# LlamaParse

LlamaParse is a **GenAI-native document parser** that can parse complex document data for any downstream LLM use case (RAG, agents).

It is really good at the following:

- ✅ **Broad file type support**: Parsing a variety of unstructured file types (.pdf, .pptx, .docx, .xlsx, .html) with text, tables, visual elements, weird layouts, and more.
- ✅ **Table recognition**: Parsing embedded tables accurately into text and semi-structured representations.
- ✅ **Multimodal parsing and chunking**: Extracting visual elements (images/diagrams) into structured formats and return image chunks using the latest multimodal models.
- ✅ **Custom parsing**: Input custom prompt instructions to customize the output the way you want it.

LlamaParse directly integrates with LlamaIndex.

The free plan is up to 1000 pages a day. Paid plan is free 7k pages per week + 0.3c per additional page by default. There is a sandbox available to test the API https://cloud.llamaindex.ai/parse ↗.

Read below for some quickstart information, or see the full documentation.

If you're a company interested in enterprise RAG solutions, and/or high volume/on-prem usage of LlamaParse, come talk to us.

## Getting Started

First, login and get an api-key from [https://cloud.llamaindex.ai/api-key ↗](https://cloud.llamaindex.ai/api-key).

Then, install the package:

```
pip install llama-cloud-services
```

## CLI Usage

Now you can parse your first PDF file using the command line interface. Use the command `llama-parse [file_paths]`. See the help text with `llama-parse --help`.

```
export LLAMA_CLOUD_API_KEY='llx-...'

# output as text
llama-parse my_file.pdf --result-type text --output-file output.txt

# output as markdown
llama-parse my_file.pdf --result-type markdown --output-file output.m

# output as raw json
llama-parse my_file.pdf --output-raw-json --output-file output.json
```

## Python Usage

You can also create simple scripts:

```
from llama_cloud_services import LlamaParse

parser = LlamaParse(
    api_key="llx-...",  # can also be set in your env as LLAMA_CLOUD_
    num_workers=4,  # if multiple files passed, split in `num_workers
    verbose=True,
    language="en",  # Optionally you can define a language, default=e
)

# sync
result = parser.parse("./my_file.pdf")

# sync batch
results = parser.parse(["./my_file1.pdf", "./my_file2.pdf"])

# async
result = await parser.aparse("./my_file.pdf")
```

```
# async batch
results = await parser.aparse(["./my_file1.pdf", "./my_file2.pdf"])
```

The result object is a fully typed `JobResult` object, and you can interact with it to parse and transform various parts of the result:

```
# get the llama-index markdown documents
markdown_documents = result.get_markdown_documents(split_by_page=True

# get the llama-index text documents
text_documents = result.get_text_documents(split_by_page=False)

# get the image documents
image_documents = result.get_image_documents(
    include_screenshot_images=True,
    include_object_images=False,
    # Optional: download the images to a directory
    # (default is to return the image bytes in ImageDocument objects)
    image_download_dir="./images",
)

# access the raw job result
# Items will vary based on the parser configuration
for page in result.pages:
    print(page.text)
    print(page.md)
    print(page.images)
    print(page.layout)
    print(page.structuredData)
```

See more details about the result object in the example notebook.

## Using with file object / bytes

You can parse a file object directly:

```
from llama_cloud_services import LlamaParse

parser = LlamaParse(
    api_key="llx-...",  # can also be set in your env as LLAMA_CLOUD_
    num_workers=4,  # if multiple files passed, split in `num_workers
    verbose=True,
    language="en",  # Optionally you can define a language, default=e
)

file_name = "my_file1.pdf"
extra_info = {"file_name": file_name}

with open(f"./{file_name}", "rb") as f:
```

```
    # must provide extra_info with file_name key with passing file ob
    result = parser.parse(f, extra_info=extra_info)

# you can also pass file bytes directly
with open(f"./{file_name}", "rb") as f:
    file_bytes = f.read()
    # must provide extra_info with file_name key with passing file by
    result = parser.parse(file_bytes, extra_info=extra_info)
```

### Using with `SimpleDirectoryReader`

You can also integrate the parser as the default PDF loader in `SimpleDirectoryReader`:

```
from llama_cloud_services import LlamaParse
from llama_index.core import SimpleDirectoryReader

parser = LlamaParse(
    api_key="llx-...",  # can also be set in your env as LLAMA_CLOUD_
    result_type="markdown",  # "markdown" and "text" are available
    verbose=True,
)

file_extractor = {".pdf": parser}
documents = SimpleDirectoryReader(
    "./data", file_extractor=file_extractor
).load_data()
```

Full documentation for `SimpleDirectoryReader` can be found on the LlamaIndex Documentation.

## Examples



- Raw API Usage
- Result Object Tour

## Documentation

https://docs.cloud.llamaindex.ai/