

CLI Interface

```
docetl.cli.run(yaml_file=typer.Argument(..., help='Path to the  
YAML file containing the pipeline configuration'),  
max_threads=typer.Option(None, help='Maximum number of threads to  
use for running operations'))
```

Run the configuration specified in the YAML file.

Parameters:

Name	Type	Description	Default
yaml_file	Path	Path to the YAML file containing the pipeline configuration.	Argument(..., help='Path to the YAML file containing the pipeline configuration')
max_threads	int None	Maximum number of threads to use for running operations.	Option(None, help='Maximum number of threads to use for running operations')

Source code in `docetl/cli.py`

```

56 @app.command()
57 def run(
58     yaml_file: Path = typer.Argument(
59         ..., help="Path to the YAML file containing the pipeline
60 configuration"
61     ),
62     max_threads: int | None = typer.Option(
63         None, help="Maximum number of threads to use for running
64 operations"
65     ),
66 ):
67     """
68     Run the configuration specified in the YAML file.
69
70     Args:
71         yaml_file (Path): Path to the YAML file containing the pipeline
72 configuration.
73         max_threads (int | None): Maximum number of threads to use for
74 running operations.
75     """
76     # Get the current working directory (where the user called the
77 command)
78     cwd = os.getcwd()
79
80     # Load .env file from the current working directory
81     env_file = os.path.join(cwd, ".env")
82     if os.path.exists(env_file):
83         load_dotenv(env_file)
84
85     runner = DSLRunner.from_yaml(str(yaml_file), max_threads=max_threads)
86     runner.load_run_save()

```

```

docetl.cli.build(yaml_file=typer.Argument(..., help='Path to the
YAML file containing the pipeline configuration'),
max_threads=typer.Option(None, help='Maximum number of threads to
use for running operations'), resume=typer.Option(False,
help='Resume optimization from a previous build that may have
failed'), save_path=typer.Option(None, help='Path to save the
optimized pipeline configuration'))

```

Build and optimize the configuration specified in the YAML file. Any arguments passed here will override the values in the YAML file.

Parameters:

Name	Type	Description	Default
yaml_file	Path	Path to the YAML file containing the pipeline configuration.	Argument(..., help='Path to the YAML file containing the pipeline configuration')
max_threads	int None	Maximum number of threads to use for running operations.	Option(None, help='Maximum number of threads to use for running operations')
model	str	Model to use for optimization. Defaults to "gpt-4o".	<i>required</i>
resume	bool	Whether to resume optimization from a previous run. Defaults to False.	Option(False, help='Resume optimization from a previous build that may have failed')
save_path	Path	Path to save the optimized pipeline configuration.	Option(None, help='Path to save the optimized pipeline configuration')

Source code in `docetl/cli.py`

```

13 @app.command()
14 def build(
15     yaml_file: Path = typer.Argument(
16         ..., help="Path to the YAML file containing the pipeline
17 configuration"
18     ),
19     max_threads: int | None = typer.Option(
20         None, help="Maximum number of threads to use for running
21 operations"
22     ),
23     resume: bool = typer.Option(
24         False, help="Resume optimization from a previous build that may
25 have failed"
26     ),
27     save_path: Path = typer.Option(
28         None, help="Path to save the optimized pipeline configuration"
29     ),
30 ):
31     """
32     Build and optimize the configuration specified in the YAML file.
33     Any arguments passed here will override the values in the YAML file.
34
35     Args:
36         yaml_file (Path): Path to the YAML file containing the pipeline
37 configuration.
38         max_threads (int | None): Maximum number of threads to use for
39 running operations.
40         model (str): Model to use for optimization. Defaults to "gpt-4o".
41         resume (bool): Whether to resume optimization from a previous run.
42 Defaults to False.
43         save_path (Path): Path to save the optimized pipeline
44 configuration.
45     """
46     # Get the current working directory (where the user called the
47 command)
48     cwd = os.getcwd()
49
50     # Load .env file from the current working directory
51     env_file = os.path.join(cwd, ".env")
52     if os.path.exists(env_file):
53         load_dotenv(env_file)
54
55     runner = DSLRunner.from_yaml(str(yaml_file), max_threads=max_threads)
56     runner.optimize(
57         save=True,
58         return_pipeline=False,
59         resume=resume,
60         save_path=save_path,
61     )

```

`docetl.cli.clear_cache()`

Clear the LLM cache stored on disk.

Source code in `docetl/cli.py`

```
84 @app.command()
85 def clear_cache():
86     """
87     Clear the LLM cache stored on disk.
88     """
89     cc()
```