AIRFLOW SUMMIT

Custom Operators in Action:

A Guide to Extending Airflow's Capabilities.

Shalabh Agarwal
Senior Software Engineer @ Walmart

Custom Operators in Action: A Guide to Extending Airflow's Capabilities

Empowering data engineers to build maintainable, reusable solutions for complex workflow challenges

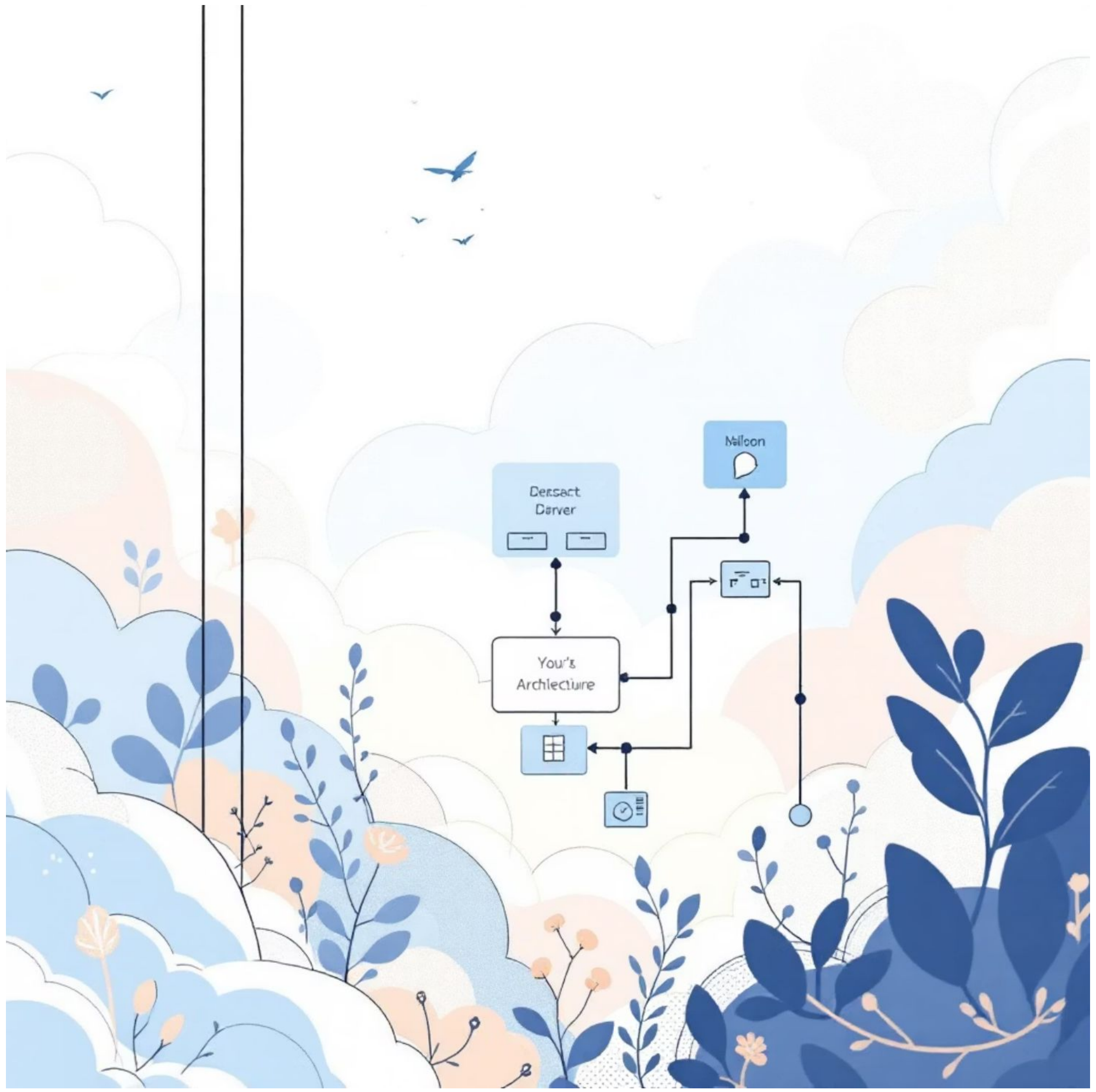
Agenda

Why Custom Operators Matter

Beyond Built-in Limitations

Airflow's extensive operator library covers most use cases, but enterprise workflows often require specialised logic that doesn't fit standard patterns.

Custom operators bridge the gap between generic functionality and business-specific requirements, ensuring your DAGs remain clean and maintainable.



When to Build vs. Buy

Evaluate Existing Solutions

Check Airflow providers, community plugins, and third-party packages first

Assess Customisation Needs

Can existing operators be extended or configured to meet requirements?

Build Custom Solution

Create when business logic is unique, reusable, or requires specific integrations

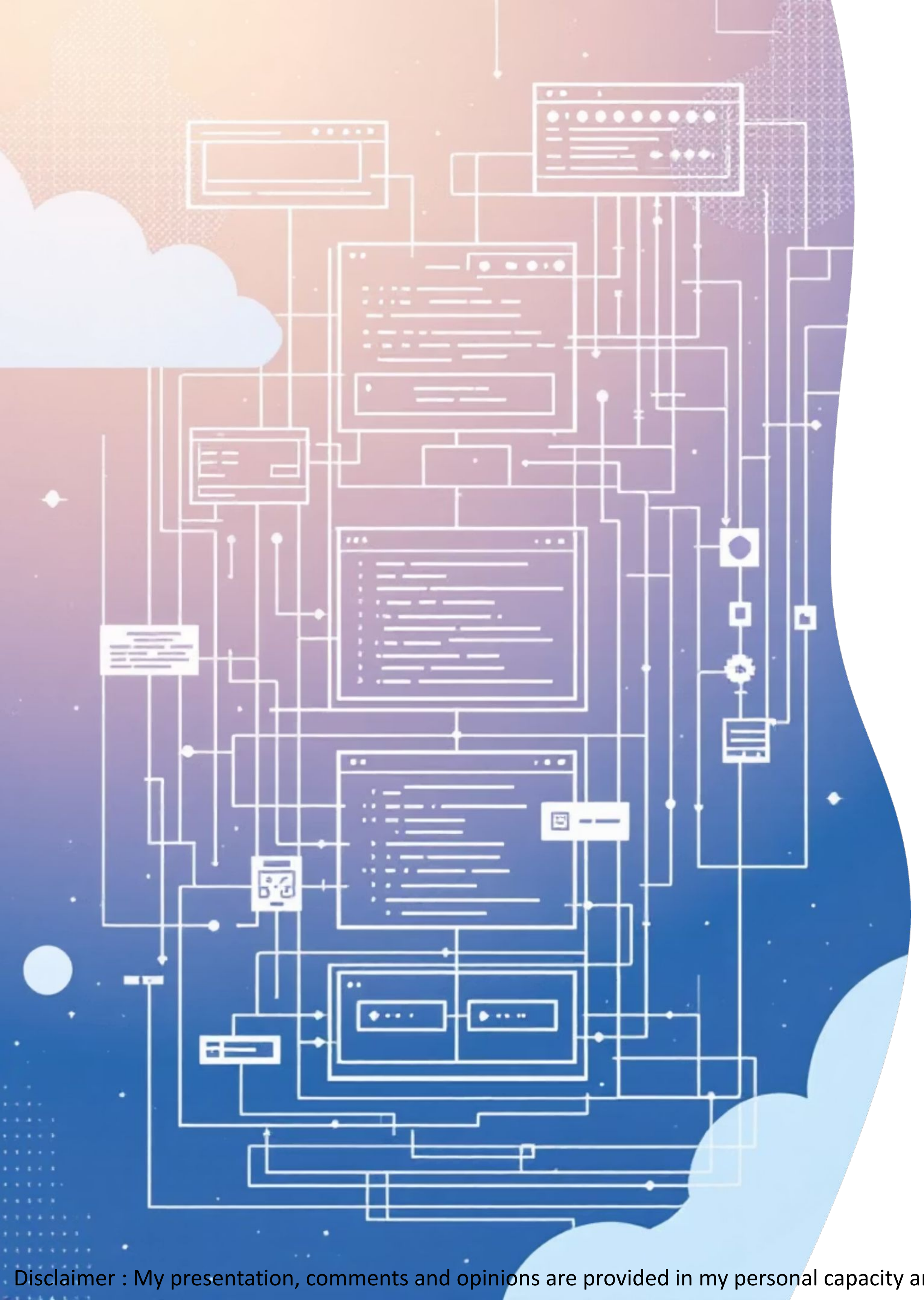
Decision Framework

✅ Build Custom When

- Complex business logic spans multiple tasks
- Proprietary system integrations required
- Repeated patterns across multiple DAGs
- Performance optimisation needed

❌ Use Existing When

- Standard operations suffice
- One-off requirements
- Tight delivery timelines
- Limited maintenance resources



Architecture Patterns for Maintainable Operators

01

Single Responsibility Principle

Each operator should handle one specific task or business function

02

Configuration-Driven Design

Expose parameters through constructor arguments for flexibility

03

Error Handling & Logging

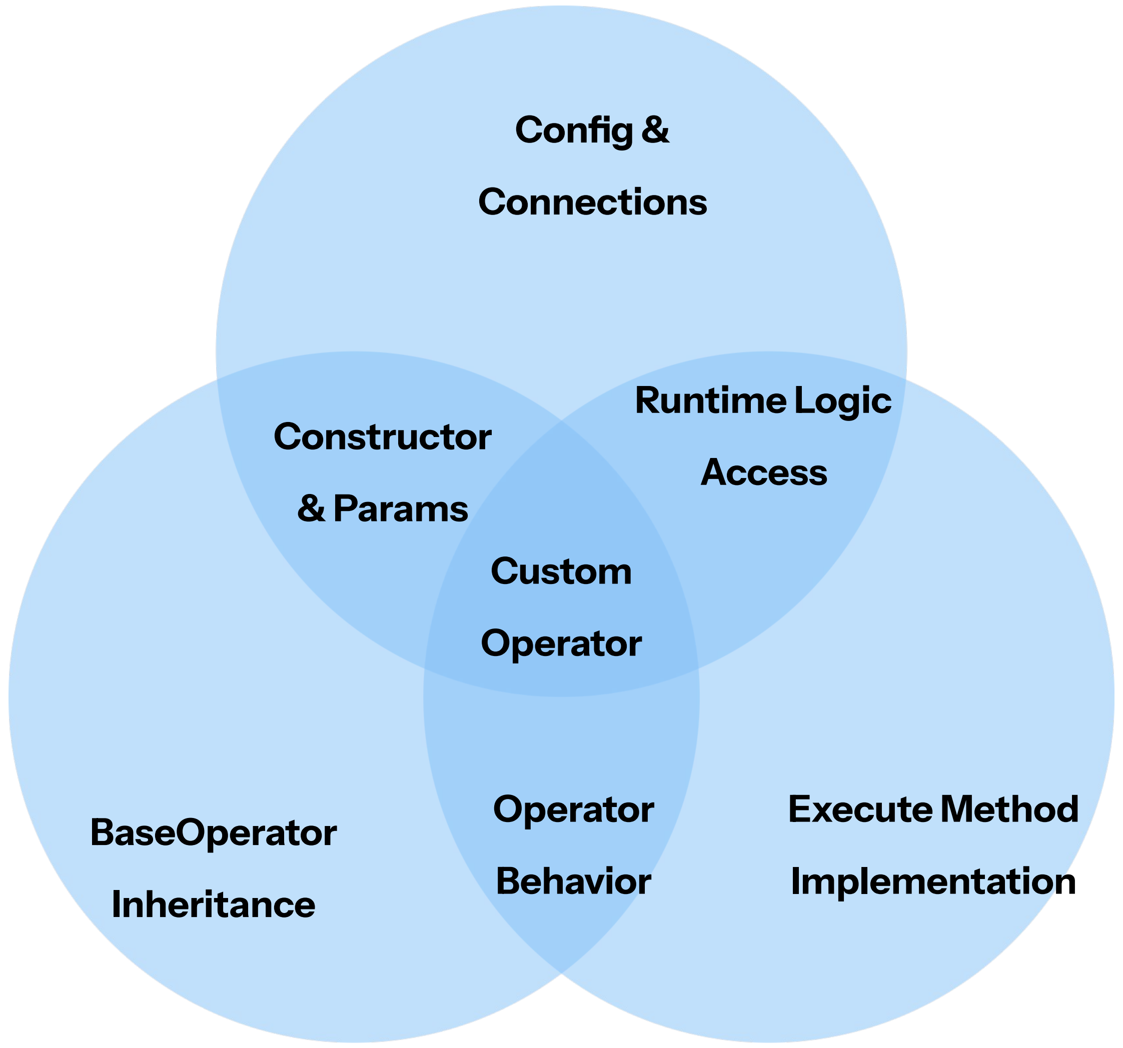
Implement comprehensive error handling with meaningful log messages

04

Testing & Documentation

Include unit tests and clear documentation for future maintenance

Core Architecture Components

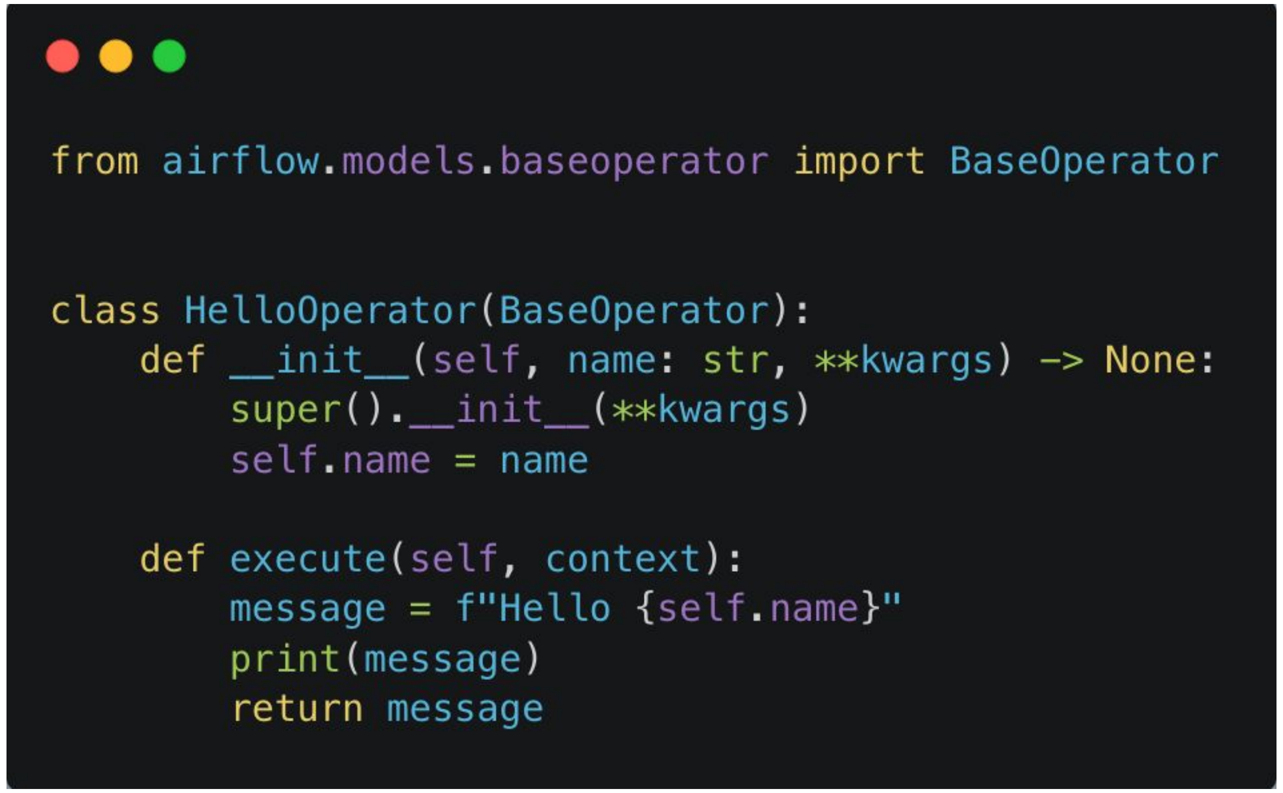


Custom operators inherit from BaseOperator and implement the execute method, whilst managing connections and configuration through Airflow's standard patterns.

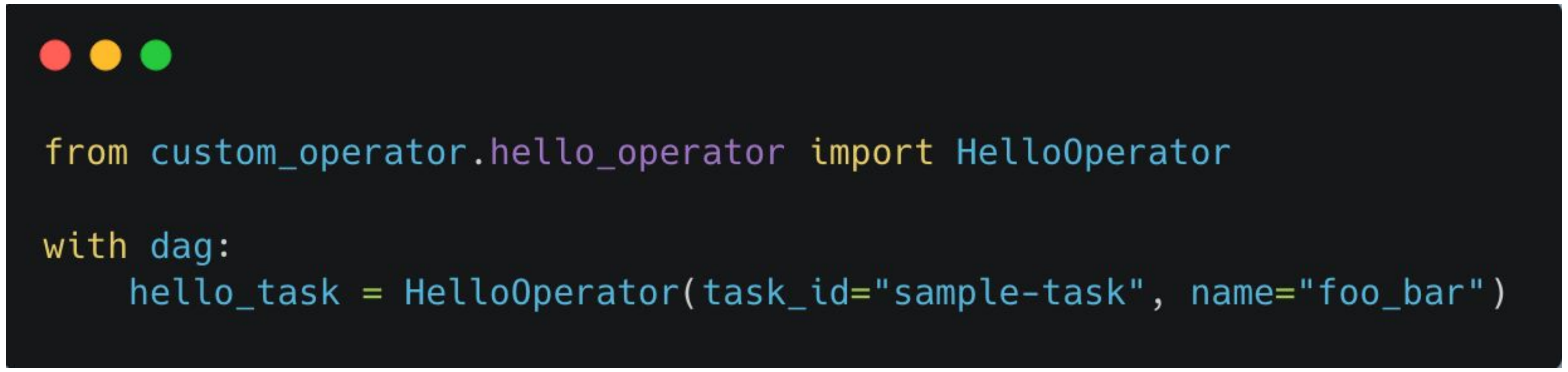
How do I create it ?

1. Custom Hello World Operator
2. Custom File Validation Operator

Custom Hello World Operator

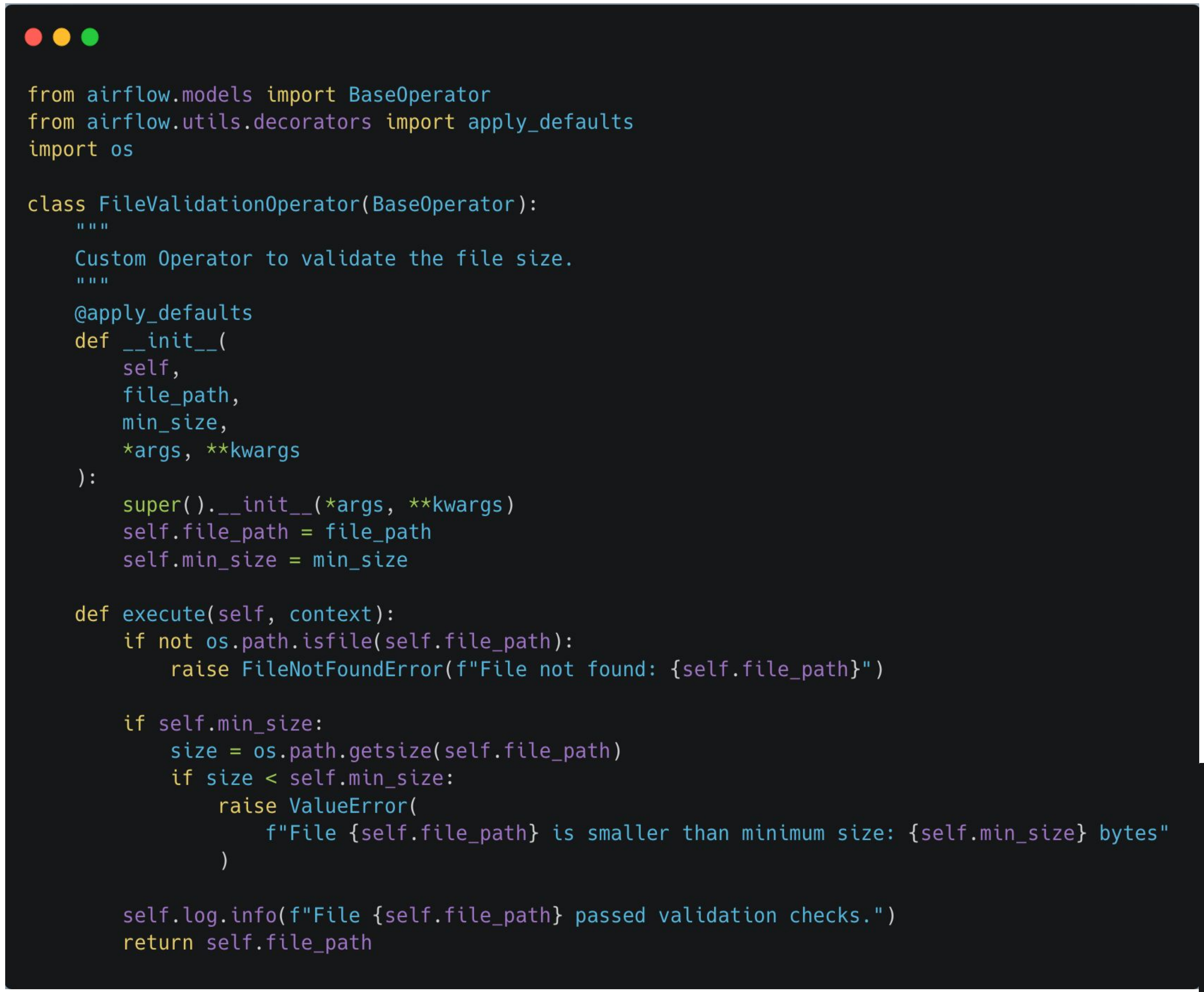


```python
from airflow.models.baseoperator import BaseOperator


class HelloOperator(BaseOperator):
    def __init__(self, name: str, **kwargs) -> None:
        super().__init__(**kwargs)
        self.name = name

    def execute(self, context):
        message = f"Hello {self.name}"
        print(message)
        return message
```

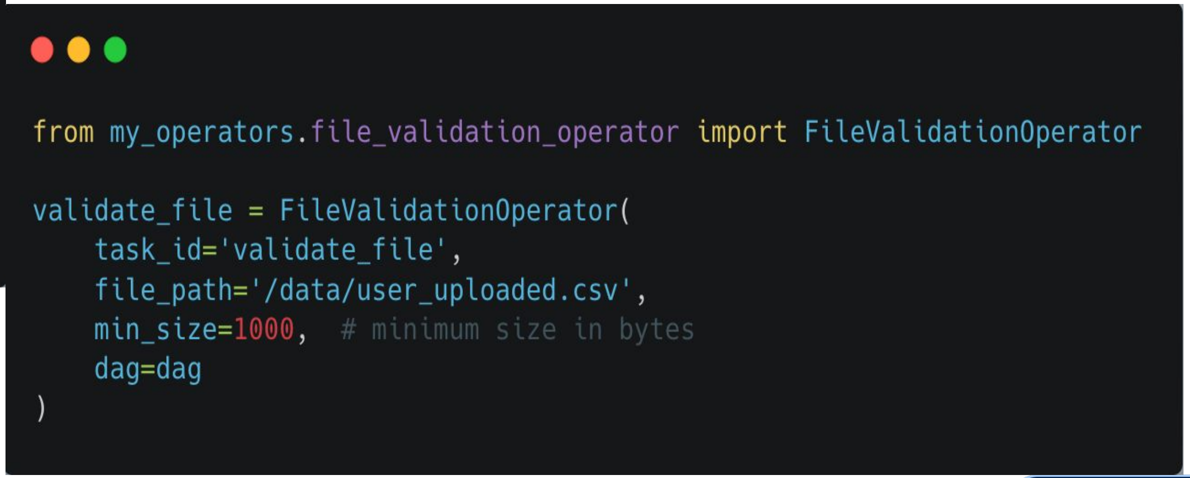


```python
from custom_operator.hello_operator import HelloOperator

with dag:
    hello_task = HelloOperator(task_id="sample-task", name="foo_bar")
```

Custom File Validation Operator



```python
from airflow.models import BaseOperator
from airflow.utils.decorators import apply_defaults
import os

class FileValidationOperator(BaseOperator):
    """
    Custom Operator to validate the file size.
    """
    @apply_defaults
    def __init__(
        self,
        file_path,
        min_size,
        *args, **kwargs
    ):
        super().__init__(*args, **kwargs)
        self.file_path = file_path
        self.min_size = min_size

    def execute(self, context):
        if not os.path.isfile(self.file_path):
            raise FileNotFoundError(f"File not found: {self.file_path}")

        if self.min_size:
            size = os.path.getsize(self.file_path)
            if size < self.min_size:
                raise ValueError(
                    f"File {self.file_path} is smaller than minimum size: {self.min_size} bytes"
                )

        self.log.info(f"File {self.file_path} passed validation checks.")
        return self.file_path
```



```python
from my_operators.file_validation_operator import FileValidationOperator

validate_file = FileValidationOperator(
    task_id='validate_file',
    file_path='/data/user_uploaded.csv',
    min_size=1000,  # minimum size in bytes
    dag=dag
)
```

Made with GAMMA

Disclaimer : My presentation, comments and opinions are provided in my personal capacity and not as a representative of Walmart. They do not reflect the views of Walmart and are not endorsed by Walmart.

Real-World Use Cases

Data Quality Operator

Custom operators for data quality checks, reducing manual validation time and catching data issues as early as possible.

API Integration Suite

Creating reusable operators for third-party API interactions, standardising error handling and retry logic across DAGs.

ML Model Deployment

Operators for model versioning and deployment, enabling automated ML pipeline management with audit trails.

Key Takeaways

Strategic Decision Making

Evaluate existing solutions thoroughly before building custom operators

Follow Best Practices

Implement maintainable architecture patterns from day one

Start Simple, Scale Smart

Begin with focused operators and expand functionality based on real needs

Ready to extend your Airflow capabilities? Start building!

Thank You!

Shalabh Agarwal

Senior Data Engineer @ Walmart | Pythonista | Big Data Engineering | ...