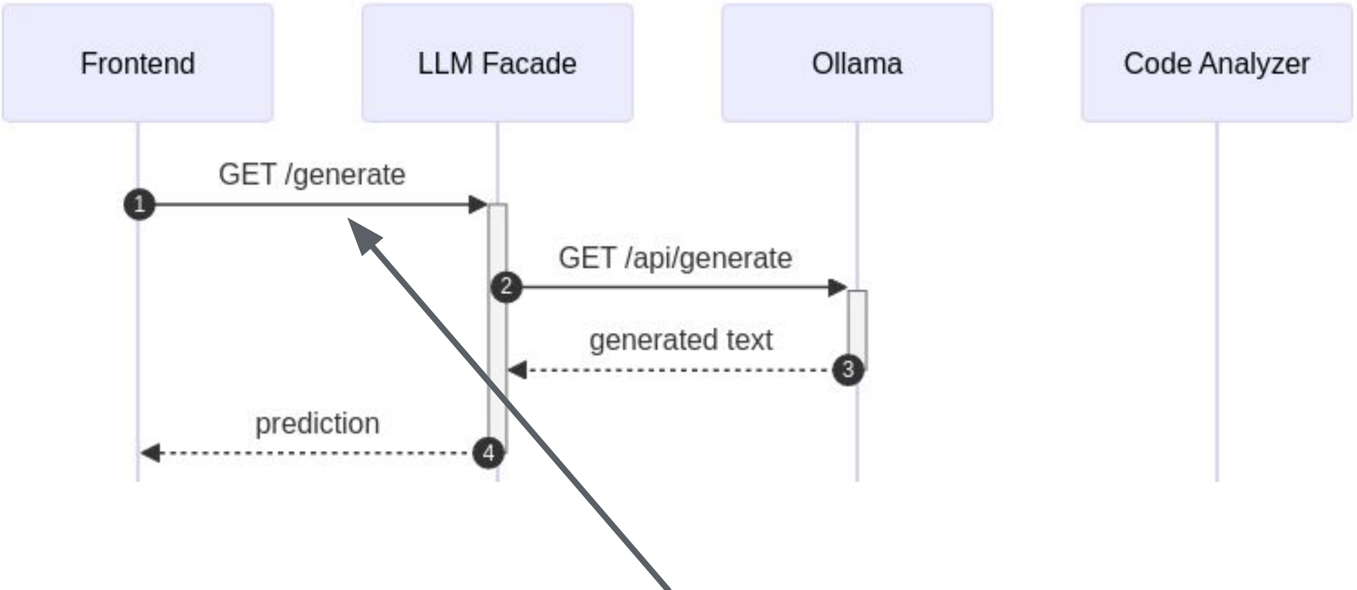
Example Generation



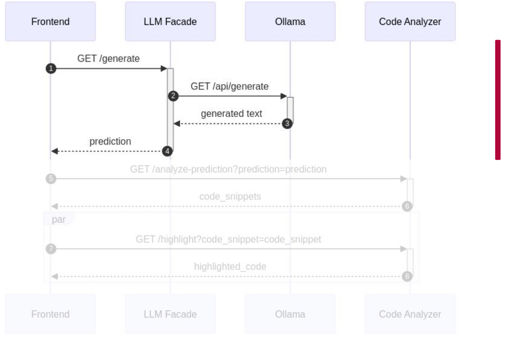


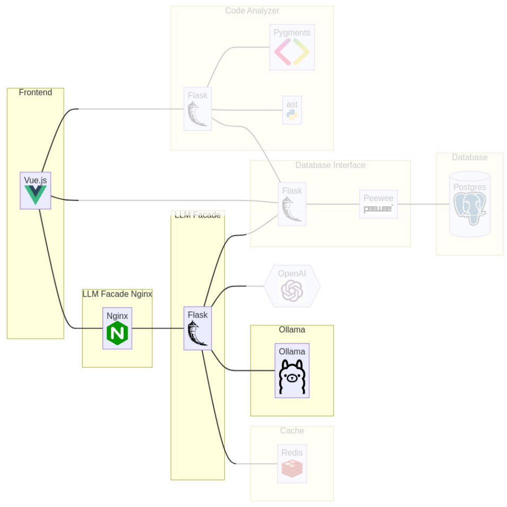


Example Generation

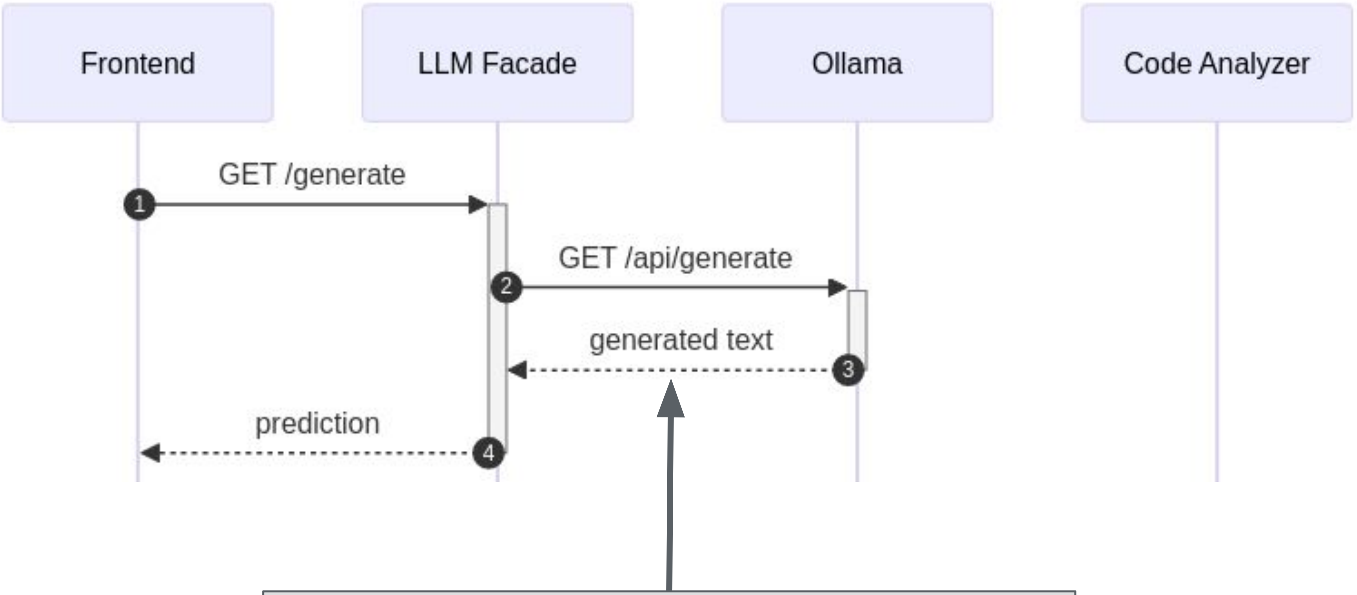


```
{
  model: “codellama:7b-instruct,
  function: “DataFrame.at”,
  prompt: “You are a ...”
}
```

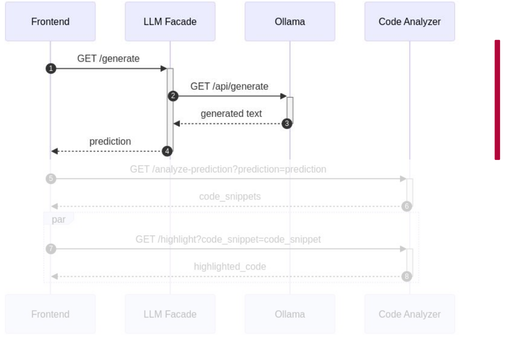


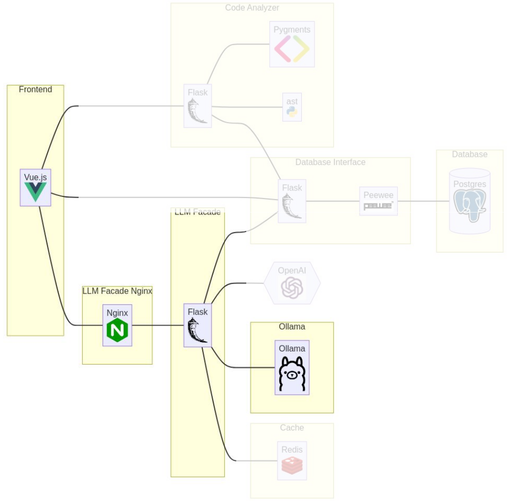


Example Generation

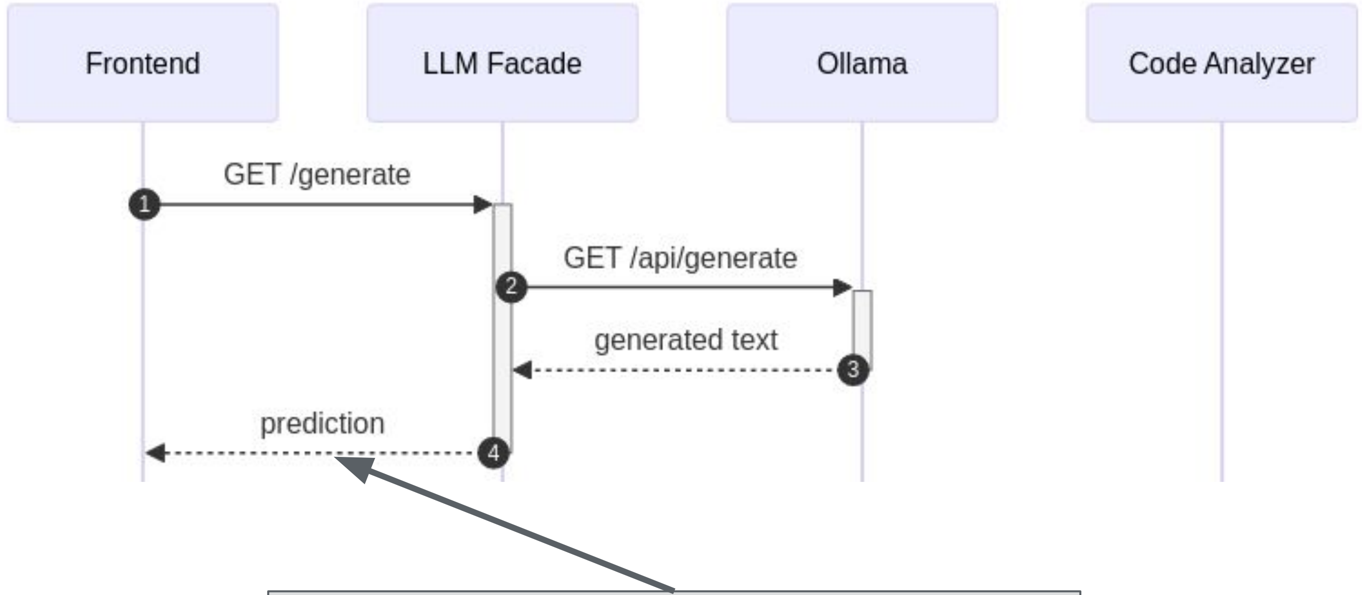


```
import pandas as pd

# create a sample DataFrame
df = pd.DataFrame({'A': [0, 2, 3],
'B': [4, 1, 6], 'C': [3, 2, 1]},
index=[4, 5, 6])

...
```

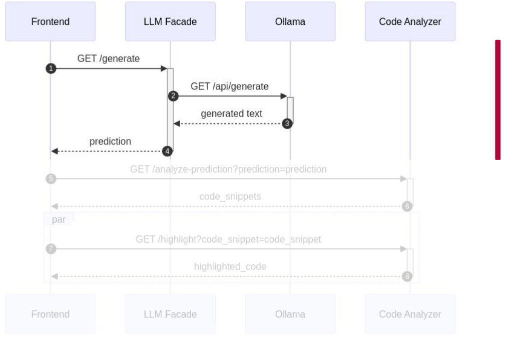


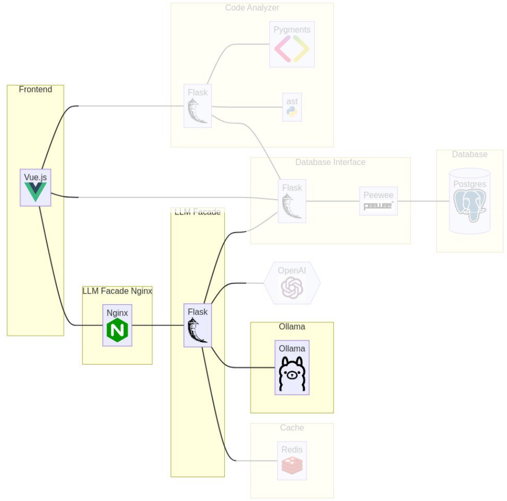


Example Generation

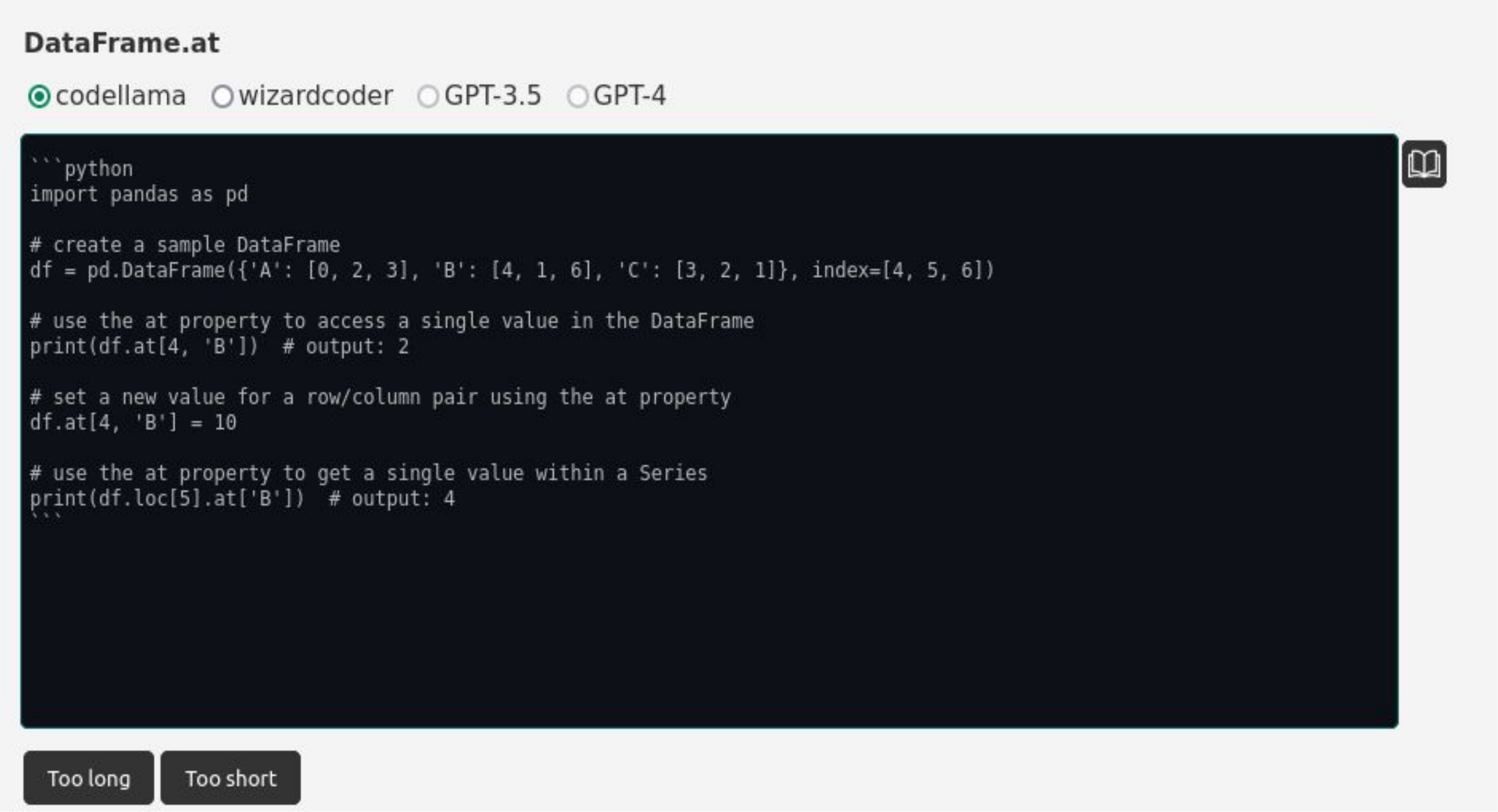


```
{
  text: “import pandas as pd ...”,
  model: “codellama:7b-instruct”,
  token_amount: 456
}
```





Example Generation

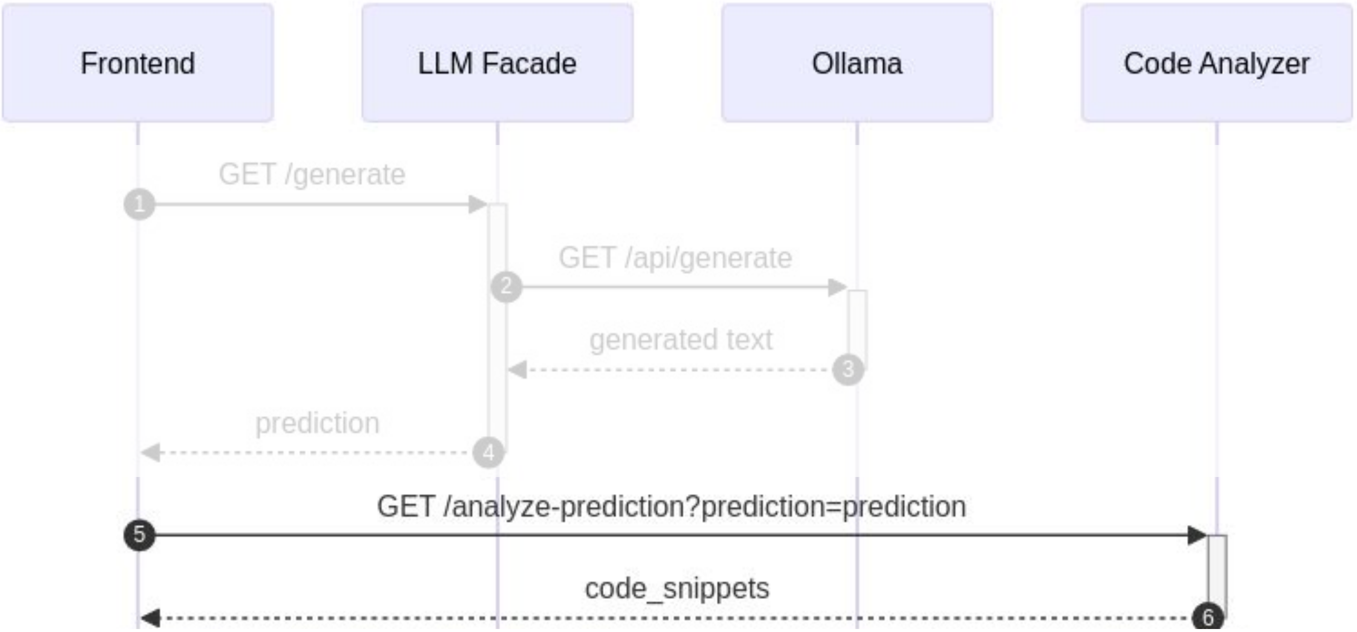


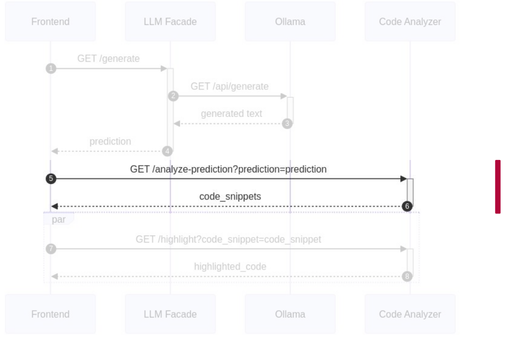
DataFrame.at

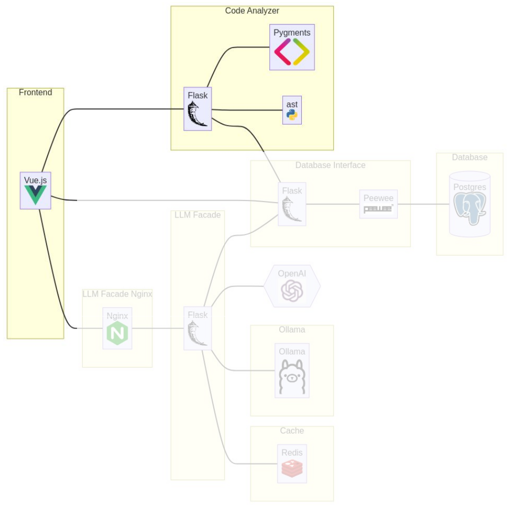
● codellama ○ wizardcoder ○ GPT-3.5 ○ GPT-4

~~~
```python
import pandas as pd

# create a sample DataFrame
df = pd.DataFrame({'A': [0, 2, 3], 'B': [4, 1, 6], 'C': [3, 2, 1]}, index=[4, 5, 6])

# use the at property to access a single value in the DataFrame
print(df.at[4, 'B'])  # output: 2

# set a new value for a row/column pair using the at property
df.at[4, 'B'] = 10

# use the at property to get a single value within a Series
print(df.loc[5].at['B'])  # output: 4
```
~~~

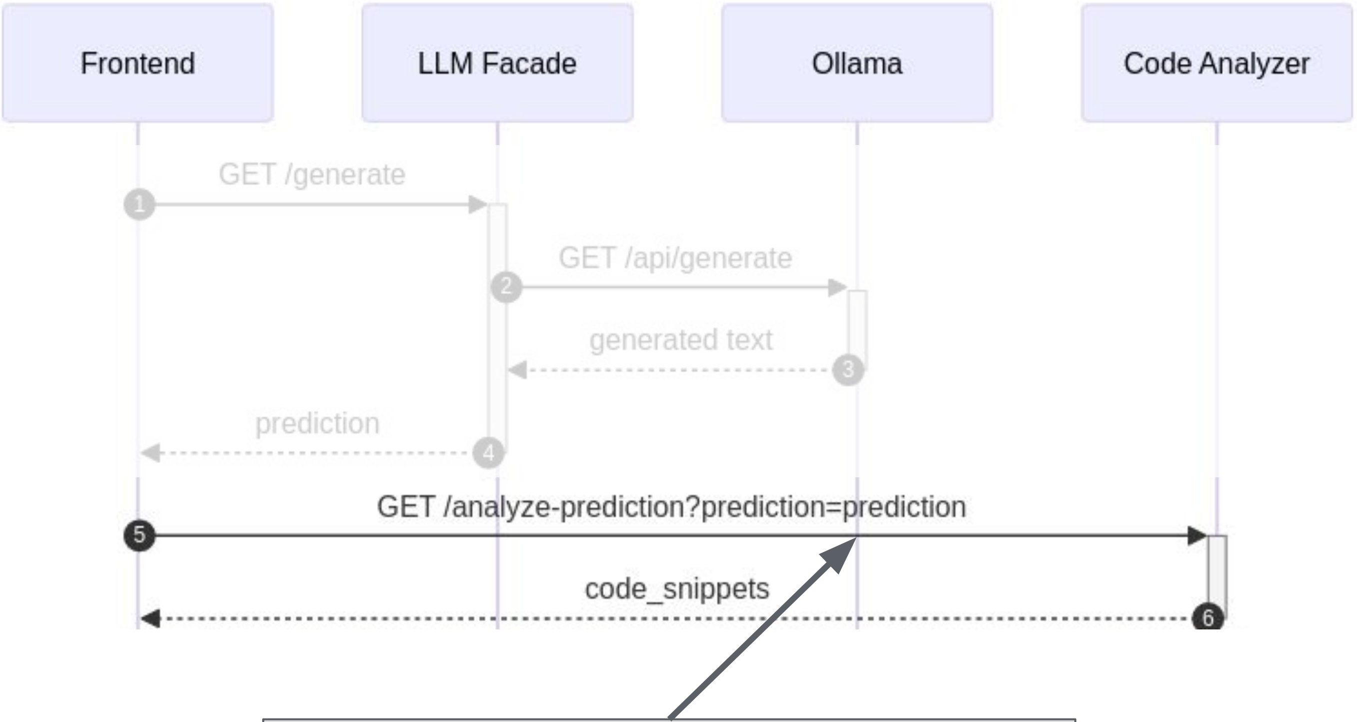
Example Generation



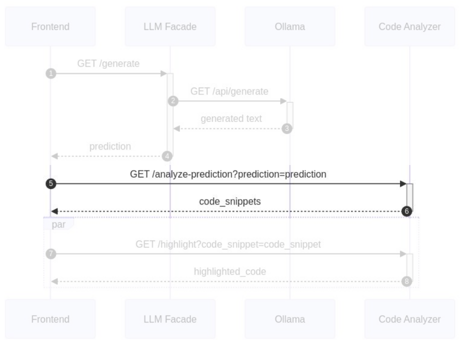


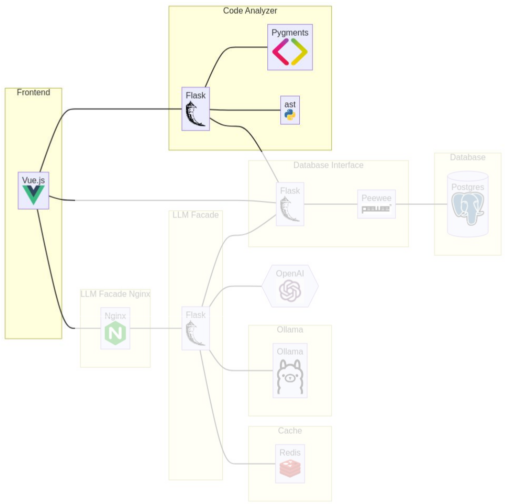


Example Generation

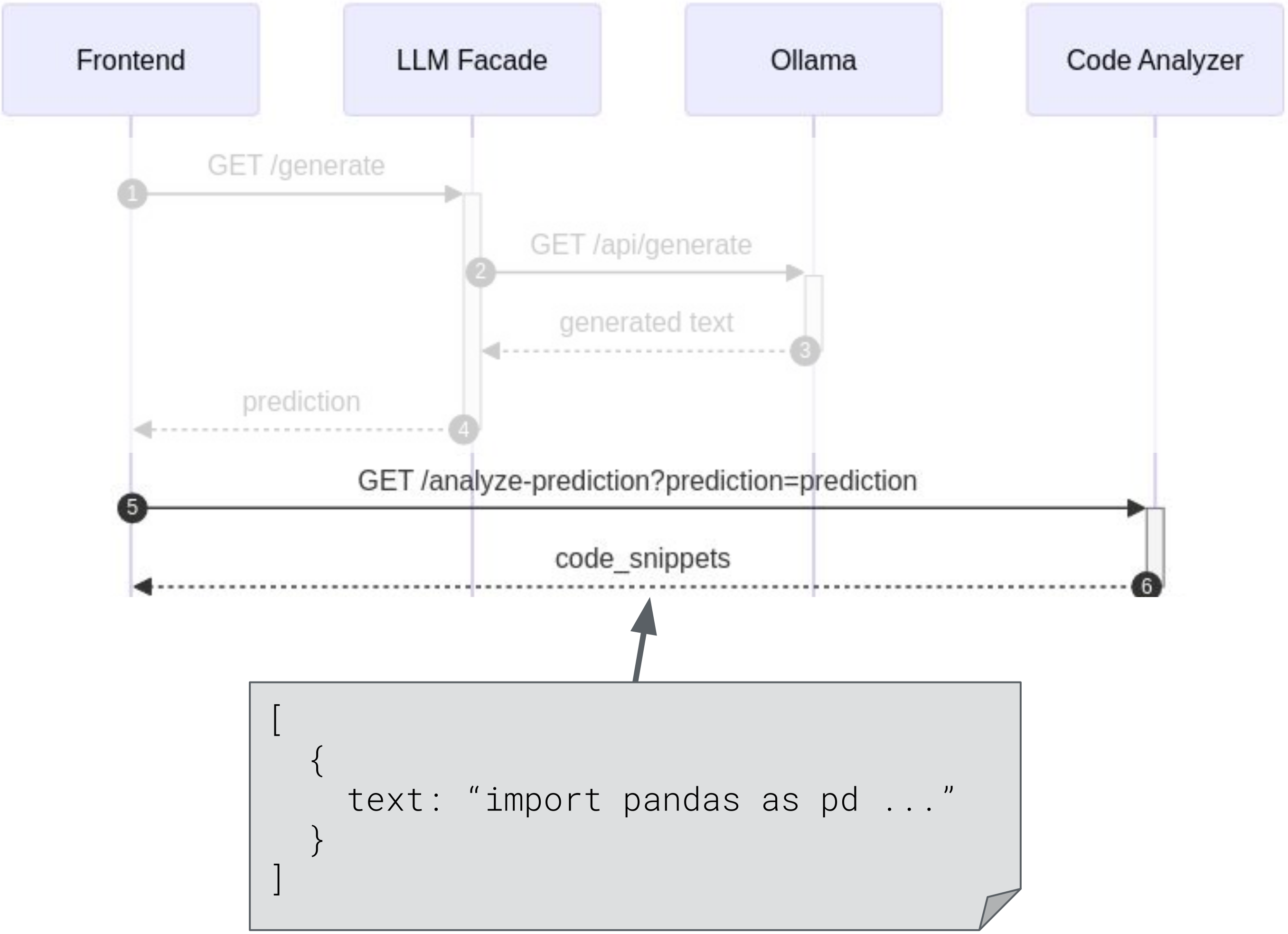


```
{
  text: “import pandas as pd ...”,
  model: “codellama:7b-instruct”,
  token_amount: 456
}
```

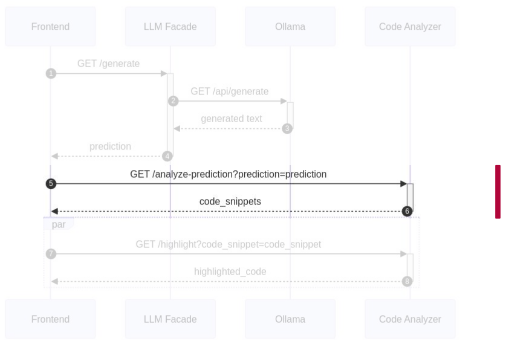


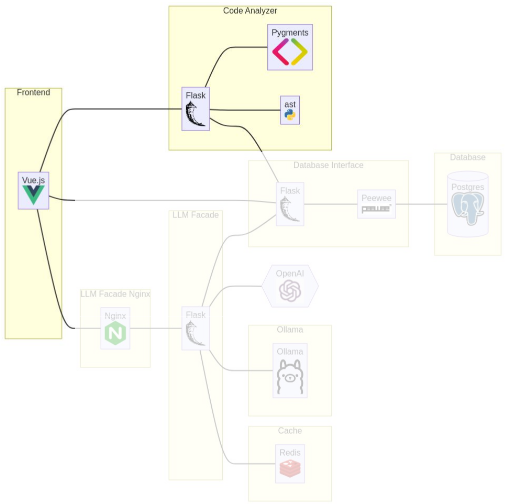


Example Generation

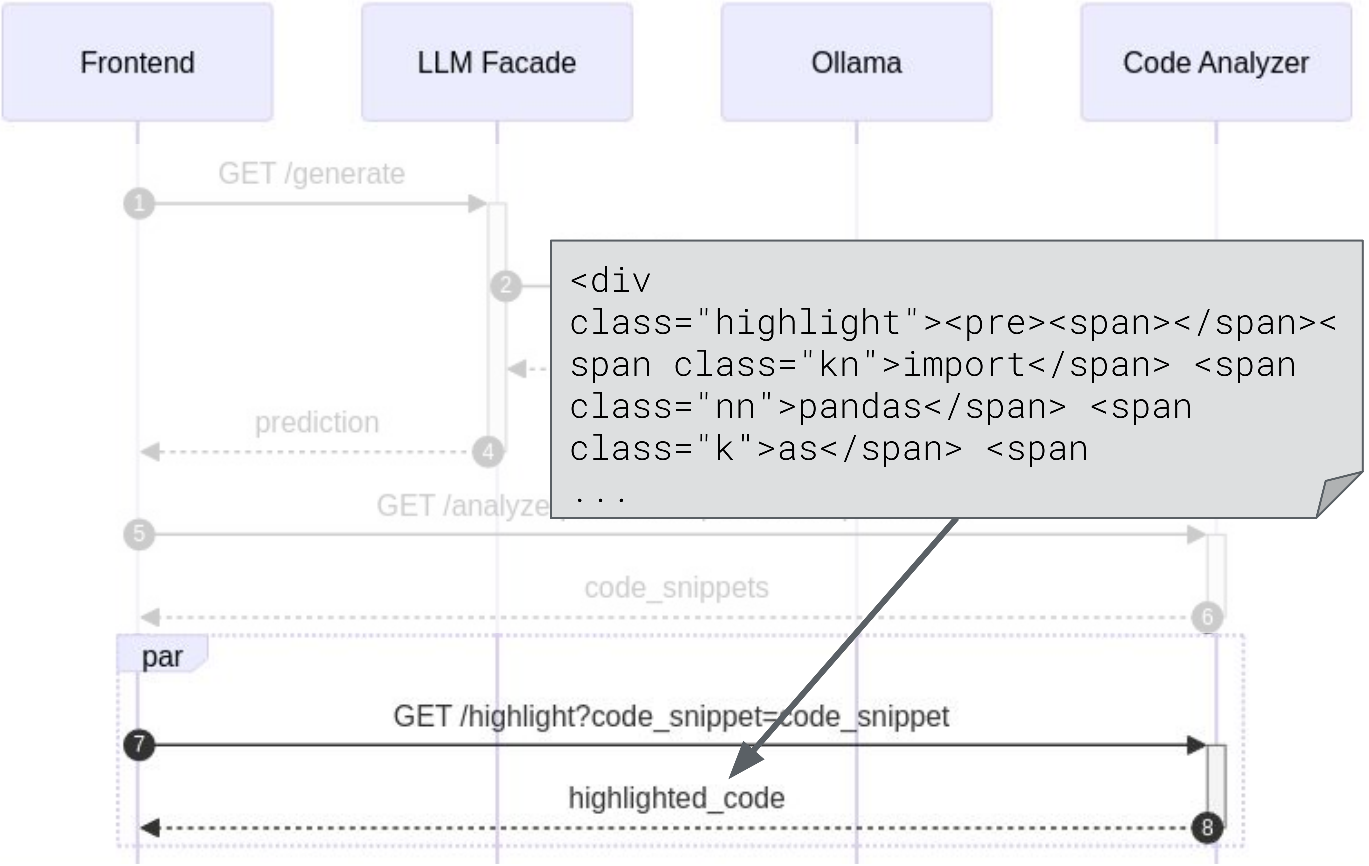


```
[
  {
    text: “import pandas as pd ...”
  }
]
```

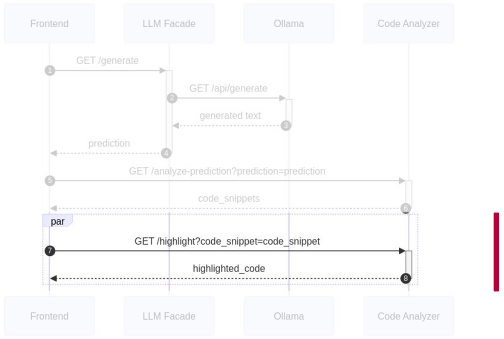


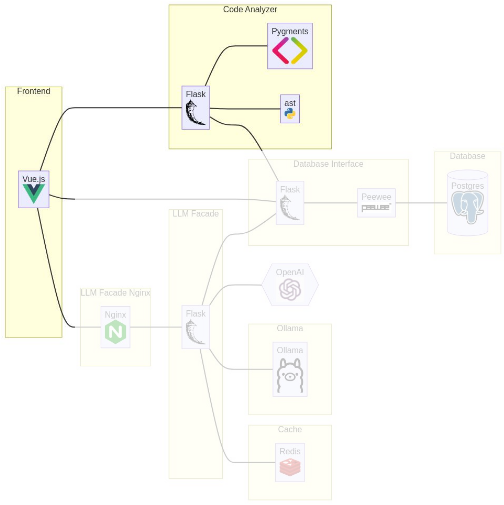


Example Generation

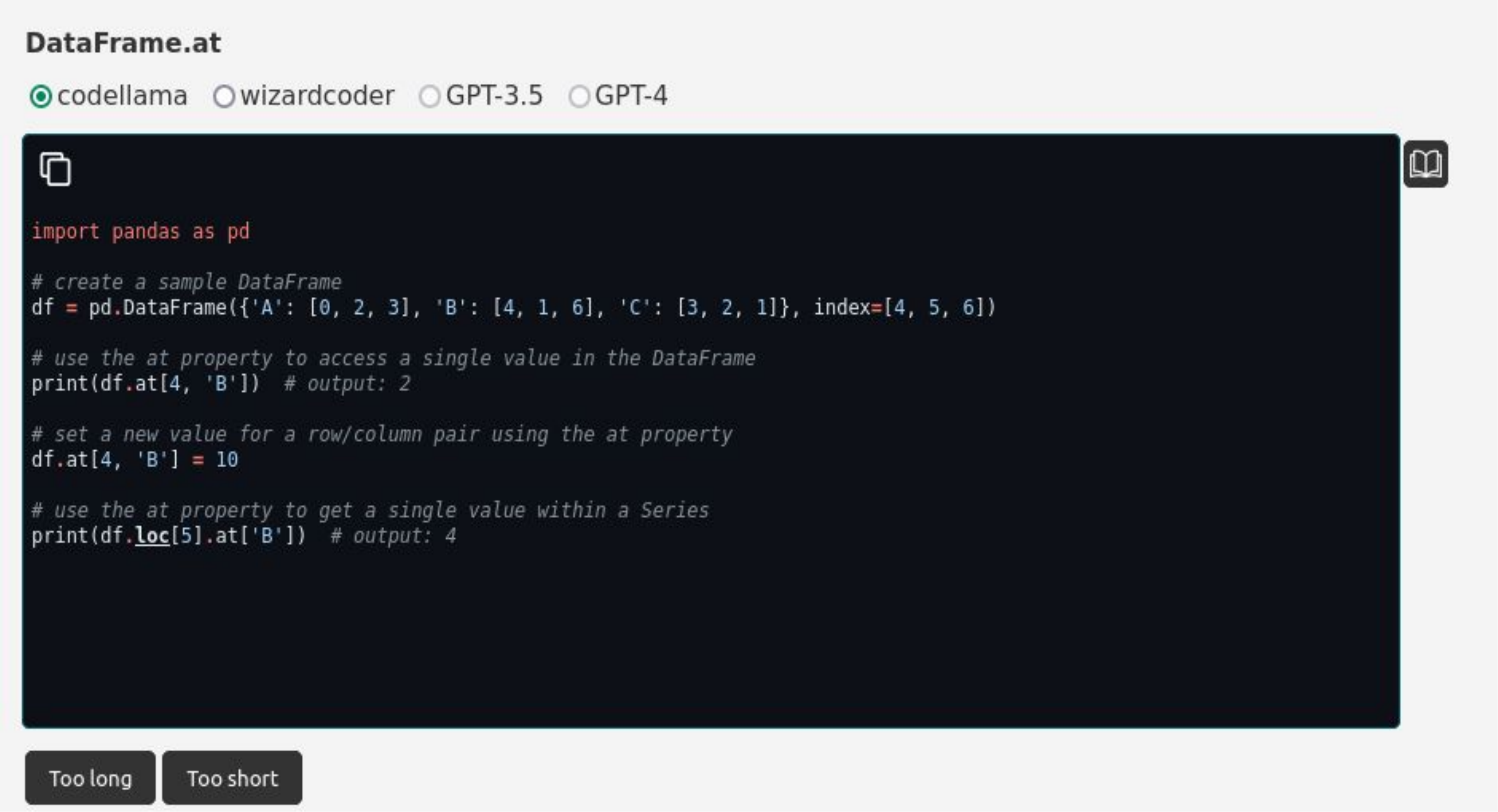


```
<div
class="highlight"><pre><span></span><
span class="kn">import</span> <span
class="nn">pandas</span> <span
class="k">as</span> <span
...
```





Example Generation



DataFrame.at

● codellama ○ wizardcoder ○ GPT-3.5 ○ GPT-4

```python
import pandas as pd

# create a sample DataFrame
df = pd.DataFrame({'A': [0, 2, 3], 'B': [4, 1, 6], 'C': [3, 2, 1]}, index=[4, 5, 6])

# use the at property to access a single value in the DataFrame
print(df.at[4, 'B'])  # output: 2

# set a new value for a row/column pair using the at property
df.at[4, 'B'] = 10

# use the at property to get a single value within a Series
print(df.loc[5].at['B'])  # output: 4
```

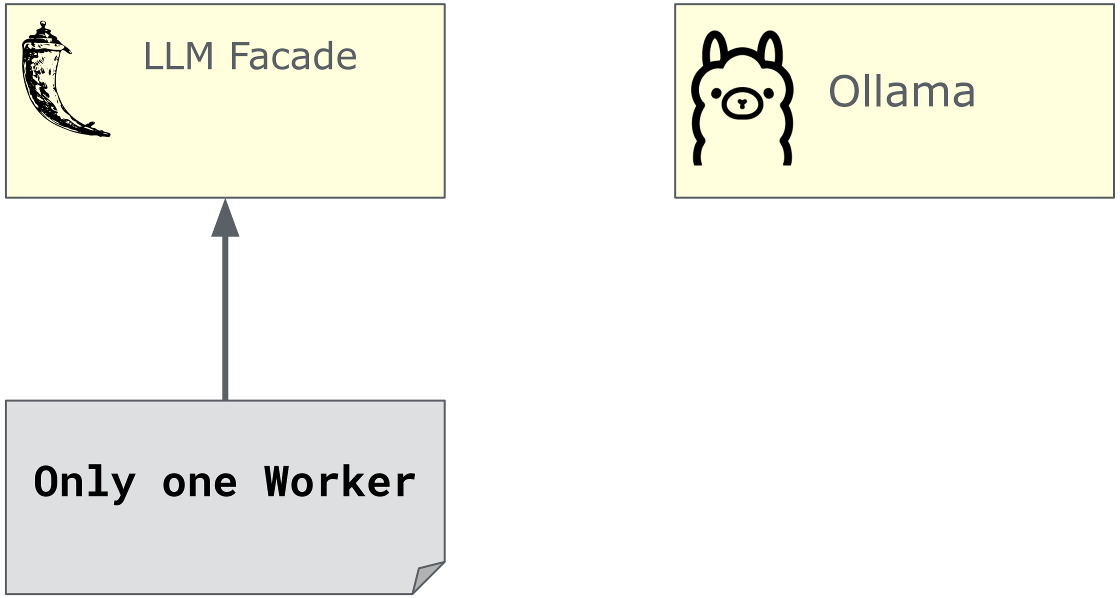
Simultaneous Example Generation

LLM Facade

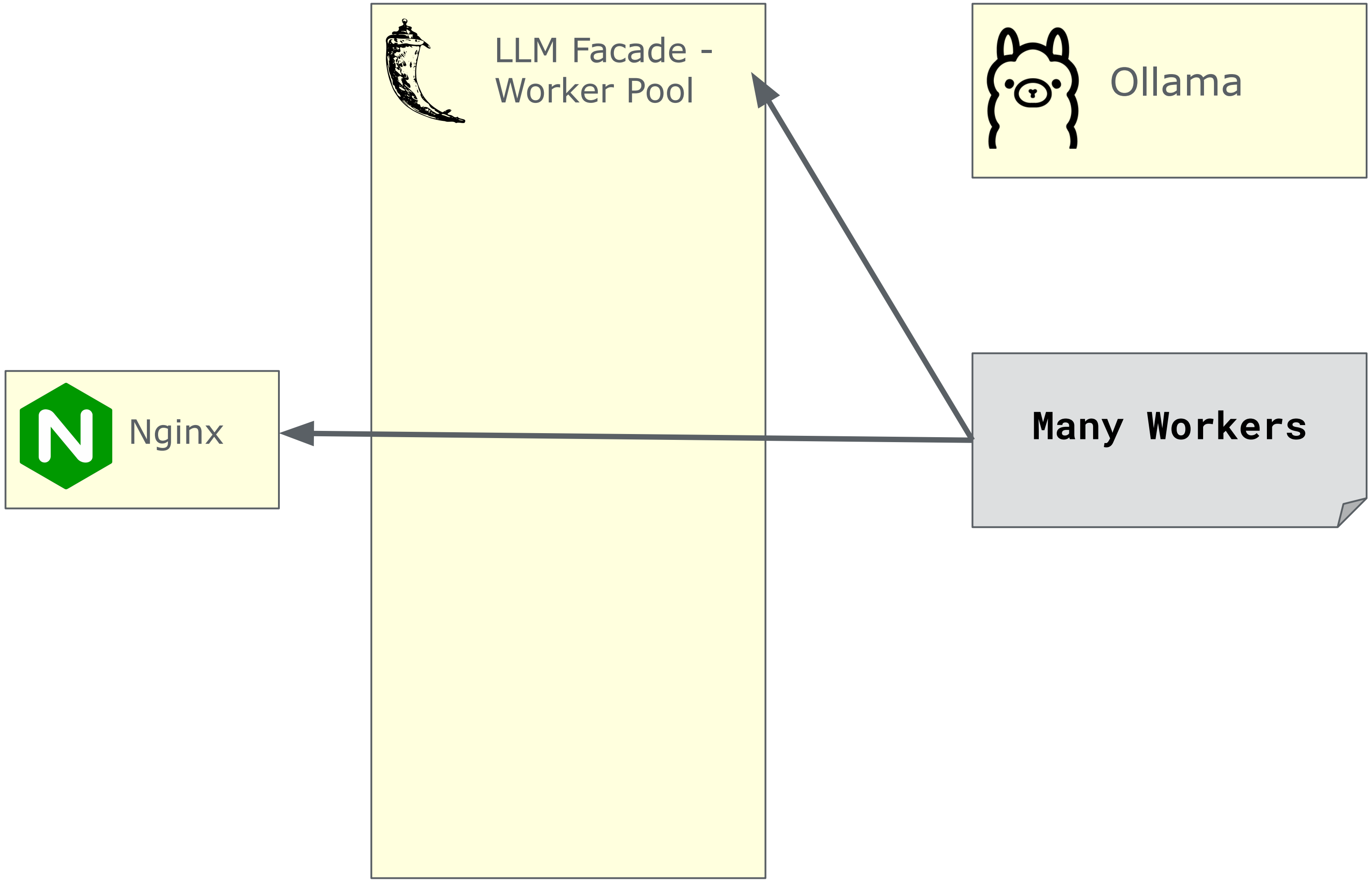
Ollama

`Can it handle multiple requests?`

Simultaneous Example Generation



Simultaneous Example Generation



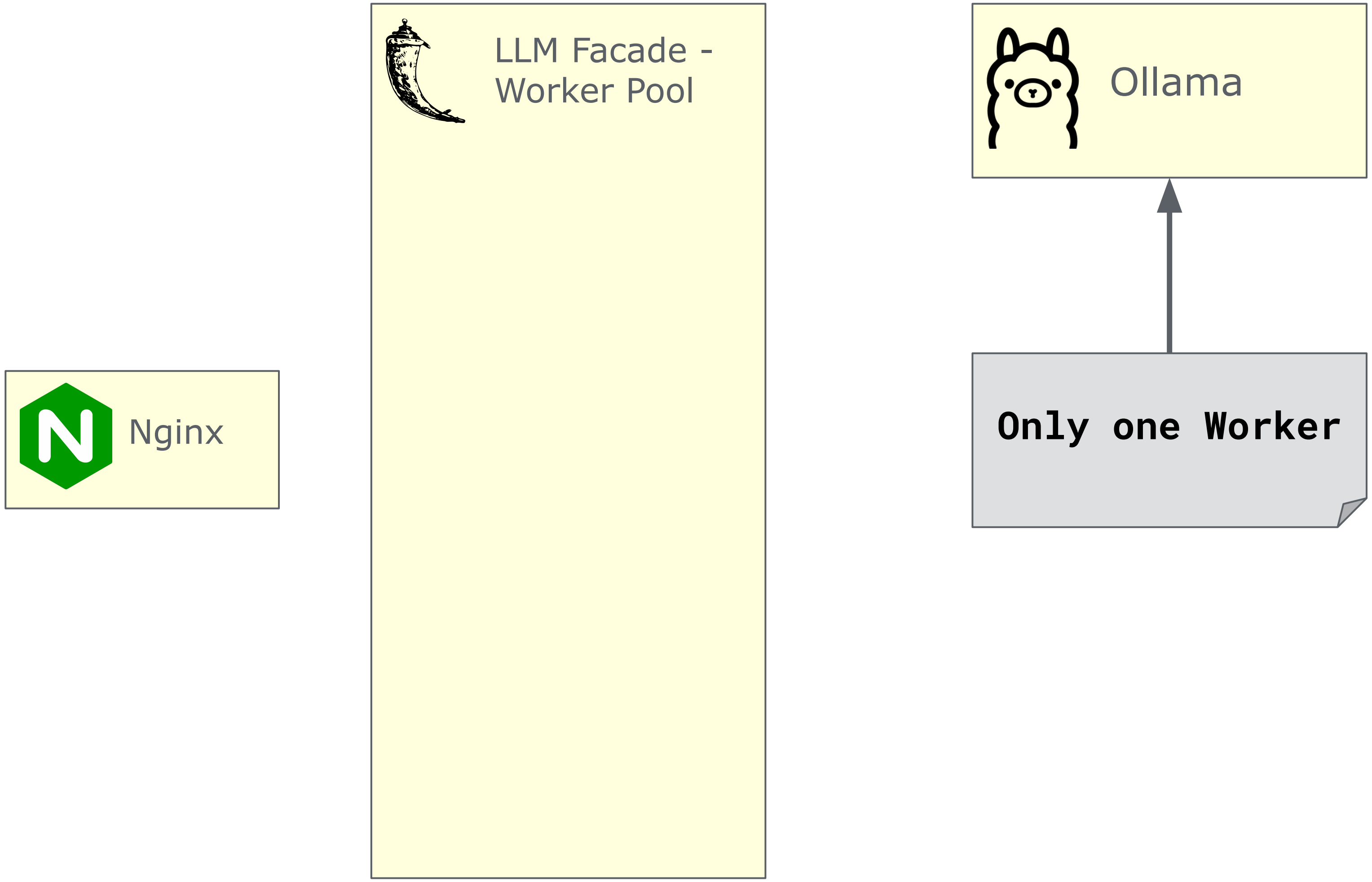
LLM Facade - Worker Pool

Ollama

Nginx

Many Workers

Simultaneous Example Generation



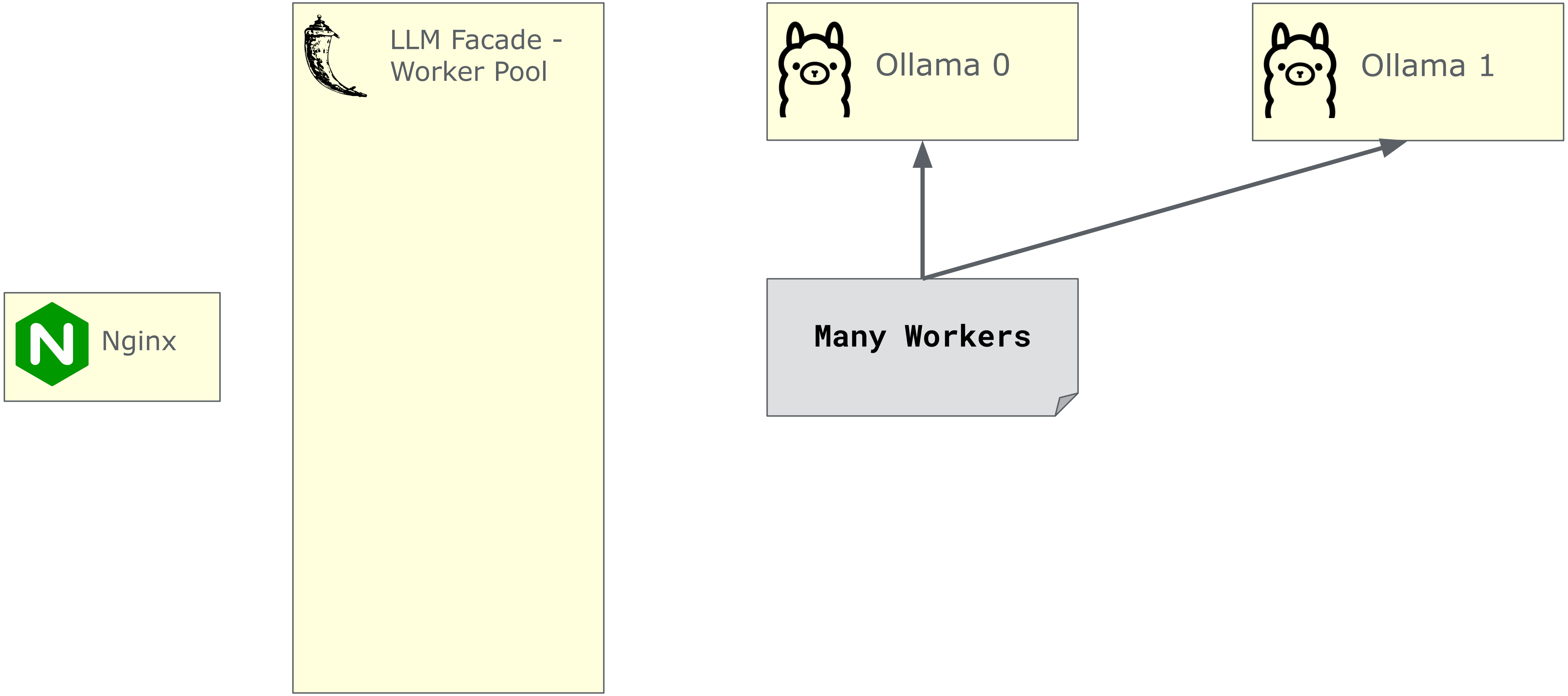
LLM Facade - Worker Pool

Ollama

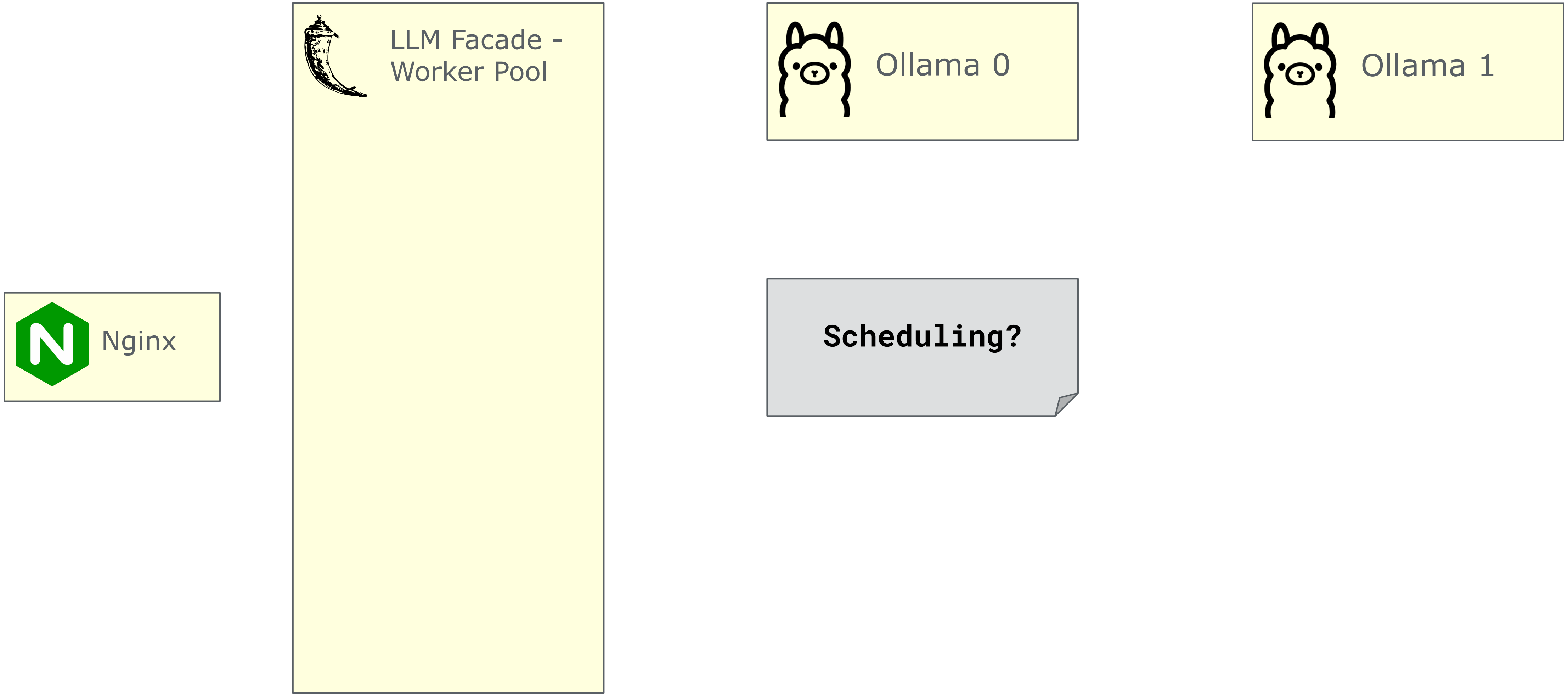
Nginx

Only one Worker

Simultaneous Example Generation



Simultaneous Example Generation



LLM Facade - Worker Pool

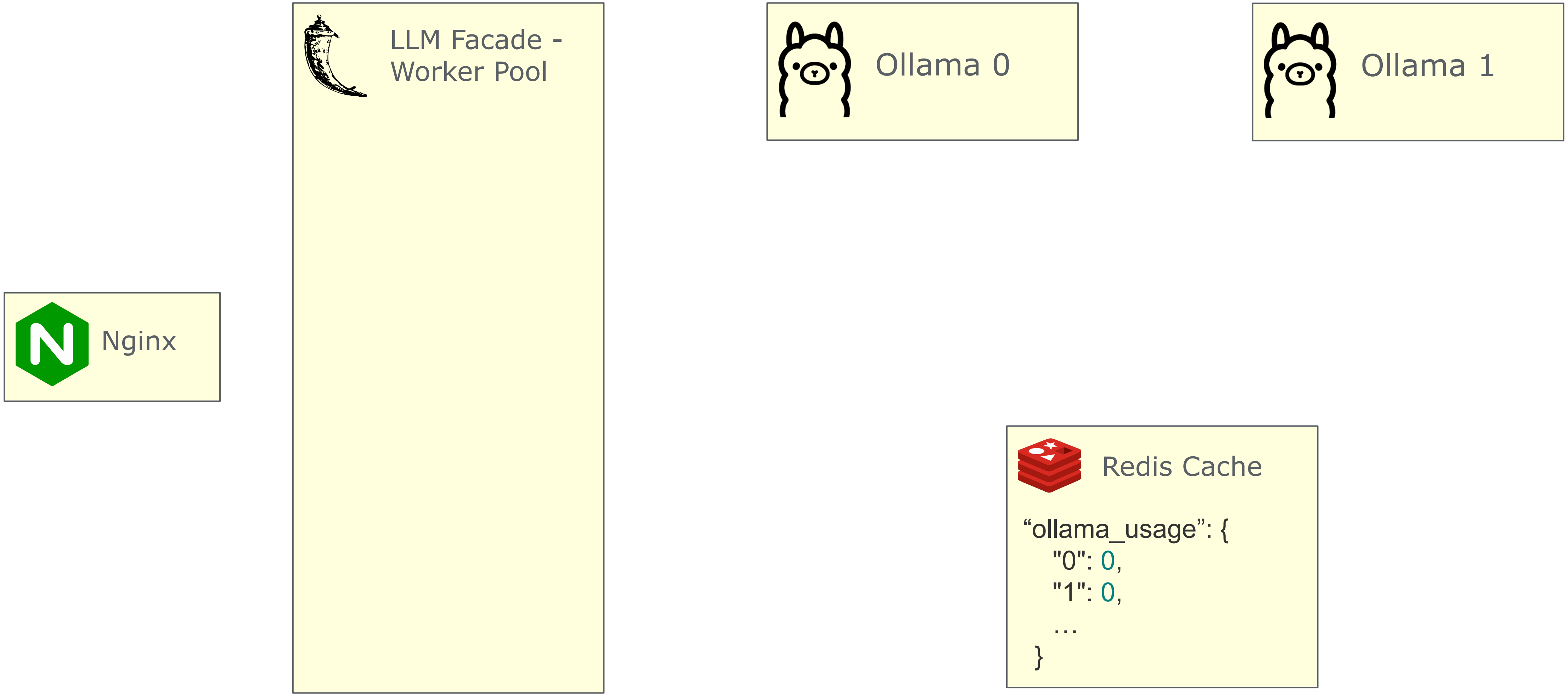
Ollama 0

Ollama 1

Nginx

Scheduling?

Simultaneous Example Generation



LLM Facade - Worker Pool

Ollama 0

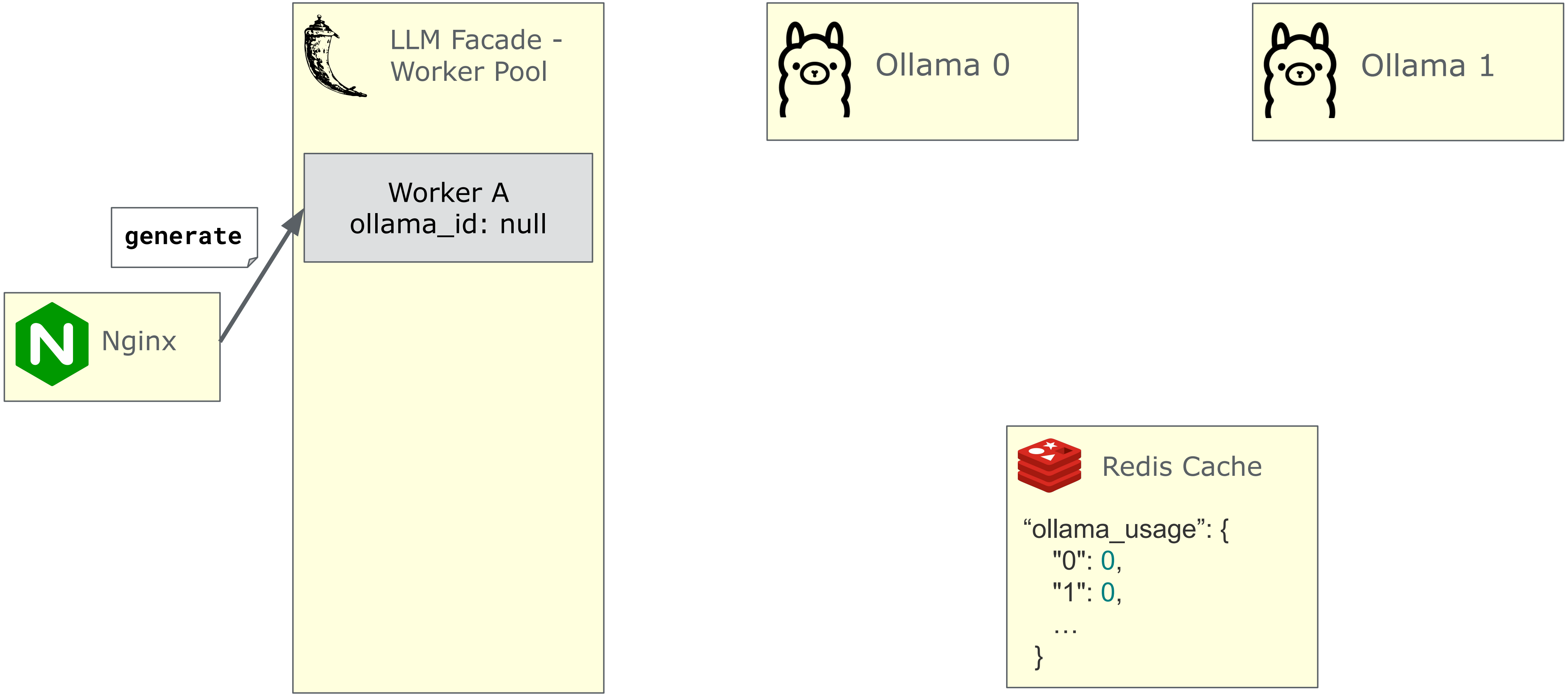
Ollama 1

Nginx

Redis Cache

```
"ollama_usage": {
   "0": 0,
   "1": 0,
   …
 }
```

Simultaneous Example Generation



LLM Facade - Worker Pool

Worker A
ollama_id: null

`generate`

Nginx

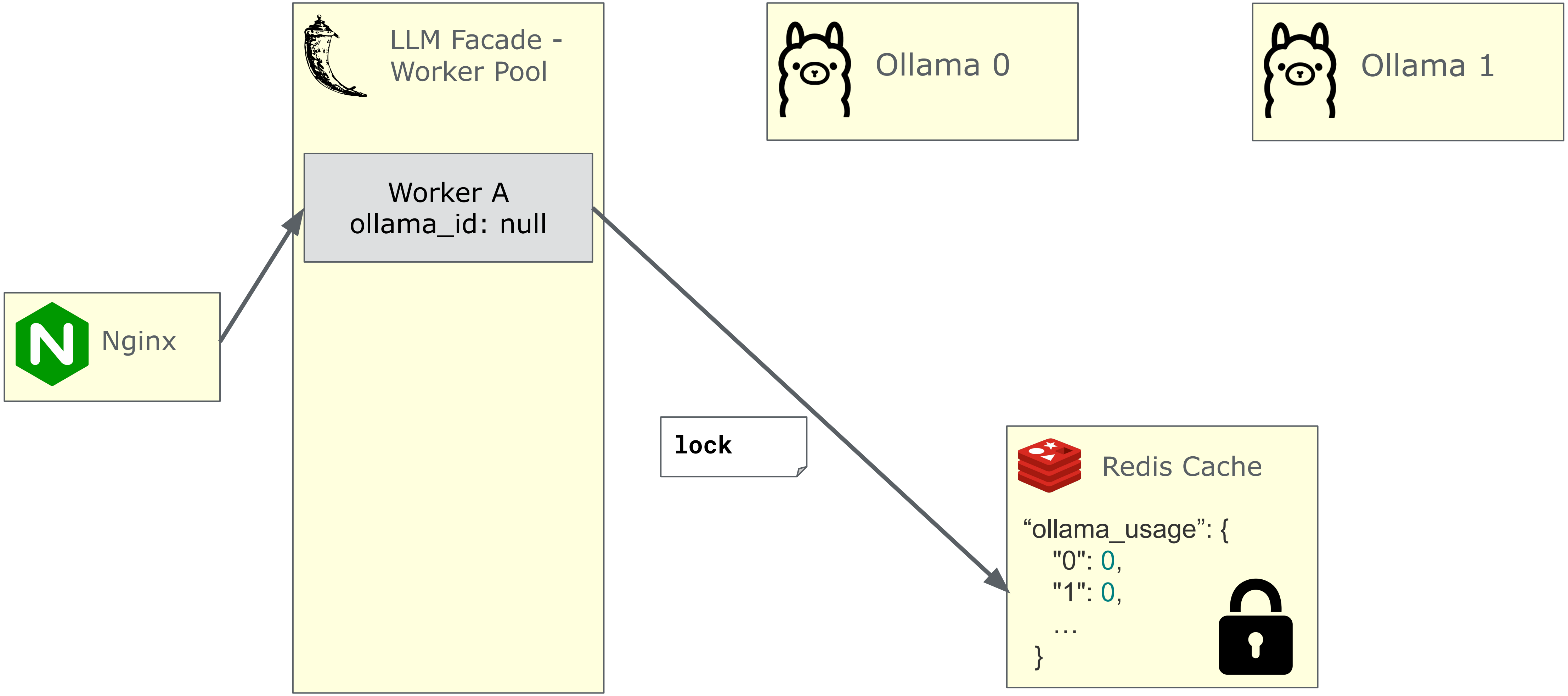
Ollama 0

Ollama 1

Redis Cache

```
“ollama_usage”: {
   "0": 0,
   "1": 0,
   …
 }
```

Simultaneous Example Generation



LLM Facade - Worker Pool

Ollama 0

Ollama 1

Worker A
ollama_id: null

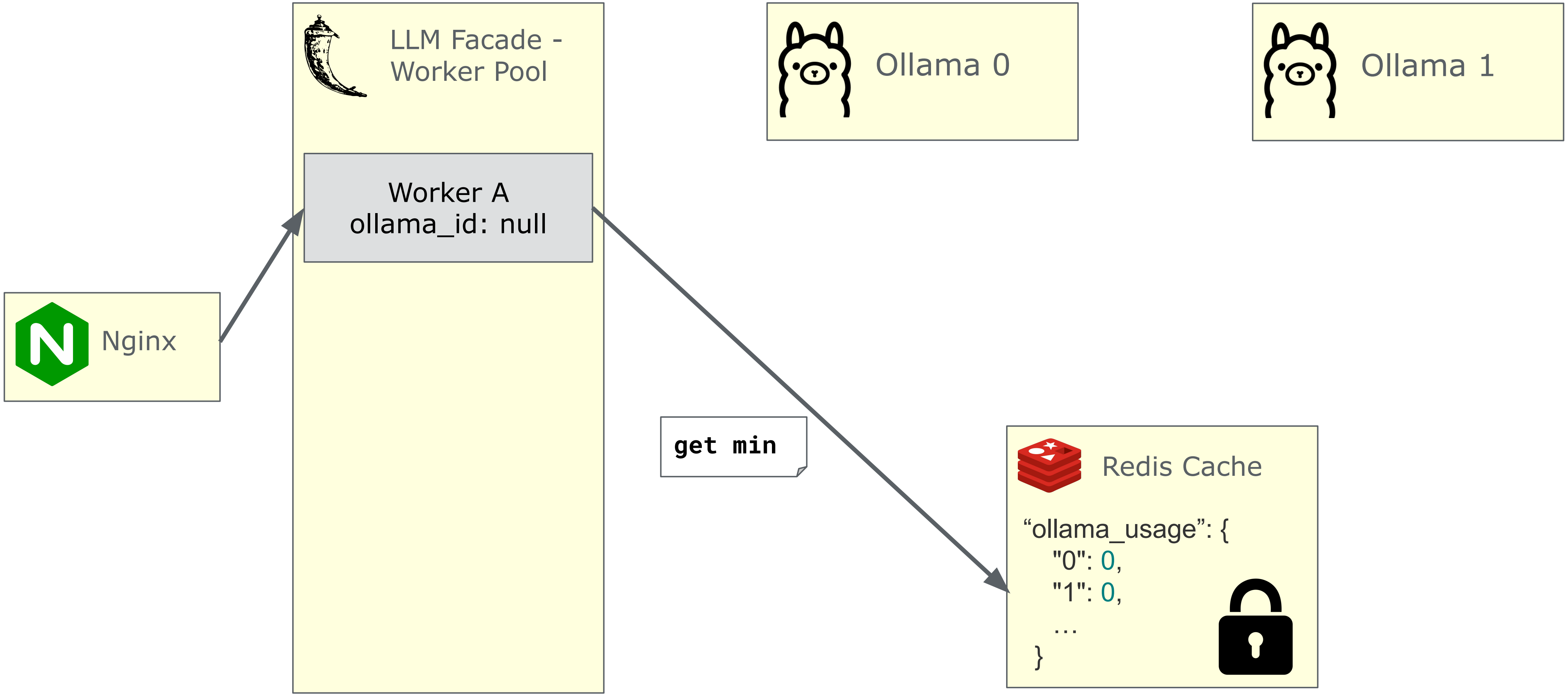
Nginx

`lock`

Redis Cache

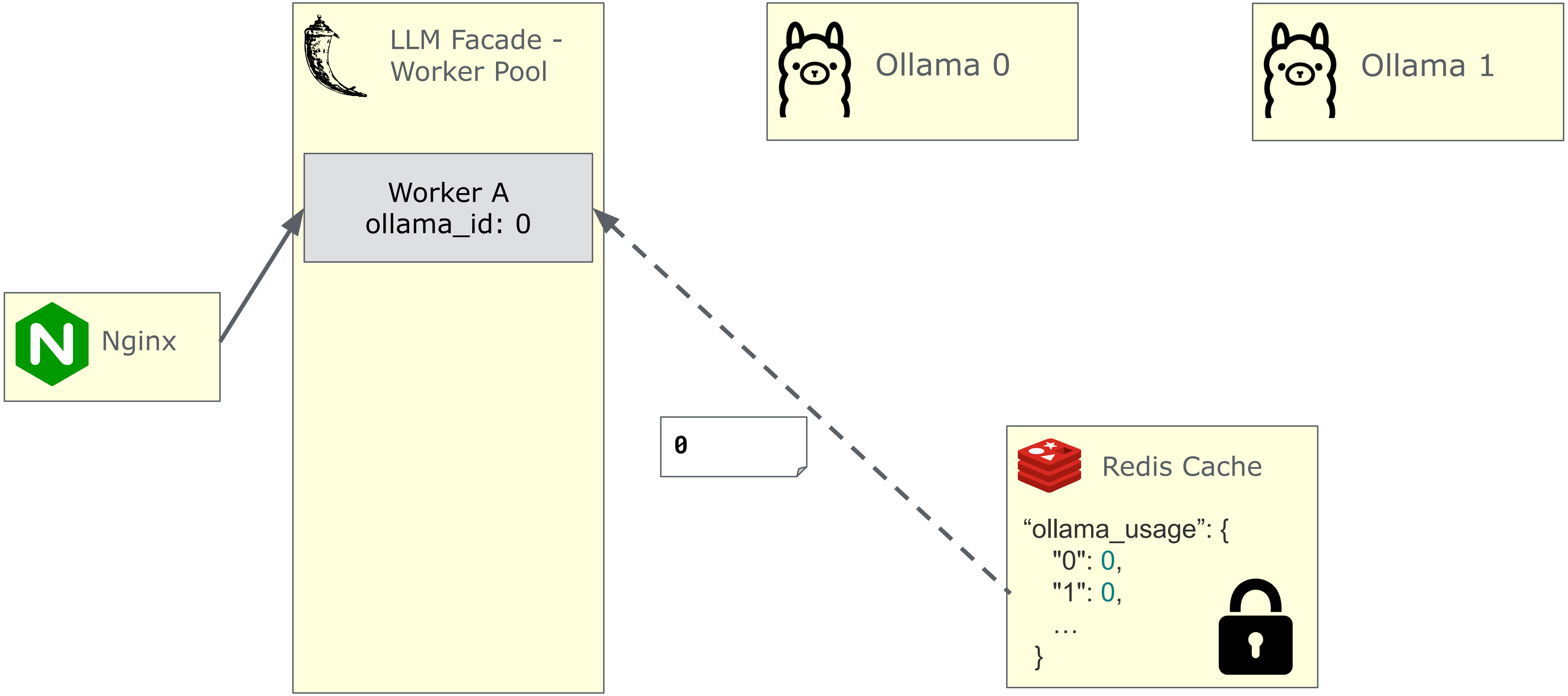
```
"ollama_usage": {
   "0": 0,
   "1": 0,
   …
 }
```

Simultaneous Example Generation



```
"ollama_usage": {
   "0": 0,
   "1": 0,
   …
 }
```

Simultaneous Example Generation



LLM Facade - Worker Pool

Ollama 0

Ollama 1

Worker A
ollama_id: 0

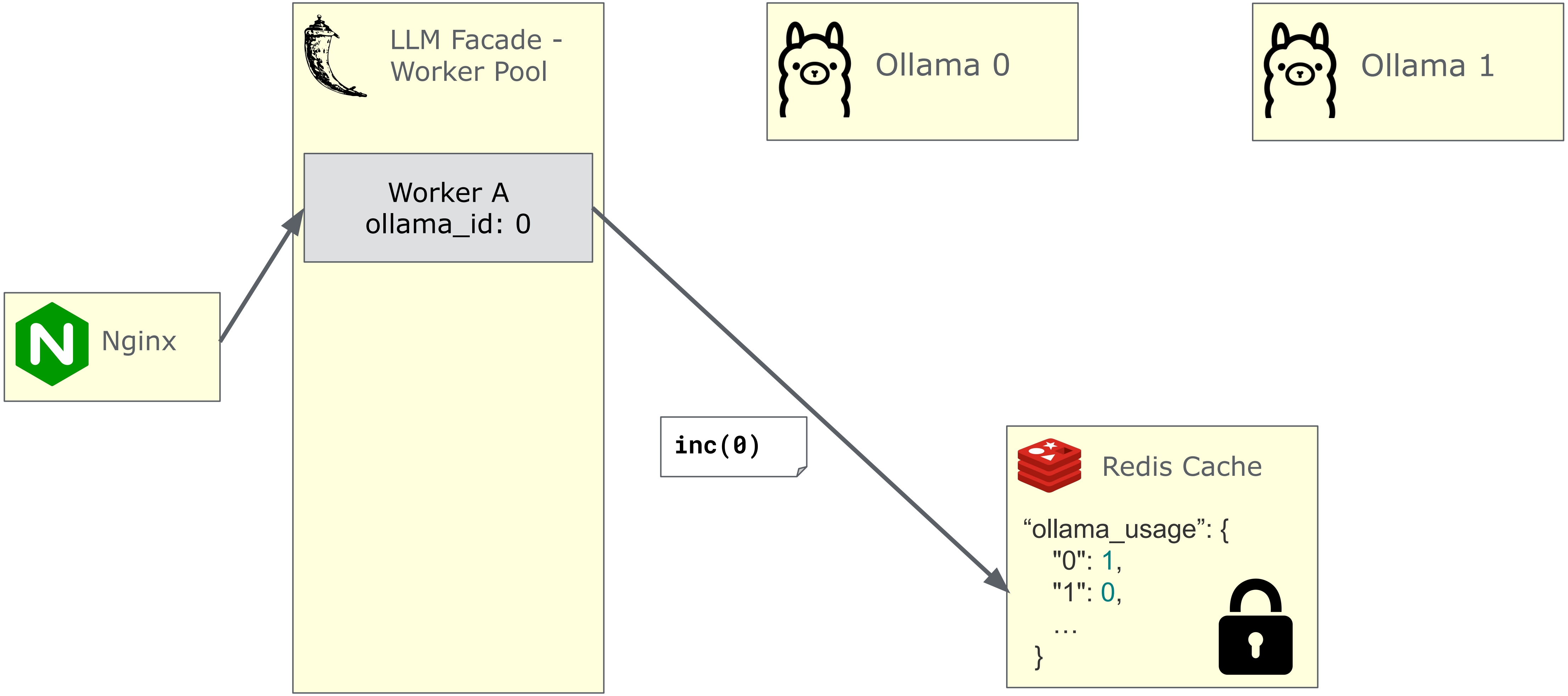
Nginx

`0`

Redis Cache

```
"ollama_usage": {
   "0": 0,
   "1": 0,
   …
 }
```

Simultaneous Example Generation



LLM Facade - Worker Pool

Worker A
ollama_id: 0

Ollama 0

Ollama 1

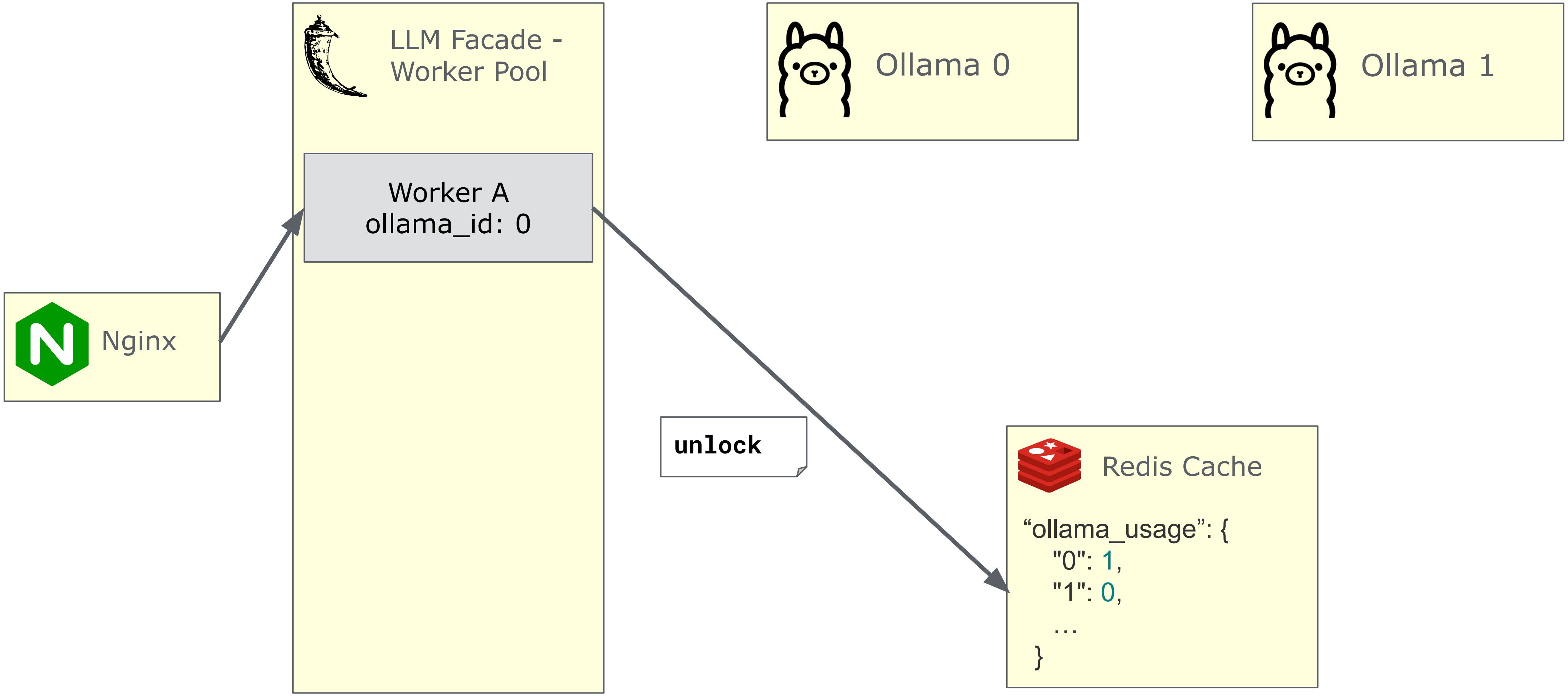
Nginx

`inc(0)`

Redis Cache

```
"ollama_usage": {
    "0": 1,
    "1": 0,
    …
 }
```

Simultaneous Example Generation



LLM Facade - Worker Pool

Ollama 0

Ollama 1

Worker A
ollama_id: 0

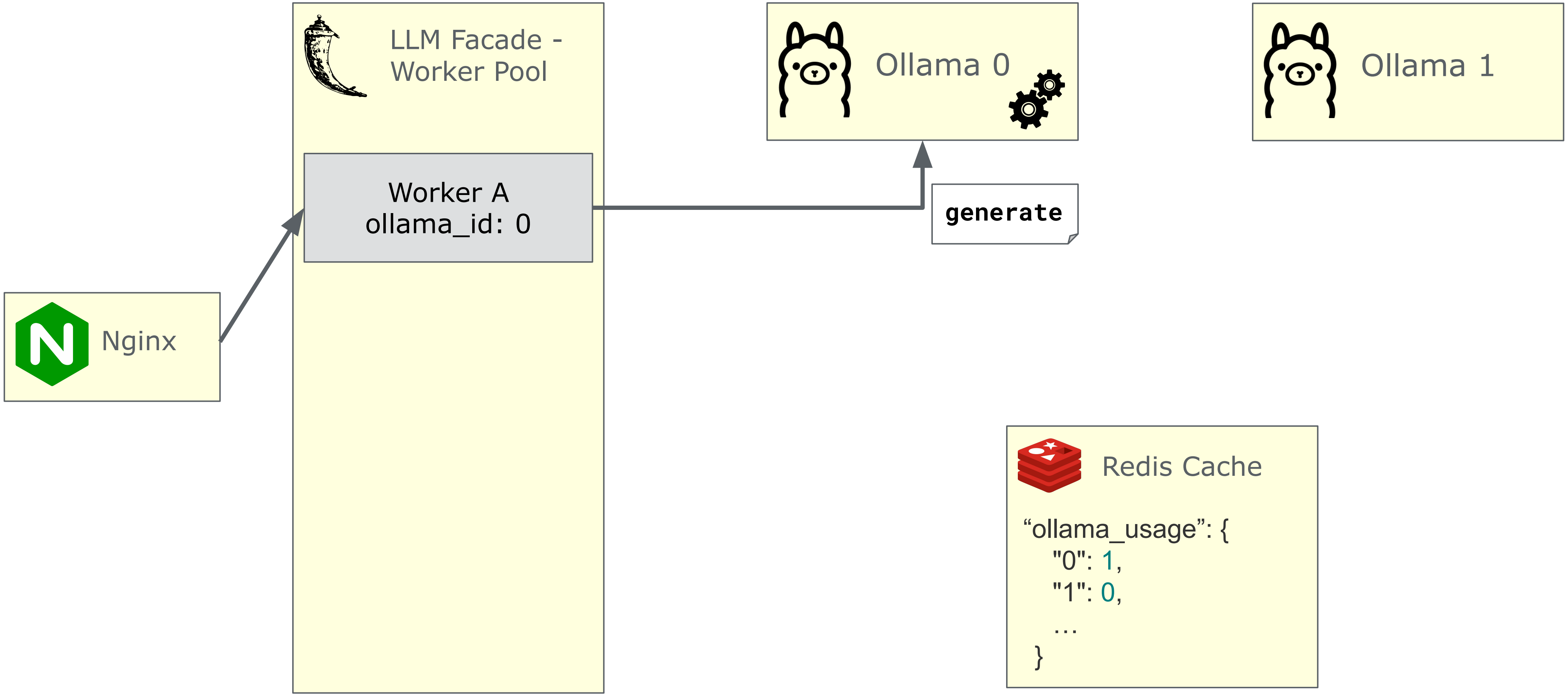
Nginx

`unlock`

Redis Cache

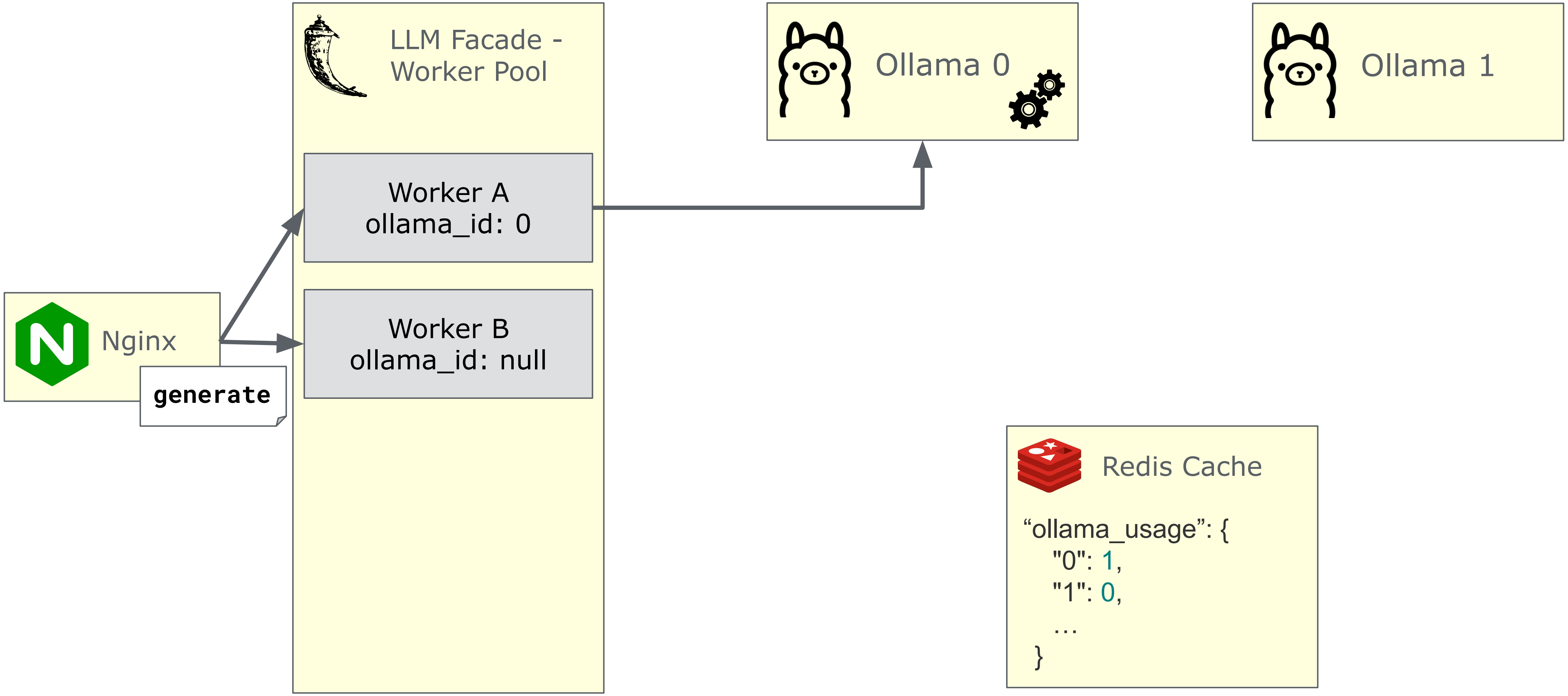
“ollama_usage”: {
  "0": 1,
  "1": 0,
  …
}

Simultaneous Example Generation



```
"ollama_usage": {
   "0": 1,
   "1": 0,
   …
 }
```

Simultaneous Example Generation



LLM Facade - Worker Pool

Worker A
ollama_id: 0

Worker B
ollama_id: null

Nginx

`generate`

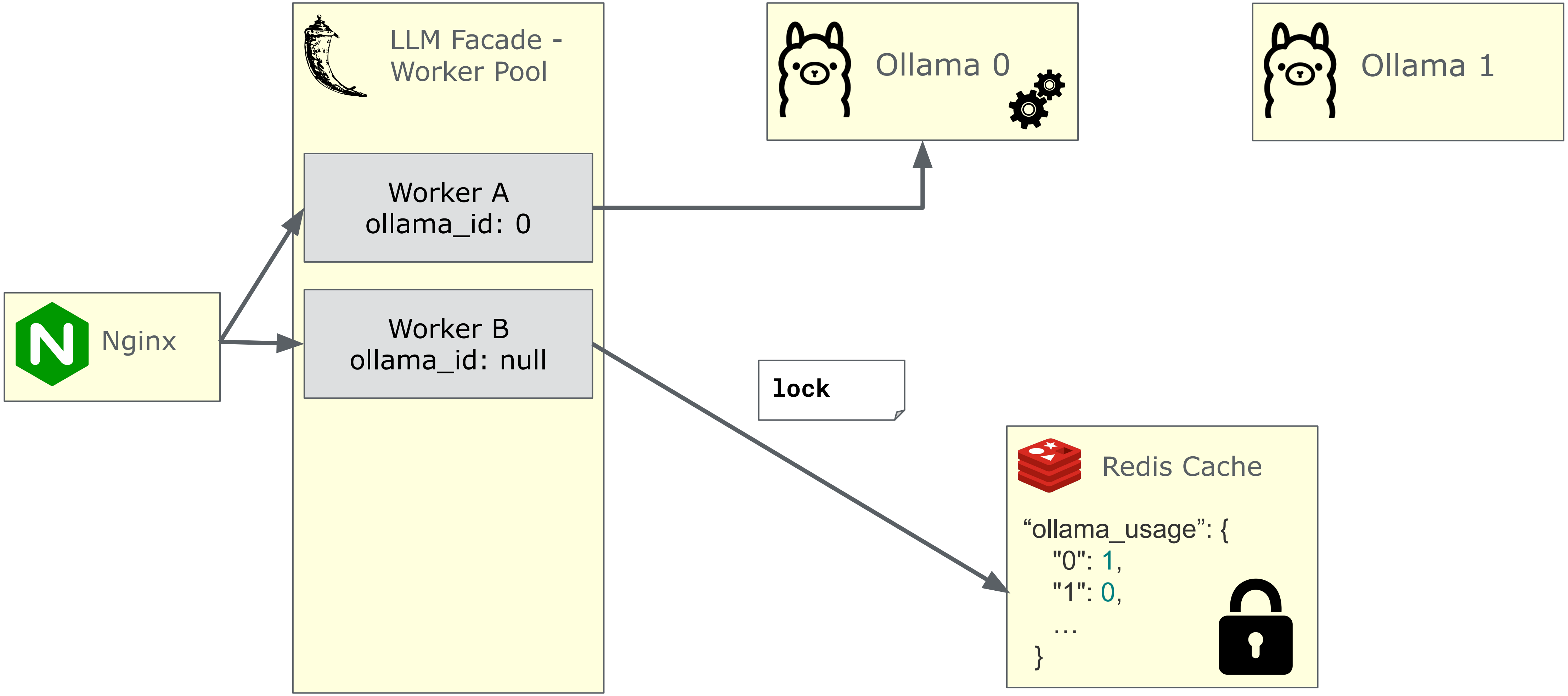
Ollama 0

Ollama 1

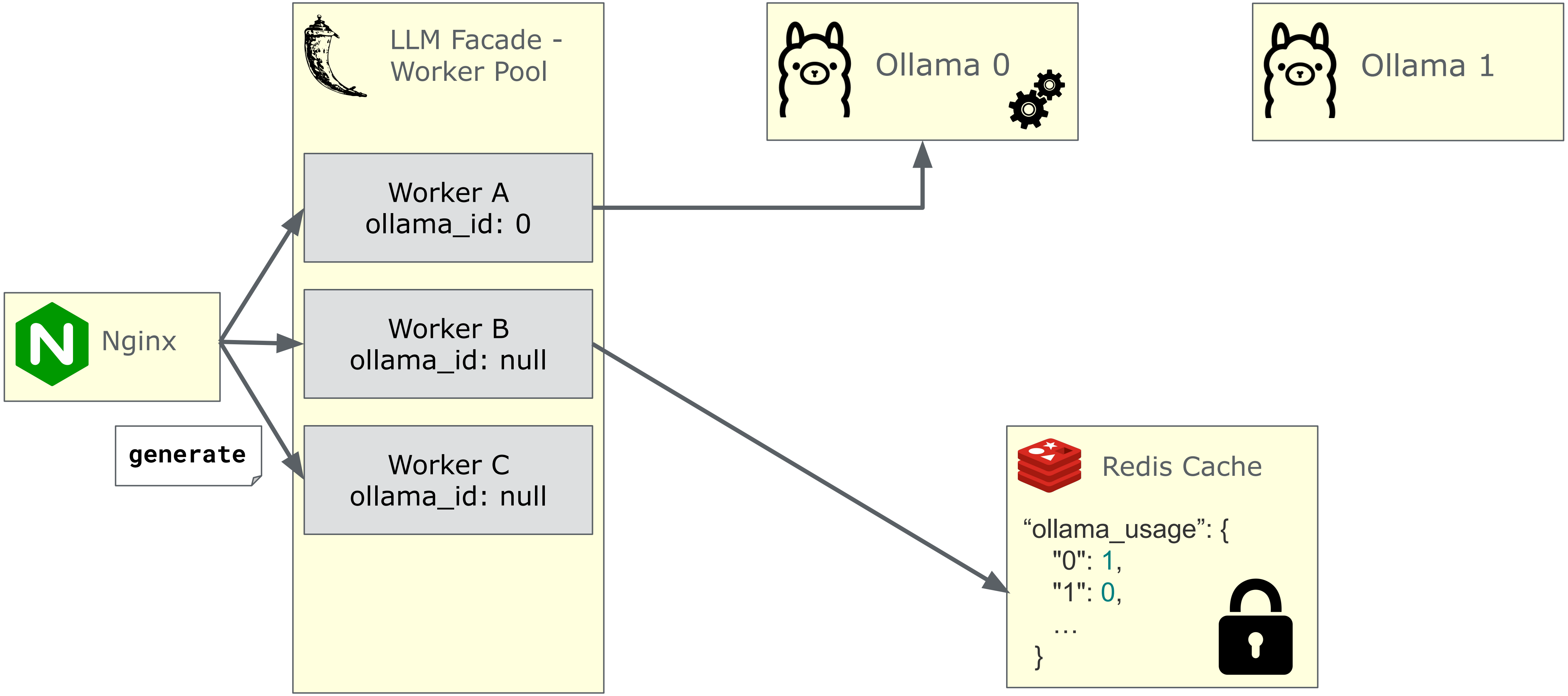
Redis Cache

```
“ollama_usage”: {
   "0": 1,
   "1": 0,
   …
 }
```

Simultaneous Example Generation



Simultaneous Example Generation



LLM Facade - Worker Pool

Worker A
ollama_id: 0

Worker B
ollama_id: null

Worker C
ollama_id: null

Ollama 0

Ollama 1

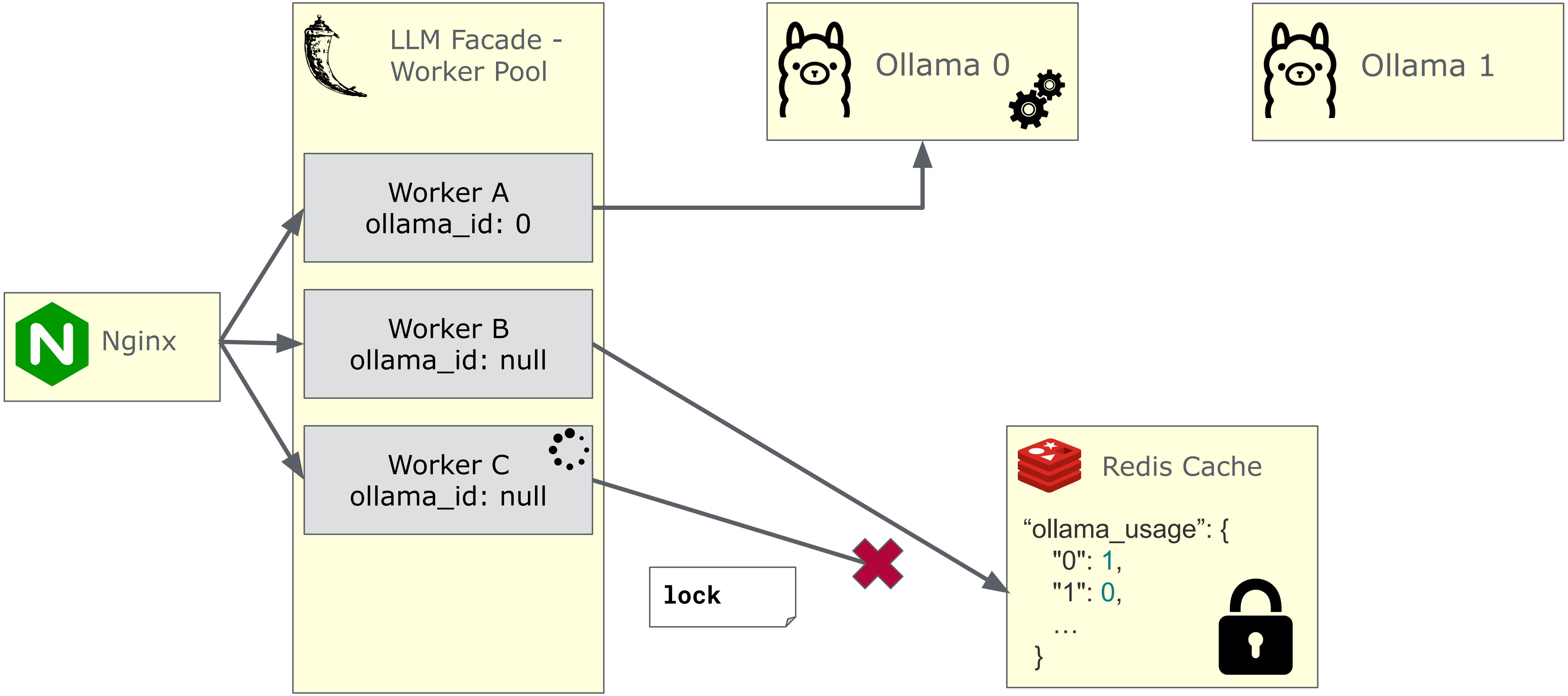
Nginx

`generate`

Redis Cache

```
"ollama_usage": {
   "0": 1,
   "1": 0,
   …
 }
```

Simultaneous Example Generation



Nginx

LLM Facade - Worker Pool

Worker A
ollama_id: 0

Worker B
ollama_id: null

Worker C
ollama_id: null

Ollama 0

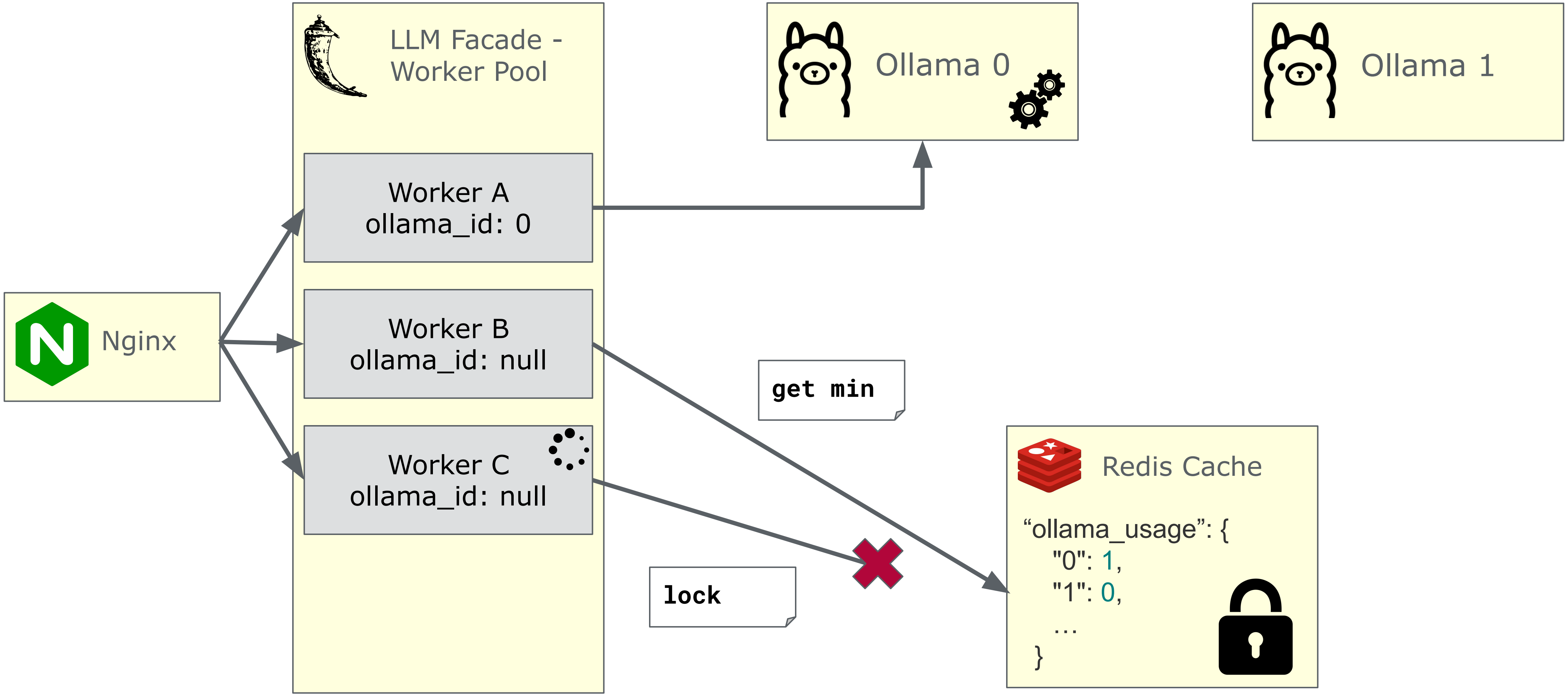
Ollama 1

`lock`

Redis Cache

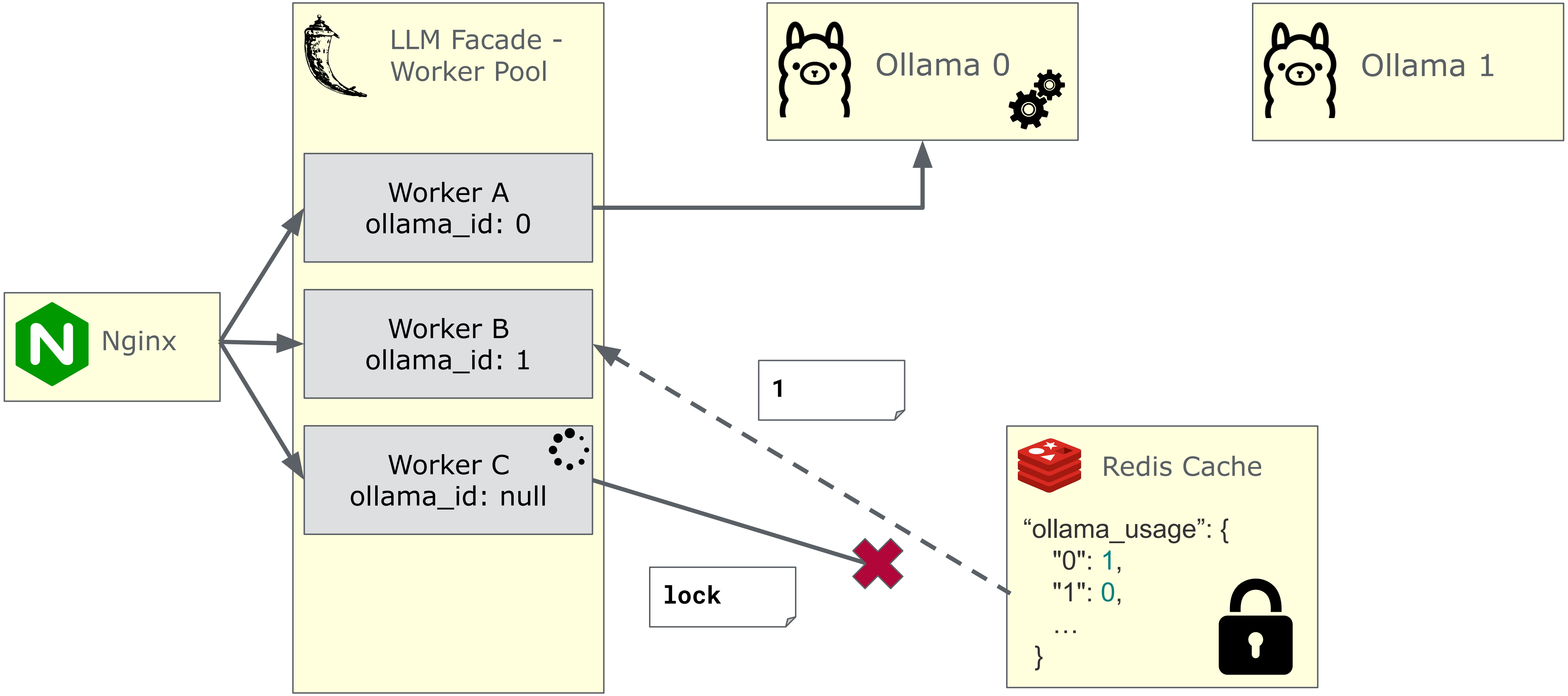
```
“ollama_usage”: {
   "0": 1,
   "1": 0,
   …
 }
```

Simultaneous Example Generation



```
“ollama_usage”: {
   "0": 1,
   "1": 0,
   …
 }
```

Simultaneous Example Generation



LLM Facade -
Worker Pool

Worker A
ollama_id: 0

Worker B
ollama_id: 1

Worker C
ollama_id: null

Ollama 0

Ollama 1

Nginx

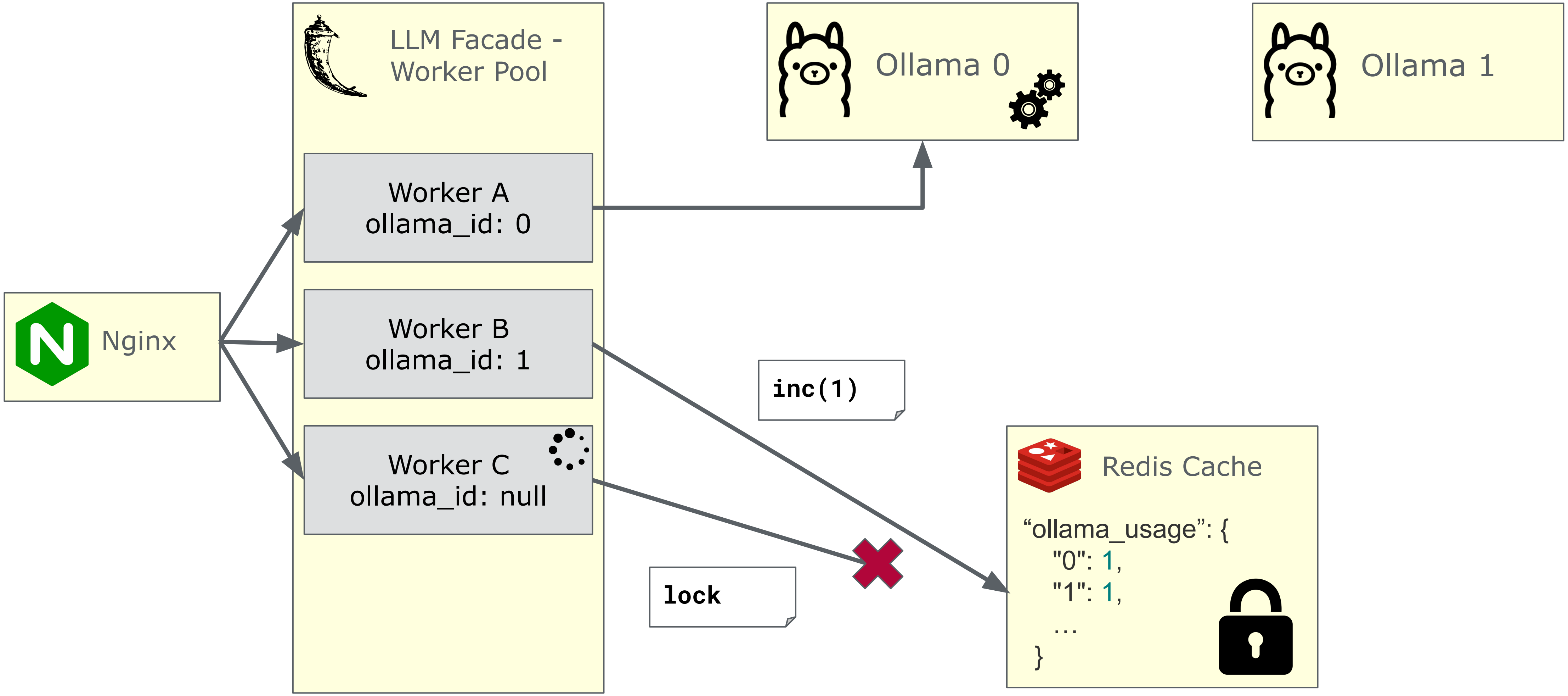
1

lock

Redis Cache

“ollama_usage”: {
"0": 1,
"1": 0,
…
}

Simultaneous Example Generation



LLM Facade - Worker Pool

Worker A
ollama_id: 0

Worker B
ollama_id: 1

Worker C
ollama_id: null

Nginx

Ollama 0

Ollama 1

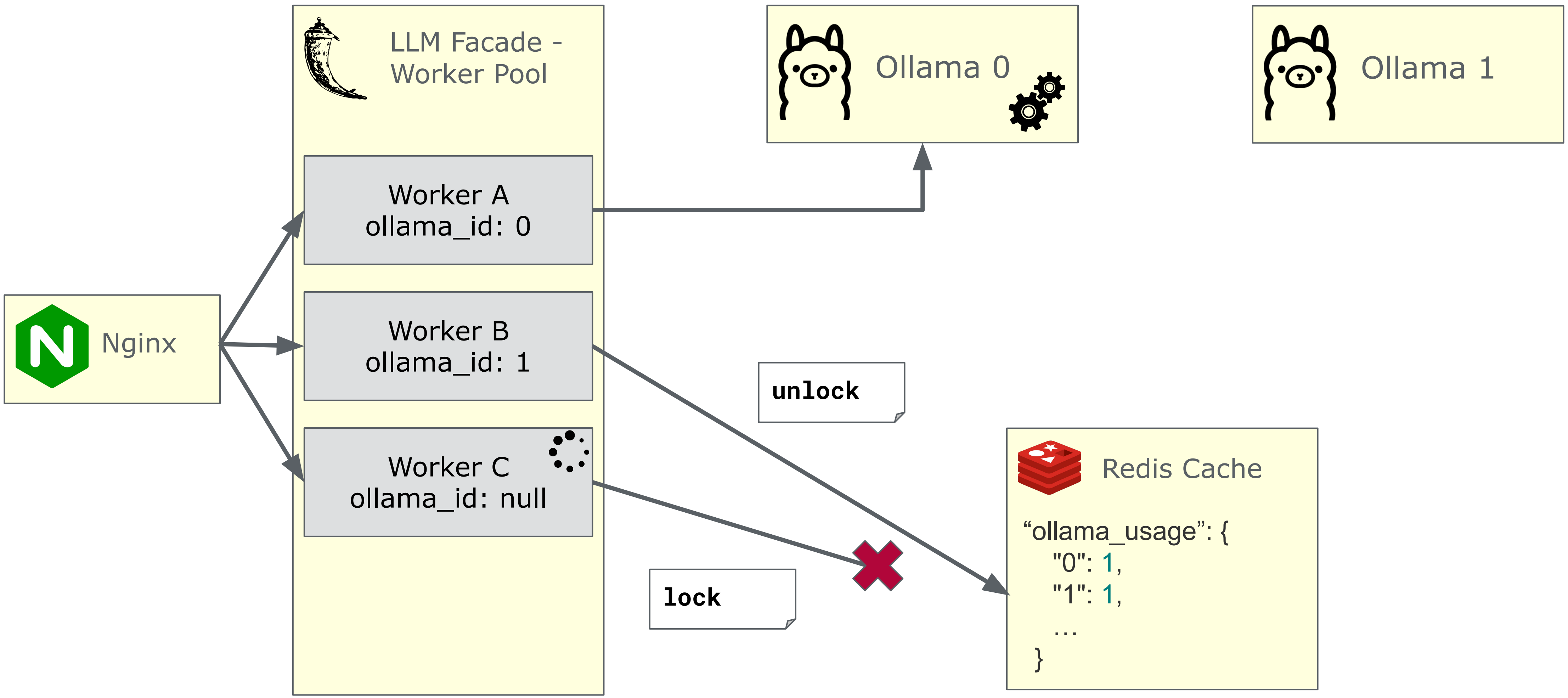
`inc(1)`

`lock`

Redis Cache

```
“ollama_usage”: {
   "0": 1,
   "1": 1,
   …
 }
```

Simultaneous Example Generation



Nginx

LLM Facade -
Worker Pool

Worker A
ollama_id: 0

Worker B
ollama_id: 1

Worker C
ollama_id: null

Ollama 0

Ollama 1

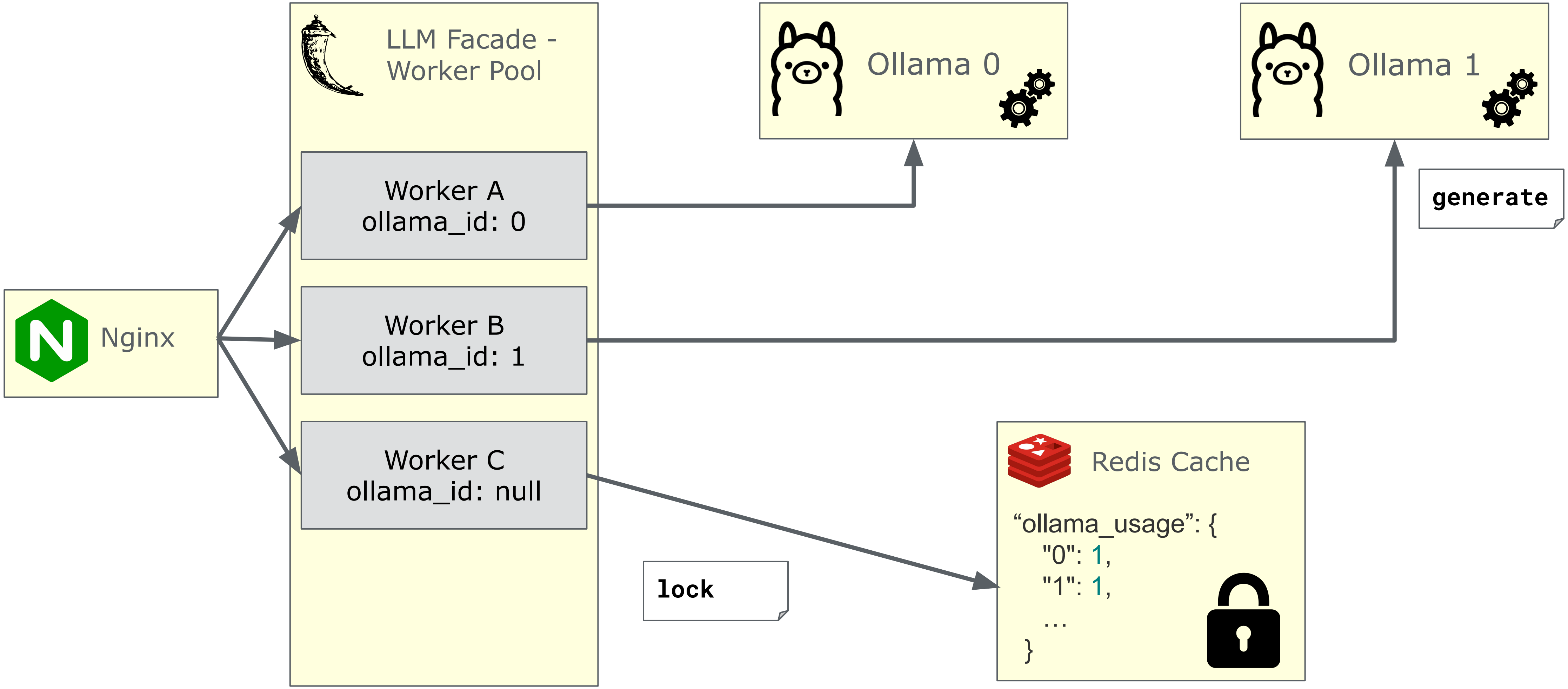
unlock

lock

Redis Cache

“ollama_usage”: {
"0": 1,
"1": 1,
…
}

Simultaneous Example Generation



LLM Facade - Worker Pool

Worker A
ollama_id: 0

Worker B
ollama_id: 1

Worker C
ollama_id: null

Nginx

Ollama 0

Ollama 1

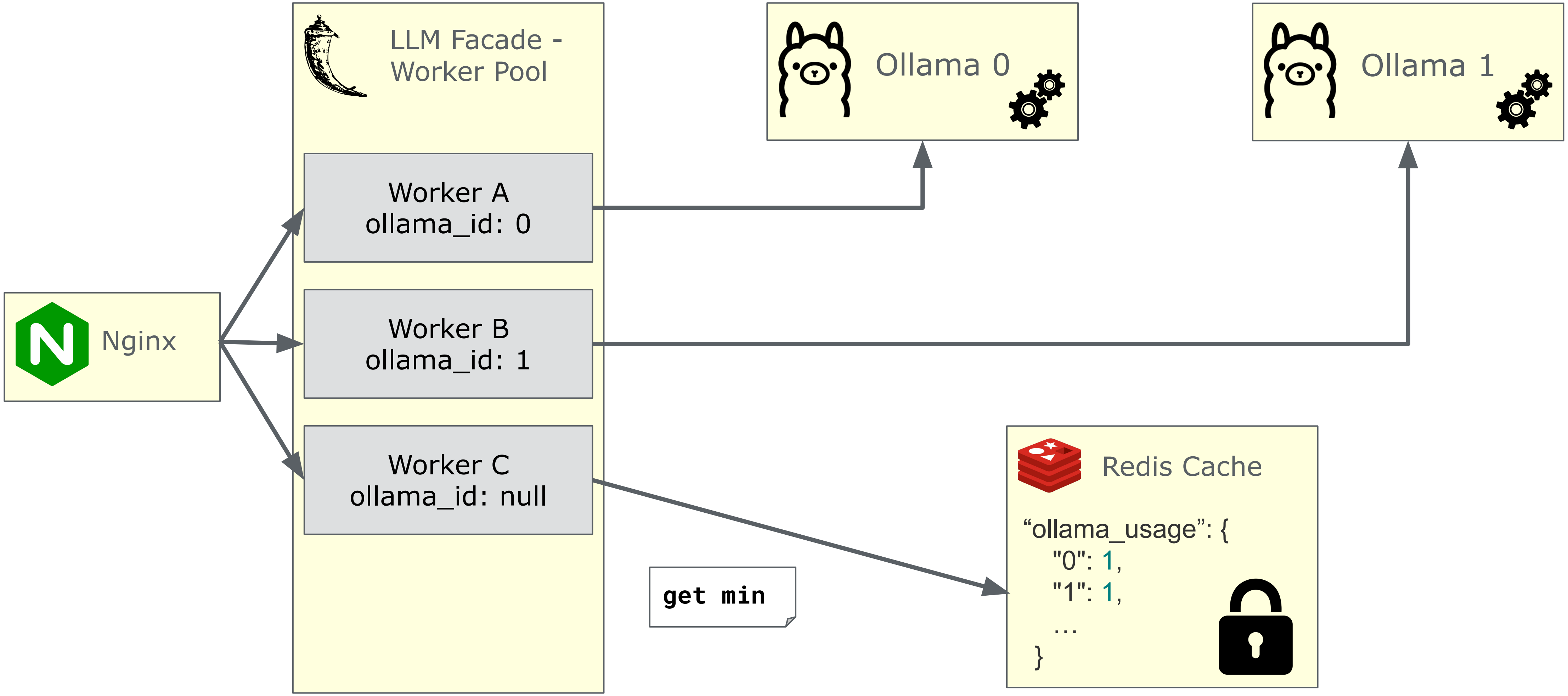
`generate`

`lock`

Redis Cache

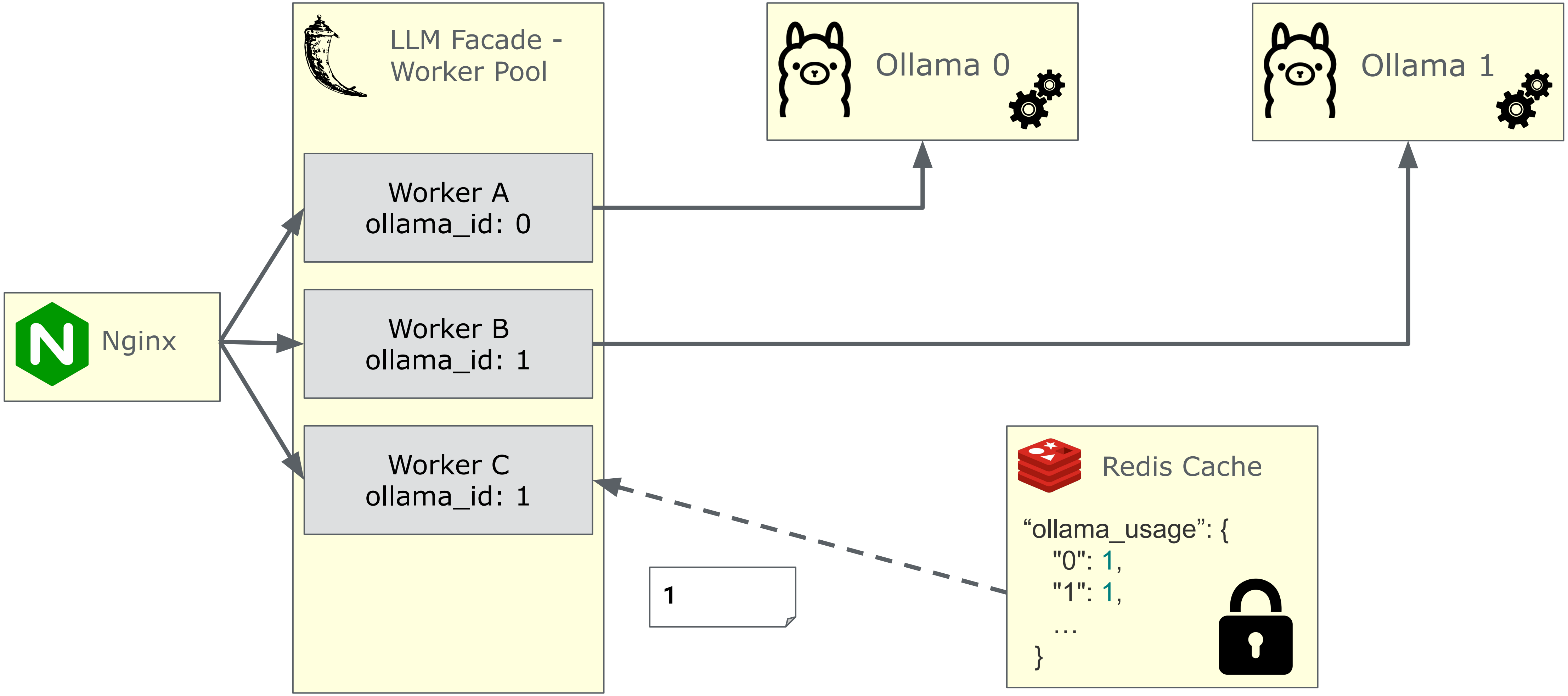
“ollama_usage”: {
"0": 1,
"1": 1,
…
}

Simultaneous Example Generation



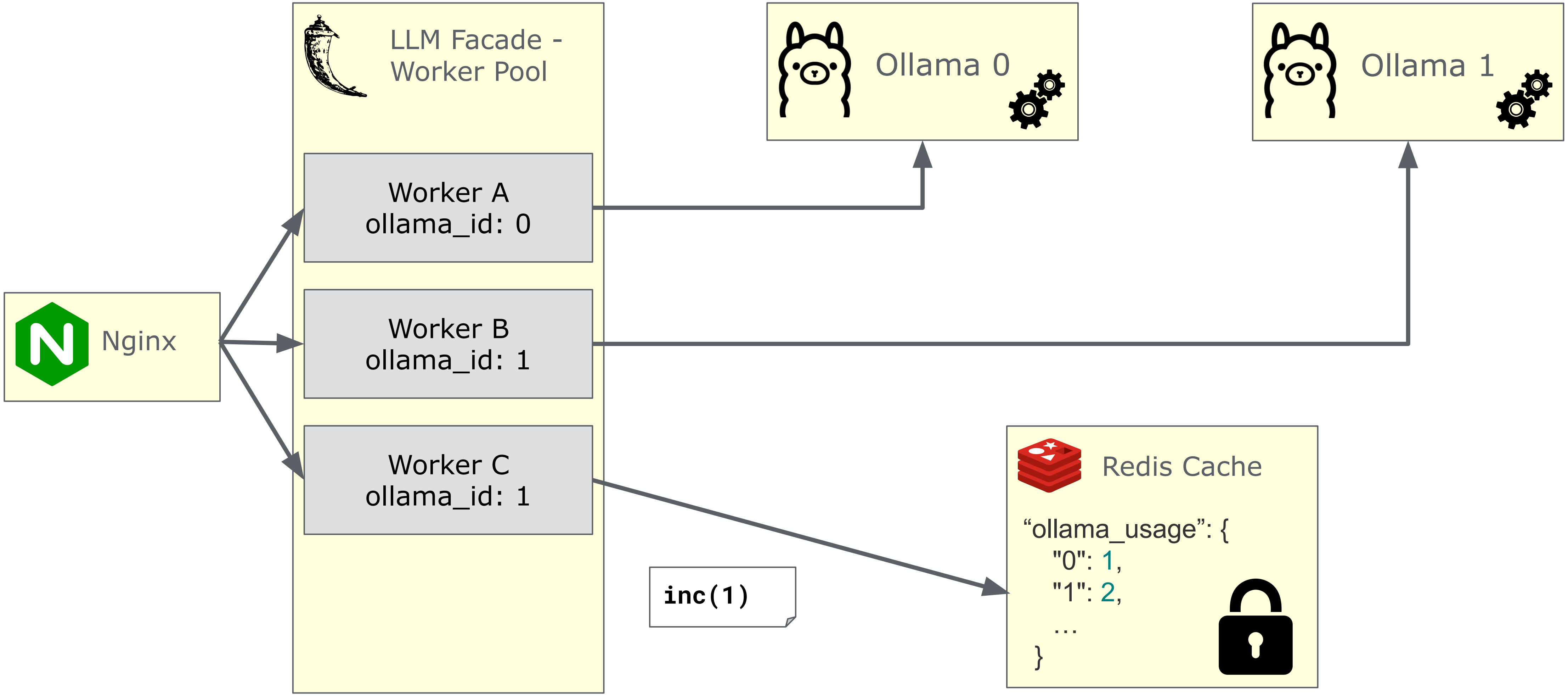
```
“ollama_usage”: {
   "0": 1,
   "1": 1,
   …
 }
```

Simultaneous Example Generation



```
“ollama_usage”: {
   "0": 1,
   "1": 1,
   …
 }
```

Simultaneous Example Generation



Nginx

LLM Facade - Worker Pool

Worker A
ollama_id: 0

Worker B
ollama_id: 1

Worker C
ollama_id: 1

Ollama 0

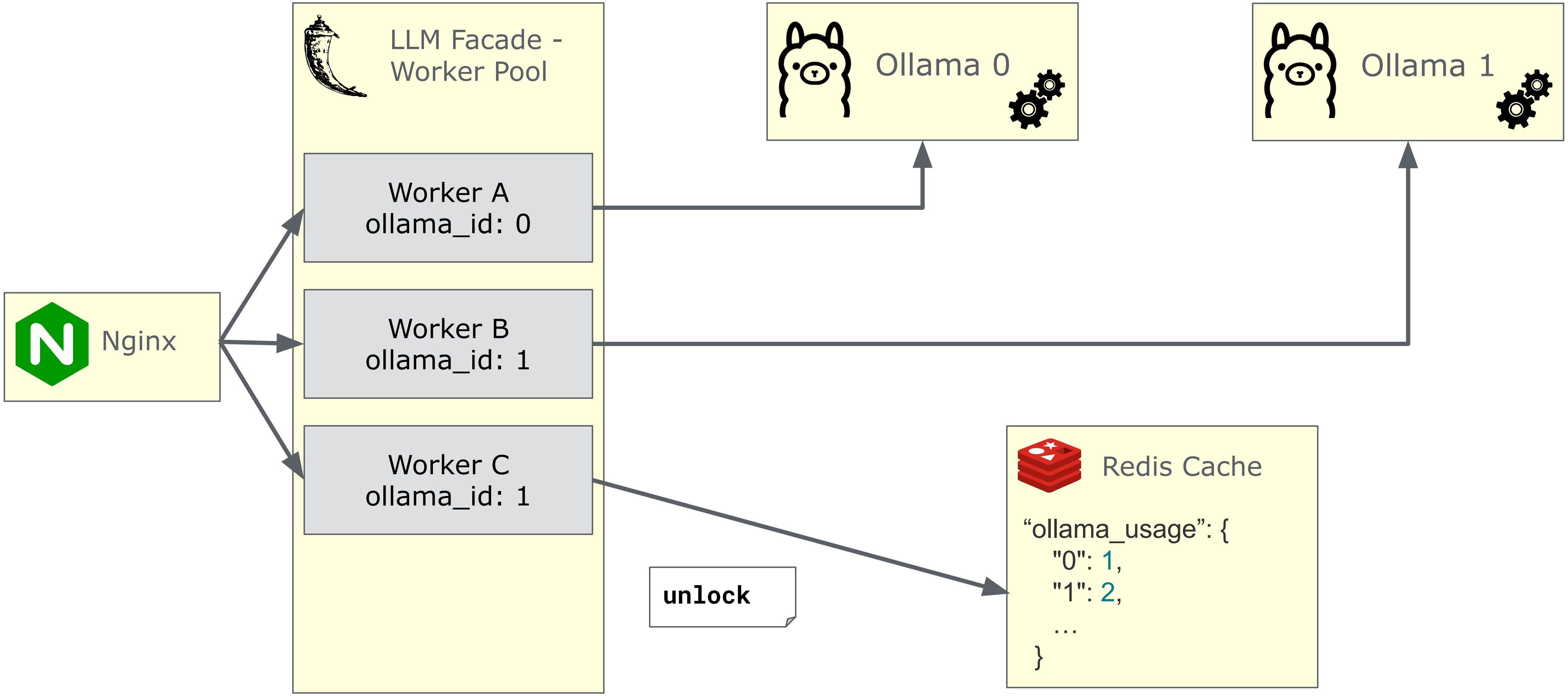
Ollama 1

`inc(1)`

Redis Cache

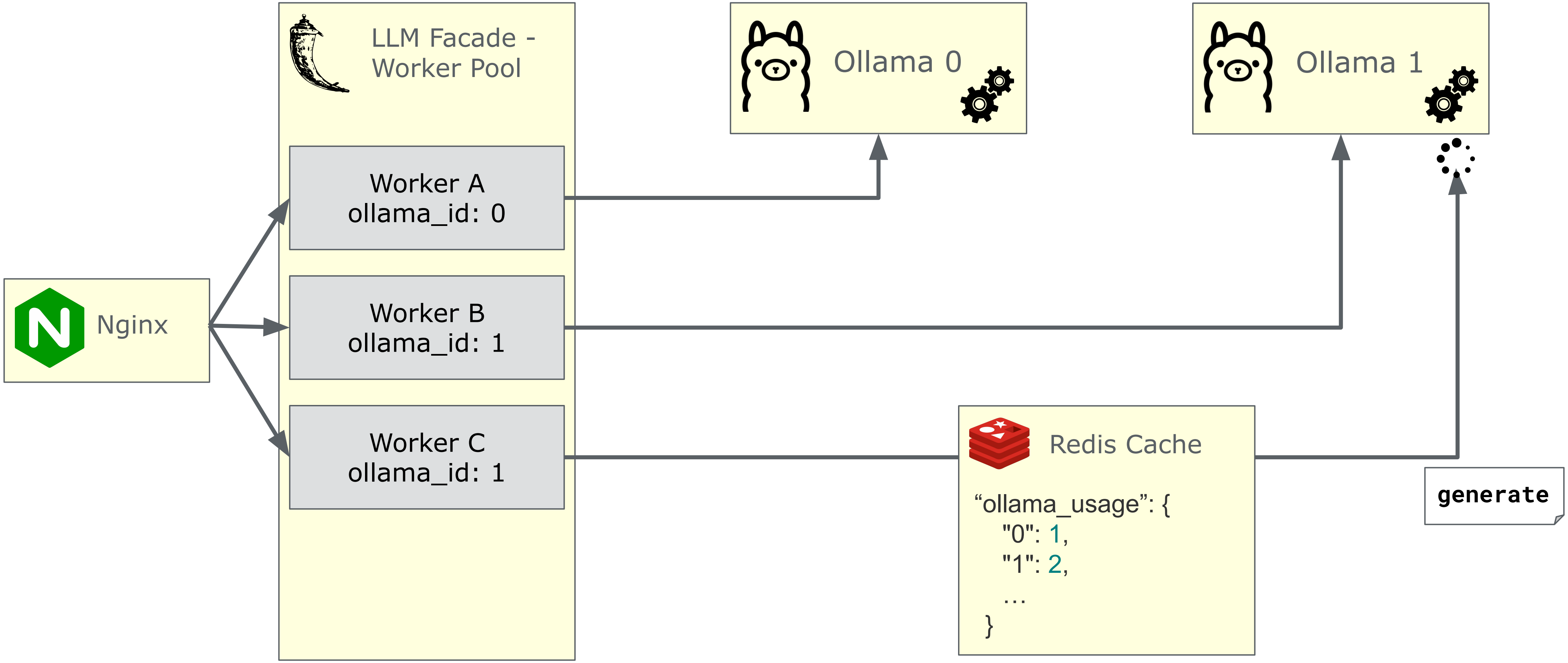
```
“ollama_usage”: {
   "0": 1,
   "1": 2,
   …
 }
```

Simultaneous Example Generation



```
“ollama_usage”: {
   "0": 1,
   "1": 2,
   …
 }
```

Simultaneous Example Generation



LLM Facade - Worker Pool

Worker A
ollama_id: 0

Worker B
ollama_id: 1

Worker C
ollama_id: 1

Nginx

Ollama 0

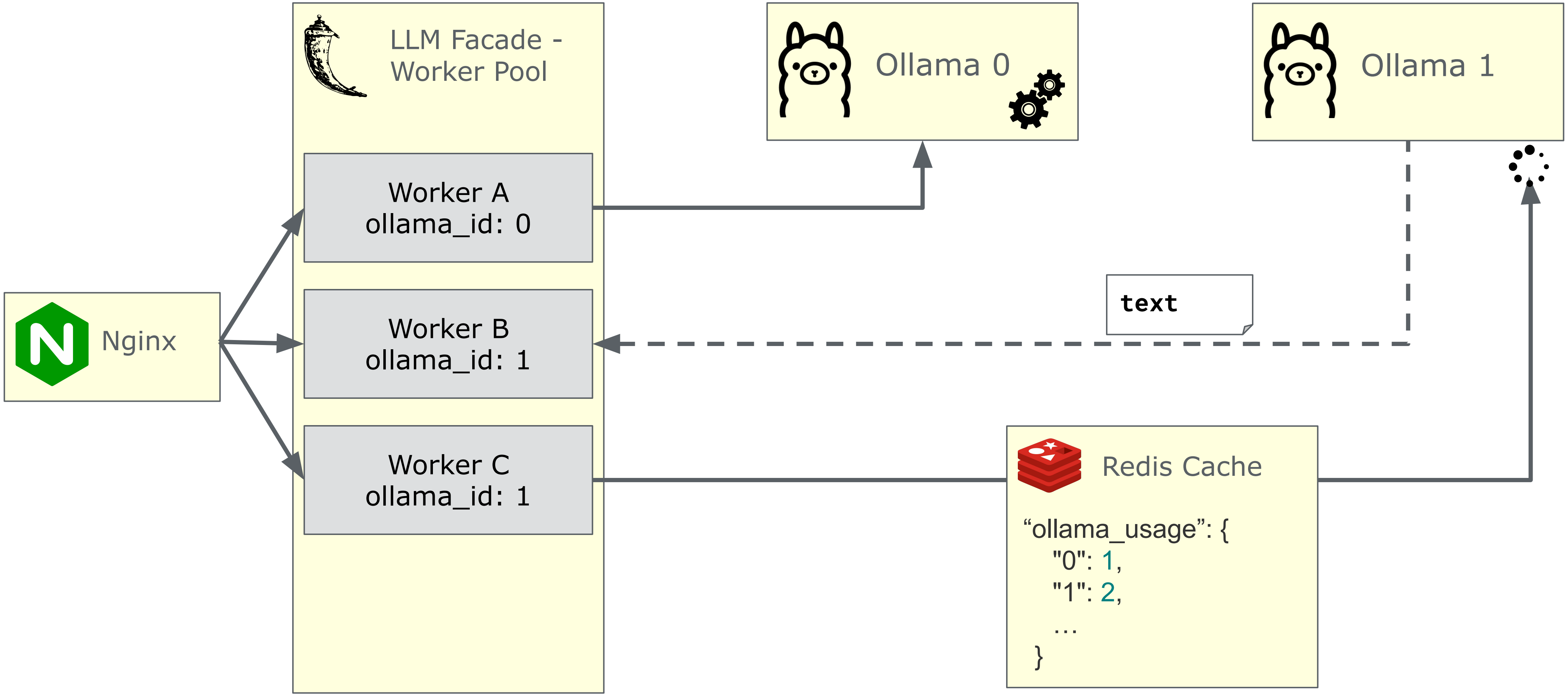
Ollama 1

Redis Cache

```
“ollama_usage”: {
   "0": 1,
   "1": 2,
   …
 }
```

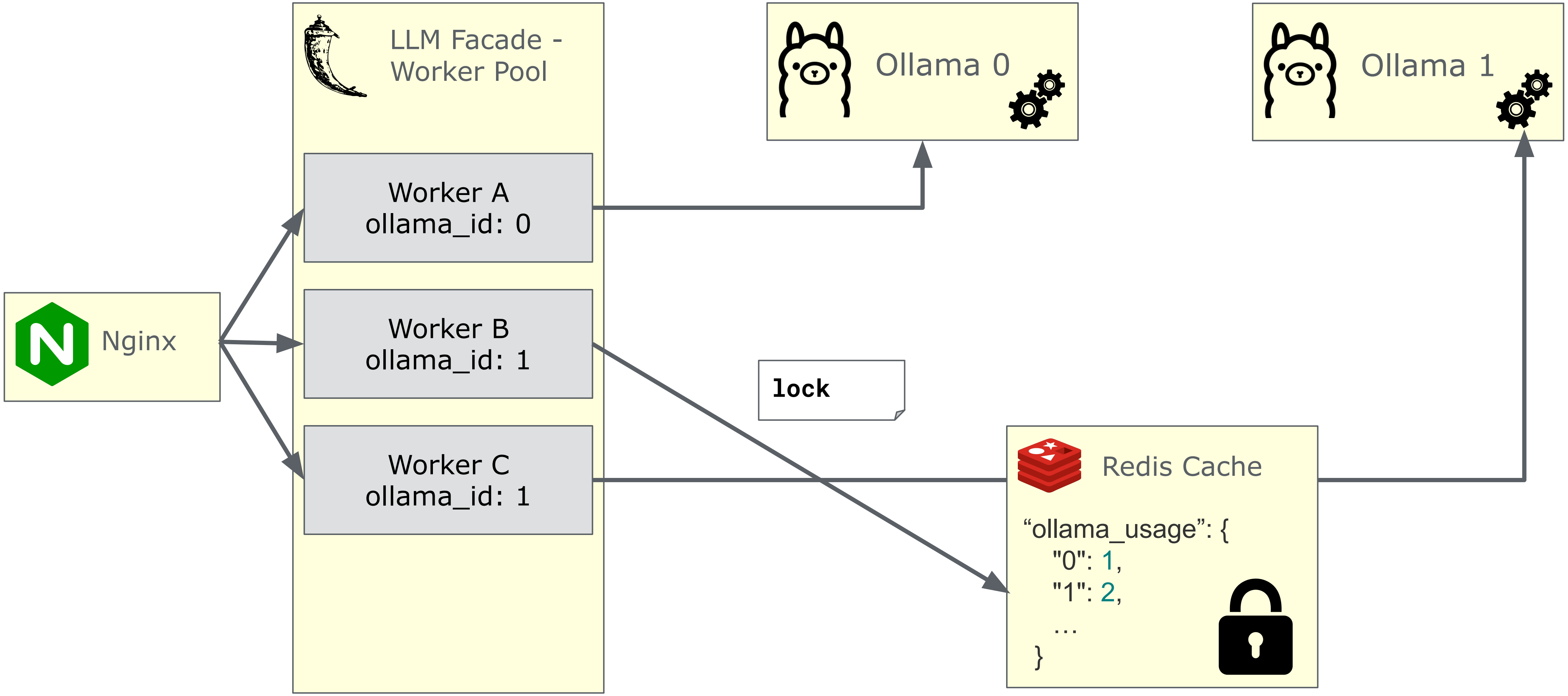
`generate`

Simultaneous Example Generation



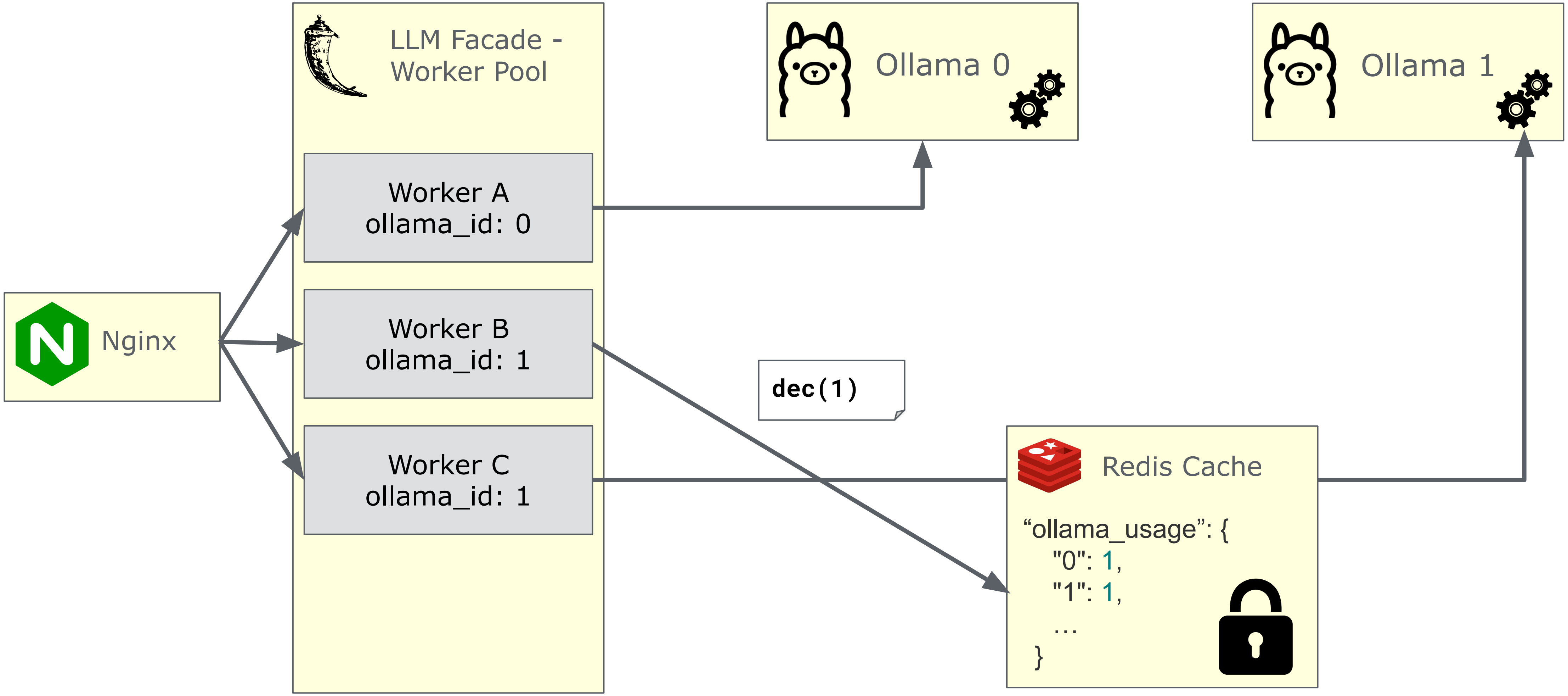
```
"ollama_usage": {
   "0": 1,
   "1": 2,
   …
 }
```

Simultaneous Example Generation



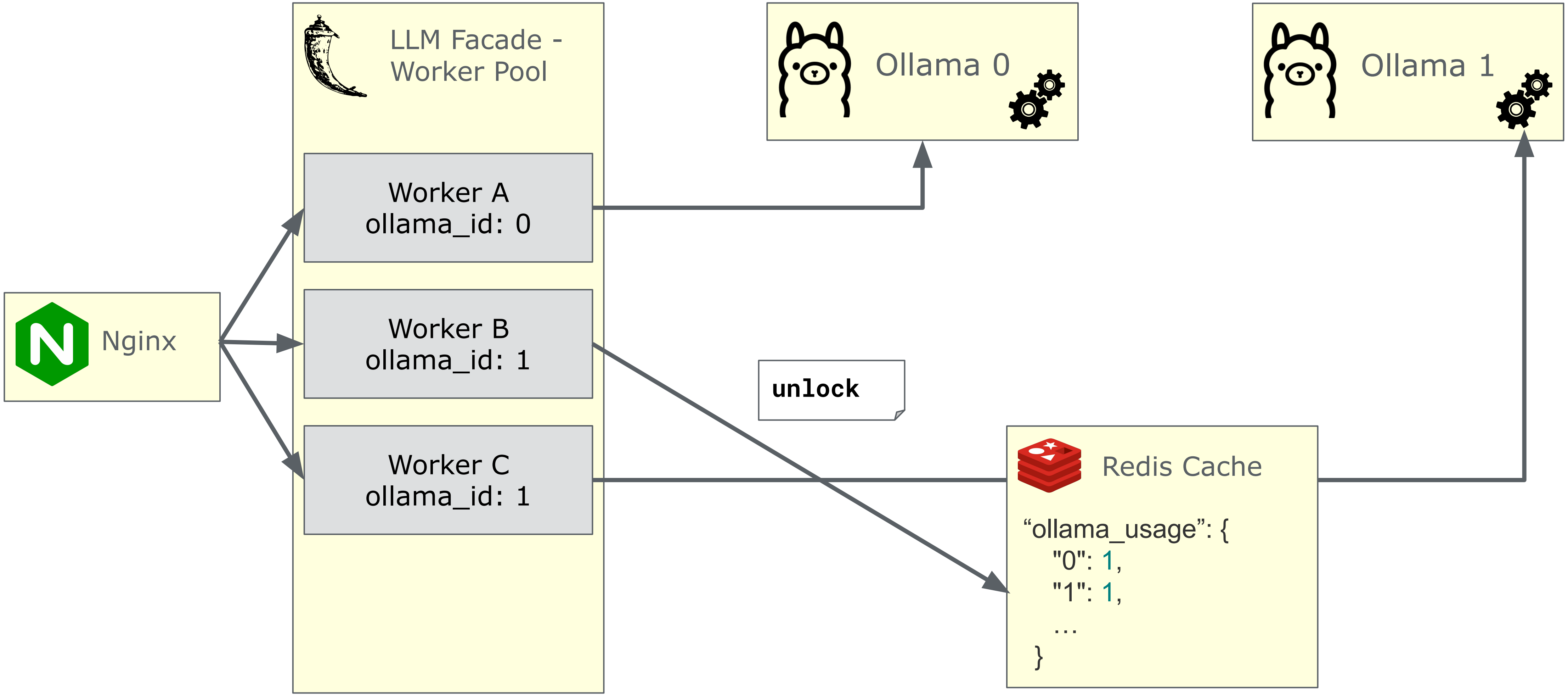
```
"ollama_usage": {
   "0": 1,
   "1": 2,
   …
}
```

Simultaneous Example Generation



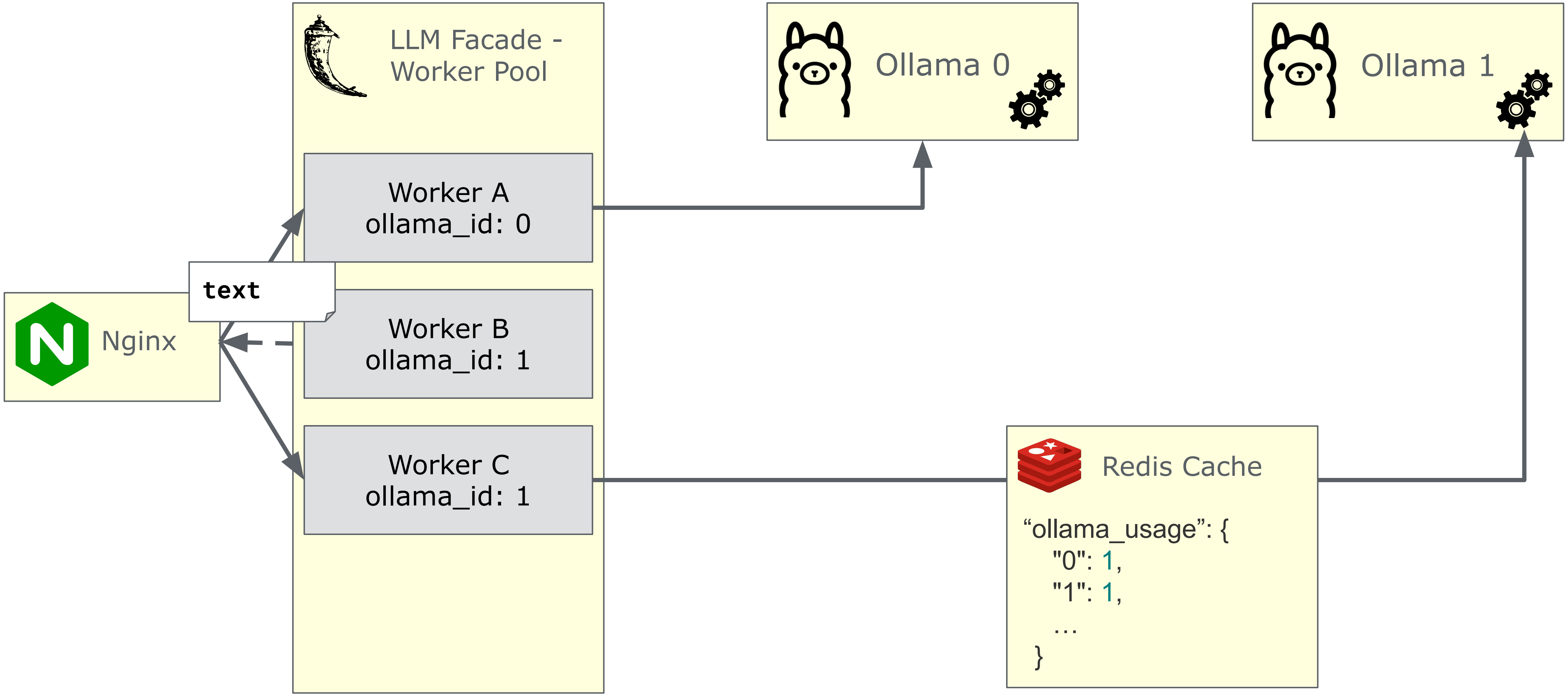
```
"ollama_usage": {
   "0": 1,
   "1": 1,
   ...
 }
```

Simultaneous Example Generation



```
“ollama_usage”: {
   "0": 1,
   "1": 1,
   …
 }
```

Simultaneous Example Generation



LLM Facade - Worker Pool

Worker A
ollama_id: 0

Worker B
ollama_id: 1

Worker C
ollama_id: 1

`text`

Nginx

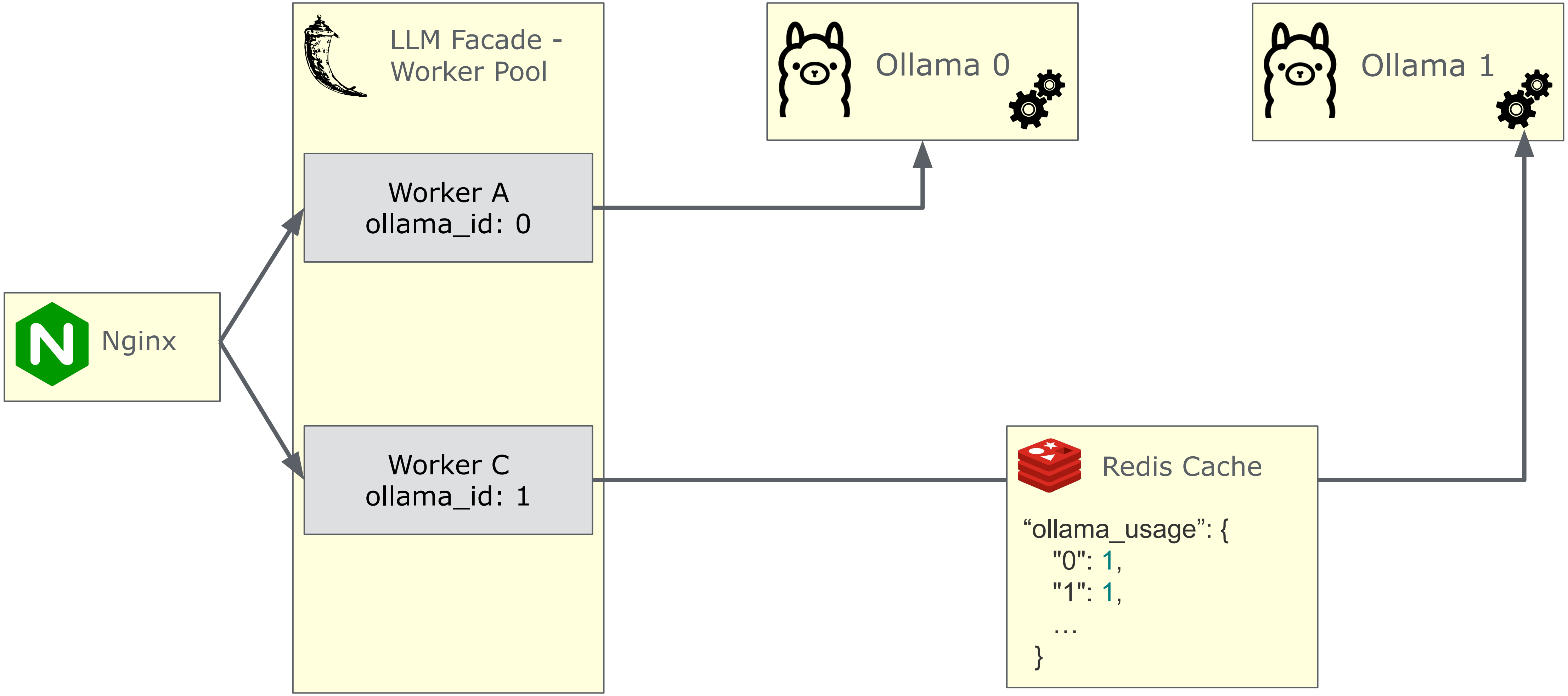
Ollama 0

Ollama 1

Redis Cache

```
“ollama_usage”: {
   "0": 1,
   "1": 1,
   …
 }
```

Simultaneous Example Generation



Nginx

LLM Facade - Worker Pool

Worker A
ollama_id: 0

Worker C
ollama_id: 1

Ollama 0

Ollama 1

Redis Cache

```
“ollama_usage”: {
   "0": 1,
   "1": 1,
   …
 }
```

Challenges

Code Related

- Simultaneous Generation
- Syntax Highlighting
- Prompt Engineering to avoid explanation in code
- Hard to find suitable metrics
    - Code Length
    - Code Complexity
    - Maintainability

Project Related

- User Feedback: Server restricted to HPI network
- Linux and Windows incompatibilities

Challenges

Code Related

"You are as experienced as a senior software developer. You will get a piece of code documentation and the source code for this documentation.
Your task is to provide a helpful code example for the documentation. You are encouraged to use also other functions from the same framework to demonstrate interoperability between them if it is appropriate.

You ONLY output python code. If you output anything else than python code 100 cute little rabbits will DIE. You DON'T under any circumstances output any explanation of the code."

- ○ Maintainability

Challenges

Code Related

- Simultaneous Generation
- Syntax Highlighting
- Prompt Engineering to avoid explanation in code
- Hard to find suitable metrics
    - Code Length
    - Code Complexity
    - Maintainability

Project Related

- User Feedback: Server restricted to HPI network
- Linux and Windows incompatibilities

Learnings / Reflection

New Technologies

ollama

docker

redis

python

Vue.js

Project Related

- Improve Collective Code ownership
- Two development phases
- Focus on user experience / story

User Feedback

Methodology

- Introduction to subject
- ~15 Minutes
- Communicated incomplete features

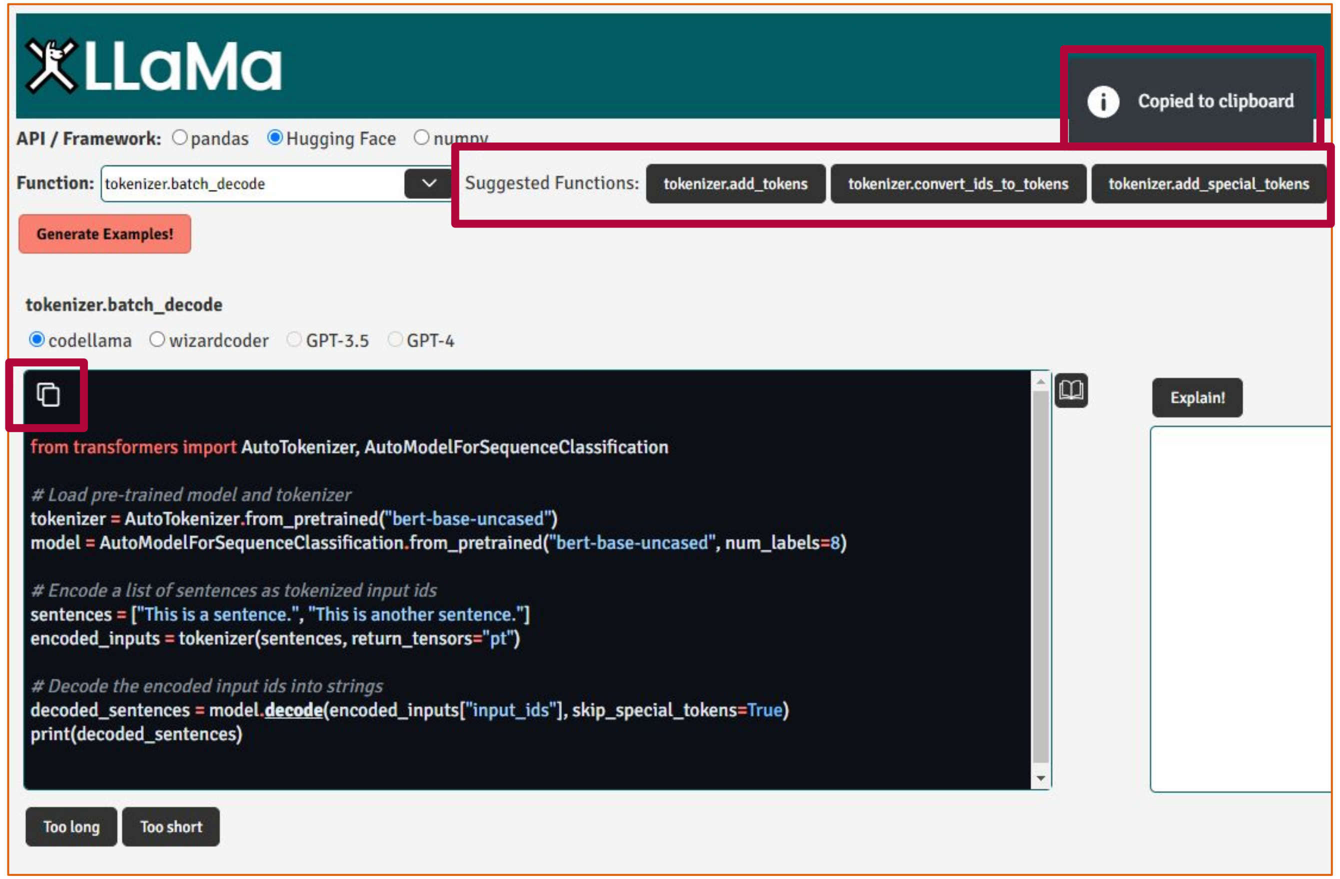
User Feedback

Methodology

- Introduction to subject
- ~15 Minutes
- Communicated incomplete features

Implemented

- Confusing function buttons
- Copy button should give visual feedback



API / Framework: ○ pandas ● Hugging Face ○ numpy

Function: tokenizer.batch_decode

Suggested Functions: tokenizer.add_tokens tokenizer.convert_ids_to_tokens tokenizer.add_special_tokens

tokenizer.batch_decode

● codellama ○ wizardcoder ○ GPT-3.5 ○ GPT-4

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification

# Load pre-trained model and tokenizer
tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
model = AutoModelForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=8)

# Encode a list of sentences as tokenized input ids
sentences = ["This is a sentence.", "This is another sentence."]
encoded_inputs = tokenizer(sentences, return_tensors="pt")

# Decode the encoded input ids into strings
decoded_sentences = model.decode(encoded_inputs["input_ids"], skip_special_tokens=True)
print(decoded_sentences)
```

User Feedback

User 1 (Molecular Biology PhD Student)

- Documentation button very helpful
- Did not notice clickable button in example
- More advanced options for generated examples and layout
- History of generated examples
- Would even just use it to read documentation, nice collection of different frameworks

User 2 (Machine Learning Engineer)

- Currently learning new framework with bad documentation, so tool would be useful if all functions were included

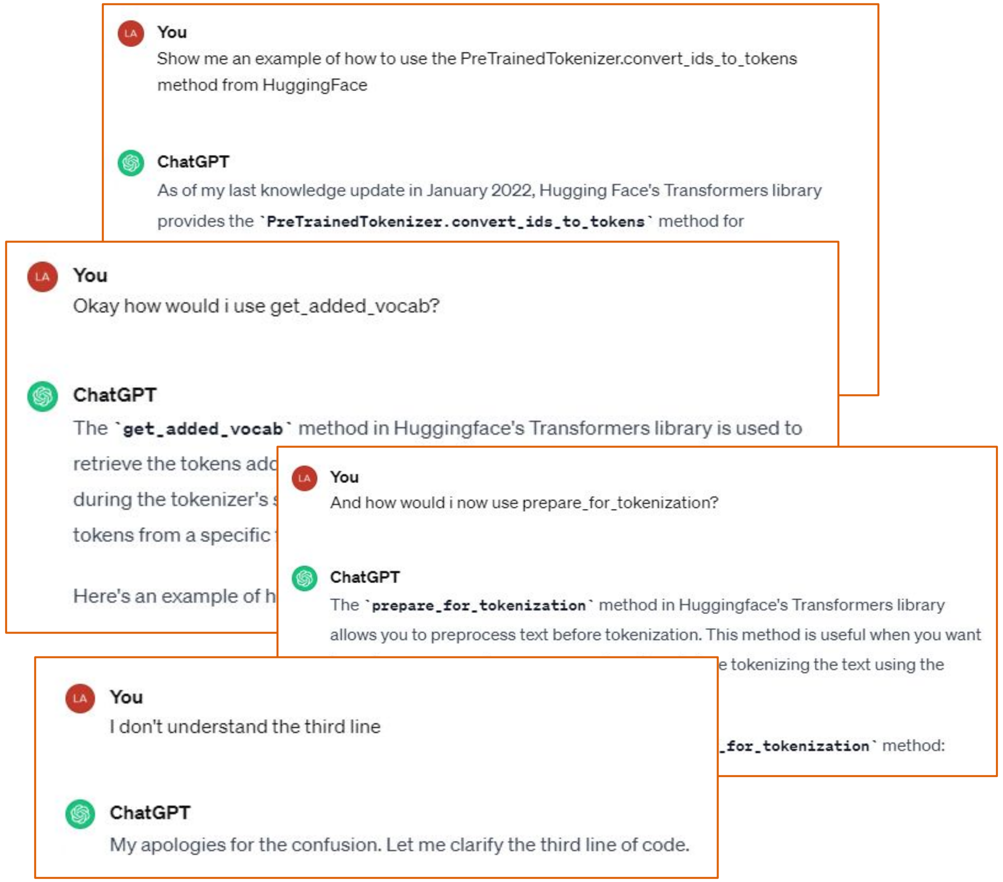
User 3 (Software Developer)

- Would use when learning new API
- Would be nice to have for programming languages, e.g. Java
- Nice to have everything in one place

Idea

Chatbots

- Helpful when working and learning new tools, but often cumbersome to use
  - Might need long prompts ✅
  - First results may be unhelpful (✅)
  - Multiple follow up prompts can be annoying ✅
- Different LLMs give different results (of different quality) ✅



You
Show me an example of how to use the PreTrainedTokenizer.convert_ids_to_tokens method from HuggingFace

ChatGPT
As of my last knowledge update in January 2022, Hugging Face's Transformers library provides the `PreTrainedTokenizer.convert_ids_to_tokens` method for

You
Okay how would i use get_added_vocab?

ChatGPT
The `get_added_vocab` method in Huggingface's Transformers library is used to retrieve the tokens add
during the tokenizer's
tokens from a specific

Here's an example of h

You
And how would i now use prepare_for_tokenization?

ChatGPT
The `prepare_for_tokenization` method in Huggingface's Transformers library allows you to preprocess text before tokenization. This method is useful when you want
e tokenizing the text using the
_for_tokenization` method:

You
I don't understand the third line

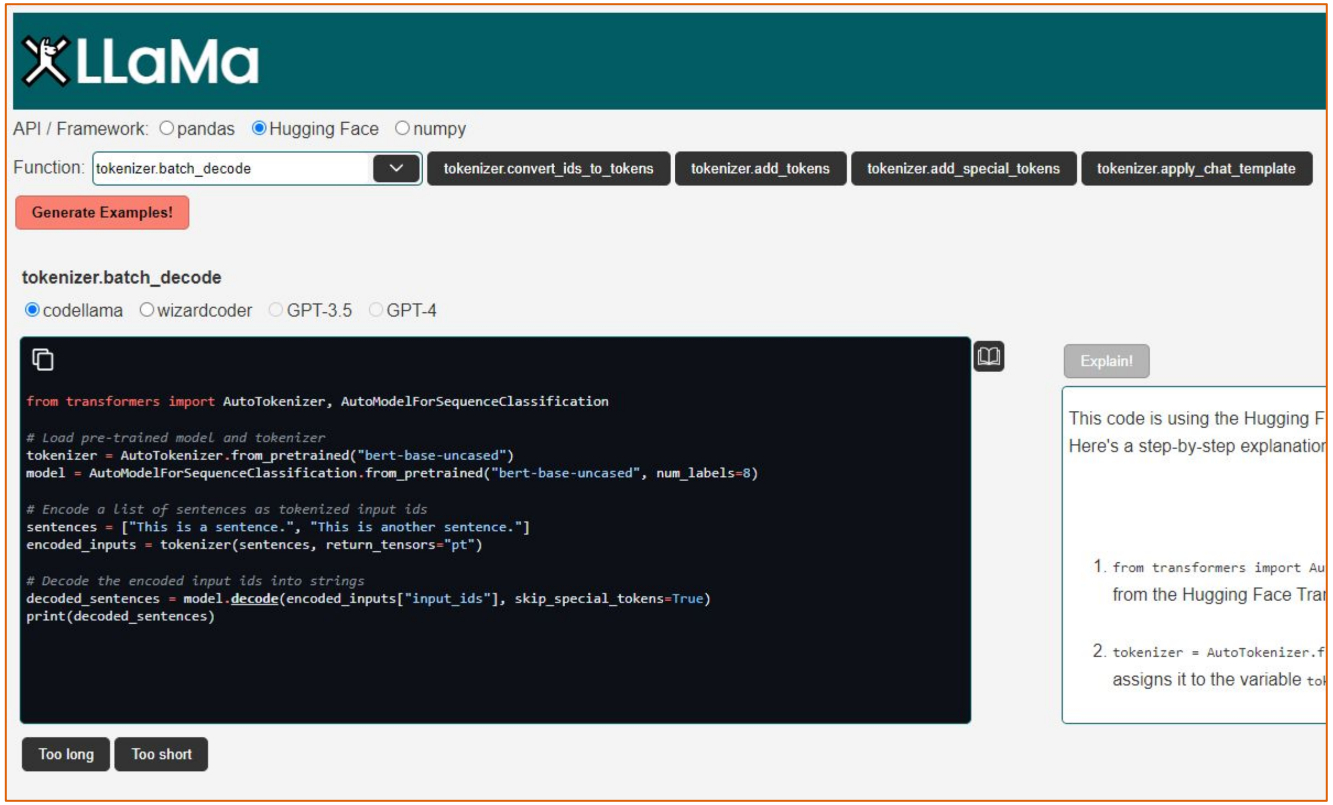
ChatGPT
My apologies for the confusion. Let me clarify the third line of code.

APIs

- Some API references are very detailed and helpful, others are not ✅

Future Work

- Add entire APIs to database
- Add Context integration
- Add GPT or any other LLM support
- Collect and use user feedback


xLLaMa
tokenizer.convert_ids_to_tokens
tokenizer.add_tokens
tokenizer.add_special_tokens
tokenizer.apply_chat_template
Generate Examples!
Explain!
Too long
Too short

API / Framework: ○ pandas ● Hugging Face ○ numpy

Function: tokenizer.batch_decode

tokenizer.batch_decode

● codellama ○ wizardcoder ○ GPT-3.5 ○ GPT-4

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification

# Load pre-trained model and tokenizer
tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
model = AutoModelForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=8)

# Encode a list of sentences as tokenized input ids
sentences = ["This is a sentence.", "This is another sentence."]
encoded_inputs = tokenizer(sentences, return_tensors="pt")

# Decode the encoded input ids into strings
decoded_sentences = model.decode(encoded_inputs["input_ids"], skip_special_tokens=True)
print(decoded_sentences)
```

This code is using the Hugging F
Here's a step-by-step explanation

1. `from transformers import Au`
   from the Hugging Face Tra
2. `tokenizer = AutoTokenizer.f`
   assigns it to the variable `tok`

Hasso
Plattner
Institut
Digital Engineering · Universität Potsdam

XLLaMa

Thank you for your attention!

Links:

http://delos.eaalab.hpi.uni-potsdam.de:8080/

https://github.com/CR1337/xLLaMa

Quellen

- https://pandas.pydata.org/docs/
- https://numpy.org/doc/
- https://huggingface.co/docs
- https://ollama.ai/
- https://huggingface.co/codellama/CodeLlama-70b-hf
- https://huggingface.co/WizardLM/WizardCoder-33B-V1.1
- https://chat.openai.com/